目录

1. 相关技术

1.1 Java 语言与 Android 组件

1.1.1 Java 语言

本次课程大作业使用 Java 语言开发，鉴于安卓应用程序支持 Kotlin 和 Java 两种语言，且我个人对 Java 语言相比 Kotlin 更为熟悉一些，我最终选择了 Java 语言（与之前完成作业的选择相同）。

Java 是一门面向对象的语言，对于开发模块化的程序十分适合。同时，Java 语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

鉴于以上 Java 语言的种种优势，本次课程大作业选择了 Java 语言进行安卓开发。

1.1.2 Android 相关组件

Android 提供了多种多样的组件，本次课程作业几乎使用了本学期所讲述的所有在安卓开发过程中涉及的基础组件。包括但不限于如下的组件。

- Activity
- Intent
- Bundle
- Parcel
- TextView
- Button
- ImageView
- LinearLayout
- ConstraintLayout
- Framelayout
- Fragment
- ViewPager

- 图 1：Java 与 Android

1.2 计算机视觉 CV

全世界在近几年间见证了机器学习技术在计算机视觉领域的成熟应用，相关模型的提出（如各种优化的卷积神经网络，Vision Transformer）进一步推动了机器视觉的发展，很多研究成果已经在我们日常生活中得到了广泛及充分的应用。

计算机视觉的成熟发展其实很应该和安卓开发关联起来，安卓应用可以很容易获取到相机的权限后进行拍照，录像等任务，解决了在传统 PC 电脑上摄像头的移动性较差的问题，因此本次课程大作业我是用了计算机视觉的相关技术（如图像分类，图像增强等）与我的安卓应用结合。

图 2：一个目标检测和分类识别的简单例子

1.2.1 计算机视觉概述

计算机视觉的目标是对环境的表达和理解，核心问题是研究如何对输入的图像信息进行组织，对物体和场景进行识别，进而对图像内容给予解释。计算机视觉的发展主要经历了马尔计算机视觉，主动和目的视觉，多视几何和分层三维重建以及目前正火的基于学习的视觉。根据视觉任务的不同，计算机视觉可以主要包括以下方向（各个部分可能相互之间有些重叠）：

- 图像分类识别
- 目标检测
- 目标跟踪
- 图像分割
- GAN
- 图像滤波与降噪
- 图像增强
- 图像风格化
- 图像检索
- 三维视觉：三维感知，位姿估计(SLAM)，三维重建，三维理解
- 视频理解
- 多模态融合

1.2.2 与本次课程作业相关机器视觉内容详述

鉴于截止目前为止，个人对三维视觉了解并不多，本次课程大作业主要使用了二维机器视觉，针对在二维机器视觉中的经典任务：图像分类识别（现今以发展非常成熟，在多种卷积神经网络提出，以及集成学习的方法将各种模型综合后，针对分类任务的准确率可以达到几乎 100%，如 MNIST 数据集等）进行了实现。因本次作业主要服务的对象是针对人眼视觉稍有欠缺的用户（如色盲，色弱等），利用相关的技术可以很好解决对应用户的需求，同时根据我个人的调研，目前市面上也较缺少相关的应用，可以让色盲，色弱人群不借助他人的帮助就可以自己解决一些生活中的问题。

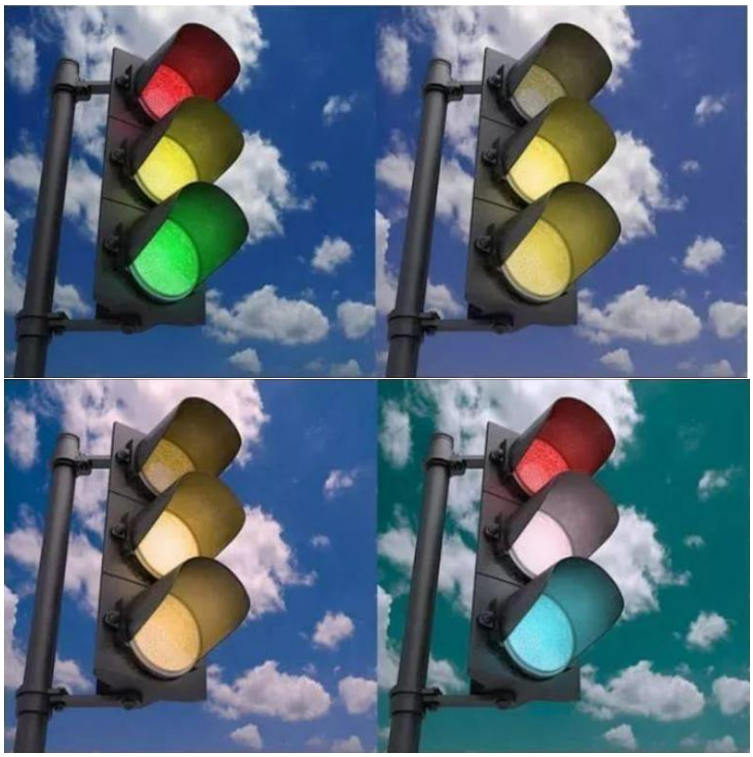

图 3：左上为正常人眼中的红绿灯，右上为红色盲，左下为绿色盲，右下为蓝色盲

相关问题如，红绿色盲无法分辨街道中的是红灯还是绿灯(如今红绿灯的形态多种多样)，或者蓝橙色盲有相关的需求等，都可以借助本次个人课程大作业完成的 app 实现。

本次课程大作业完成的 app 支持颜色识别与分类，以及针对摄像机中的场景进行多种视觉增强的功能，可以适合多种较为普遍的色盲，色弱人群使用。

1.3 OpenGL 图形库

OpenGL 是用于渲染 2D、3D 矢量图形的跨语言、跨平台的应用程序编程接口（API）。这个接口由近 350 个不同的函数调用组成，用来绘制从简单的图形比特到复杂的三维景象。而另一种程序接口系统是仅用于 Microsoft Windows 上的 Direct3D。OpenGL 常用于 CAD、虚拟现实、科学可视化程序和电子游戏开发。

OpenGL 的高效实现（利用了图形加速硬件）存在于 Windows，部分 UNIX 平台和 Mac OS。OpenGL 规范由 1992 年成立的 OpenGL 架构评审委员会（ARB）维护。ARB 由一些对创建一个统一的、普遍可用的 API 特别感兴趣的公司组成。

1.3.1 OpenGL 规范

OpenGL 规范描述了绘制 2D 和 3D 图形的抽象 API。尽管这些 API 可以完全通过软件实现，但它是为大部分或者全部使用硬件加速而设计的。

OpenGL 的 API 定义了若干可被客户端程序调用的函数，以及一些具名整型常量（例如，常量 GL_TEXTURE_2D 对应的十进制整数为 3553）。虽然这些函数的定义表面上类似于 C 编程语言，但它们是语言独立的。因此，OpenGL 有许多语言绑定，值得一提的包括：JavaScript 绑定的 WebGL（基于 OpenGL ES 2.0 在 Web 浏览器中的进行 3D 渲染的 API）；C 绑定的 WGL、GLX 和 CGL；iOS 提供的 C 绑定；Android 提供的 Java 和 C 绑定。

除了核心 API 要求的功能之外，GPU 供应商可以通过扩展的形式提供额外功能。扩展可能会引入新功能和新常量，并且可能放松或取消现有的 OpenGL 函数的限制。然后一个扩展就分成两部分发布：包含扩展函数原型的头文件和作为厂商的设备驱动。供应商使用扩展公开自定义的 API 而无需获得其他供应商或 Khronos Group 的支持，这大大增加了 OpenGL 的灵活性。

1.3.2 OpenGL 相关特点

OpenGL 不仅语言无关，而且平台无关。规范只字未提获得和管理 OpenGL 上下文相关的内容，而是将这些作为细节交给底层的窗口系统。出于同样的原因，OpenGL 纯粹专注于渲染，而不提供输入、音频以及窗口相关的 API。

OpenGL 具有良好的结构，直观的设计和逻辑命令。与其他图形程序包相比，OpenGL 只有很少的代码，因此执行速度高。另外 OpenGL 封装了有关基本硬件的信息，使得开发者无需针对具体的硬件特征进行设计。因此针对本次的课程大作业的视觉增强部分，OpenGL 图形库是一个很好的开发选择。

1.3.3 与本次课程作业内容相关的 OpenGL 库内容详述

本次课程大作业主要使用了 OpenGL 库的 Fragment Shader File 来对摄像头捕获的帧进行处理，以达到视觉增强的效果。

下面是软件中视觉增强功能的一个简单示例：

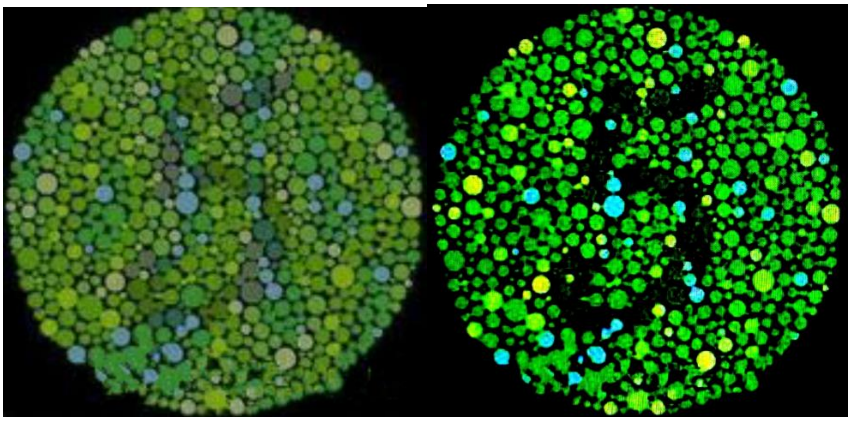

图 4：左图为图像原始效果，右图为图像增强后效果

可以看到，视觉增强后图中的数字 5 变得更加清晰了。

1.4 课程大作业创新点

其实因为安卓应用多年的发展,很多应用都有开发者完成了,而且大多数完成度都很高,例如记账本,二维码应用等,但在我的调研过程中,发现其实现阶段的应用都是面向于大众的,其实原因倒也不难理解,用户基数多才能变现吧。

因我有一个从小对颜色不敏感的朋友,因此我本次课程大作业做的应用便是面向色盲以及色弱等有视力缺陷问题人群的。

在调研过程中,我发现各大应用市场其实都没有一个成熟或者是用户较多的帮助视力缺陷人群的 app,如图 5。这也给我所做的应用提供了很好的创新点。

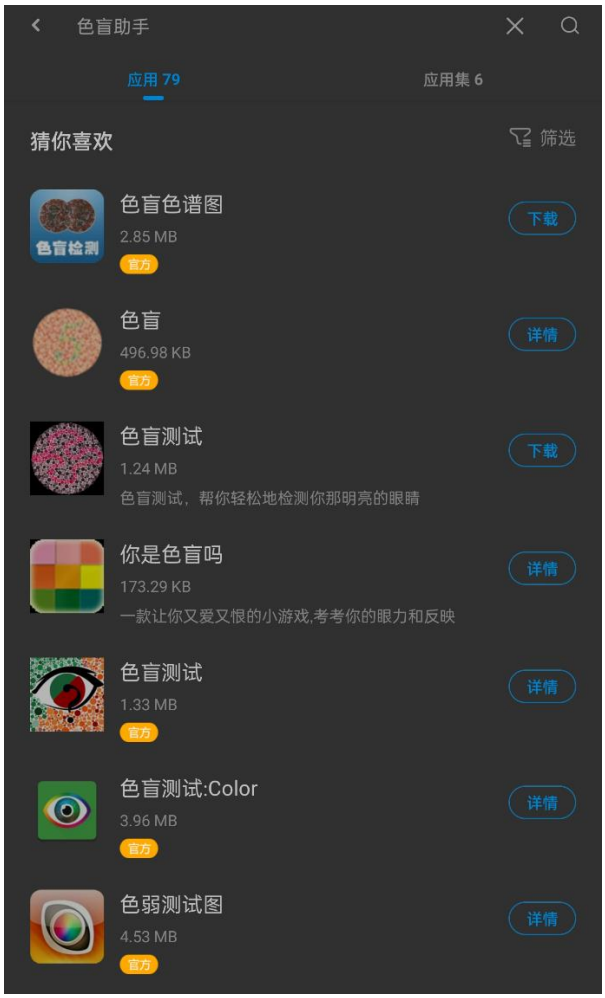

图 5：应用汇截图（多数为测试软件，可能较容易实现）

本次我完成的大作业可以让视力有缺陷人群在无需他人的帮助的情况下可以借助 app 实现一些他们本不可以做到的事情,比如帮助色盲人群识别颜色,或者是让色弱人群看到视觉增强后的实物。

基于以上所述,我本次大作业所做的应用是符号要求,且有一定实际意义的,且在市面上并没有找到很成熟的类似的安卓应用。

2. 系统功能需求

本次系统计划完成一个主要服务于视觉有缺陷人群（如色盲，色弱）的一个方便，快捷，有效的安卓 app——Sight，主要功能包含帮助识别物体颜色（如红绿灯），对视觉进行多种形式的增强（如提升对比度，对图像颜色进行重构，物体边缘检测等多达 32 种视觉增强效果）。同时 app 具有很好的扩展性，随着 CV（计算机视觉）技术的日益成熟，在不久的将来可以添加更多的功能。（如包含但不限于对 3D 物体的视觉建模与增强等）

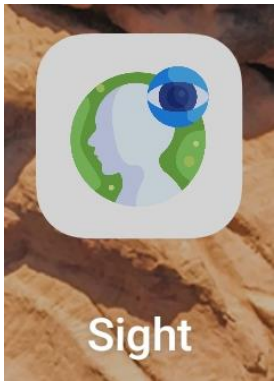

图 6：开发的安卓应用—— Sight

2.1 Sight 功能描述

Sight 作为一个服务于视觉有缺陷人群（如色盲，色弱）的 app，主要提供了两个功能：颜色识别和视觉增强。

2.1.1 Sight 颜色识别

Sight 可以获取手机的摄像机权限，对用户所需要进行识别的颜色进行拍摄，拍摄完成后即可得到系统分析得出的颜色名反馈给用户，几乎没有任何延迟。

2.1.2 Sight 视觉增强

Sight 可以获取手机的摄像机权限，生成实时的图像，针对用户选择的视觉增强的模式，实时的将所需视觉增强的画面反馈给用户，几乎没有任何延迟。

Sight 目前支持的视觉增强模式多达 32 种，可以满足用户的所有基本需求。

2.1.3 Sight 其他

Sight 预留了很多扩展空间，如在将来可以轻易的扩展更多的视觉增强模式，或者除了颜色识别之外的其他的物体识别等。

2.2 Sight 需求分析

2.2.1 Sight 环境需求分析

在硬件方面，Sight 需支持市面上几乎所有的智能机型使用，手机需要具有摄像头。

在软件方面，Sight 需可至少应用于安卓系统。

在使用方面，使用 Sight 的操作人员应无需了解相关的任何知识，Sight 的使用应该没有任何门槛，Sight 的交互环境应亲和度很高。

2.2.2 Sight 性能需求分析

针对 Sight 应用所完成的功能：如颜色识别或视觉增强，都应该是基于方便用户使用的前提完成的，因此 Sight 需要根据用户相应的交互动作进行快速的反应。具体的体现如用户提交了一张图片应该快速识别出图片中的颜色，用户选择的视觉增强模式应该快速转换为相应的模式并实时体现在画面中。

2.2.3 Sight 功能需求分析

Sight 作为一个服务于视觉有缺陷人群（如色盲，色弱）的 app，主要需要实现两个功能：颜色识别和视觉增强。

首先需要对用户行为进行描述：

这里只列举主要功能颜色识别和视觉增强，对于其他一些实现的功能如联系我们等不在此赘述。

用户使用颜色识别功能行为描述：

一个用户点击 Sight 应用图标，进入 app 界面。

App 需判断目前是否持有相关必须的设备权限，如果缺少相关权限则将获取权限的消息反馈给用户。

用户赋予 app 相关的权限。

App 界面显示出初始界面，界面包含多种功能选项供用户选择。

系统等待用户的下一步操作。

用户选择一个功能，系统打开相应的功能的界面。

如果是颜色识别功能的界面，应有拍照的按钮供用户使用，同时需要有界面显示颜色识别的结果。

用户点击拍照按钮，app 打开手机的摄像头，用户可以进行拍摄。

拍摄后用户可对拍摄结果进行预览，满足用户的需求可以选择确定，用户不满意也可以取消当前的结果，从新拍摄。

完成拍摄后系统应展现出显示颜色识别结果的界面，上面应显示出识别到的结果。

用户可以继续点击拍摄按钮进行接下来的识别，或者也可以退出当前功能界面，选择其他功能或退出 app。

用户使用模式增强功能行为描述：

一个用户点击 Sight 应用图标，进入 app 界面。

App 需判断目前是否持有相关必须的设备权限，如果缺少相关权限则将获取权限的消息

反馈给用户。

用户赋予 app 相关的权限。

App 界面显示出初始界面，界面包含多种功能选项供用户选择。

系统等待用户的下一步操作。

用户选择一个功能，系统打开相应的功能的界面。

如果是视觉增强功能的界面，应有显示实时的拍摄内容反馈给用户，如实时拍摄画面显示到屏幕上。

用户根据可选项中选择自己需要的视觉增强模式，同时也应该可以通过滑动的方式切换模式。根据用户选择到的模式系统应立即切换成相应模式的效果。

用户使用结束可以返回到初始界面使用 Sight 的其他功能，或者退出 Sight。

以上两种用户使用行为可以抽象成如下的系统活动图：

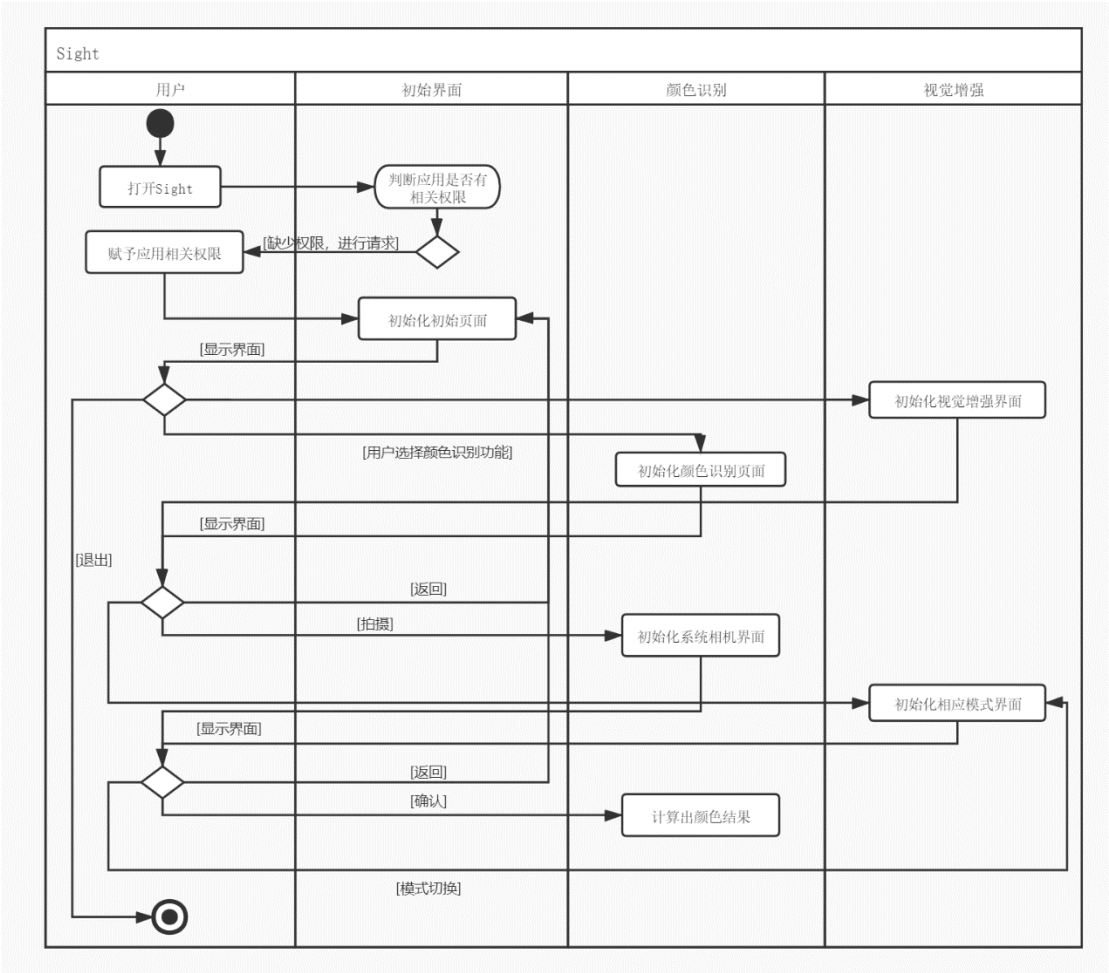

图 7：Sight 系统活动图

正如上面所说，这里只对主要功能进行需求分析，其他需完成的功能如联系我们，或者更多功能尽情期待等不在此过多赘述。

此外，可以从自然语言种抽象出多个类，可以为我们的 Sight 需求分析进行面向对象的需求分析，进行领域建模。

根据自然语言描述，识别出如下的概念类：

视觉增强，颜色识别，主页，模式选择，模式。
这里直接给出实现的软件类 uml 图：

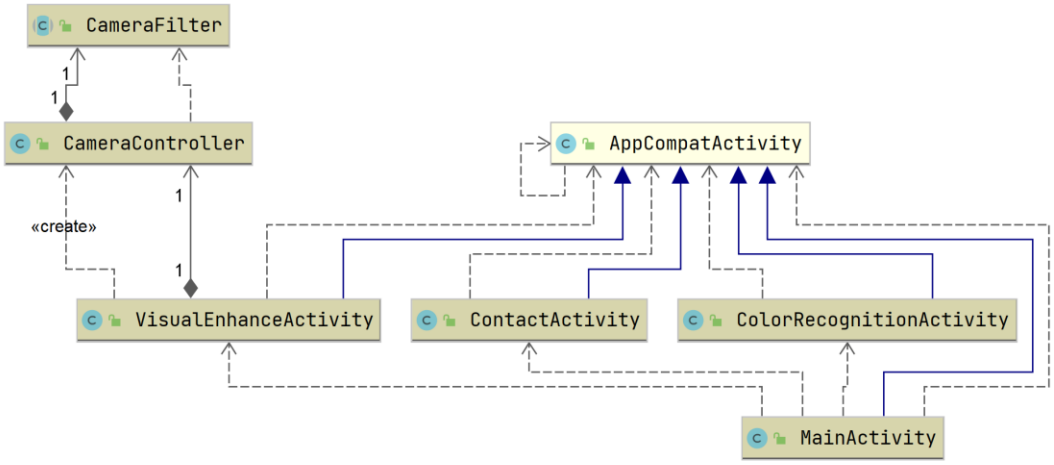

图 8：领域建模类图

其中 MainActivity 对应主页，VisualEnhanceActivity 对应视觉增强，ColorRecognitionActivity 对应颜色识别，CameraController 对应模式选择，CameraFilter 对应模式。

3. 系统设计与实现

3.1 Sight 总体设计

3.1.1 Sight 需求规定

根据 2.2.3 节的需求分析描述，Sight 系统需求如下：

- 交互：与用户进行交互，针对相应的用户动作产生不同的系统事件，给用户相应的反馈。
- 颜色识别：根据用户提供的图片，对图片进行分析，将图片输入模型中，得到模型的输出，将结果反馈给用户。
- 视觉增强：提供给用户多种视觉增强模式的选项，根据用户的选择切换成对应的视觉增强的效果。
- 联系我们：根据用户的点击操作，将相应的联系我们的信息提供给用户。
- 敬请期待：让用户可以显示的看到此界面，提示用户更多的功能会在将来开发出来，让用户可以通过联系我们给我们的工作提出建议，如需改进的点，或者相关的可以提升我们产品质量的地方。

3.1.2 Sight 运行环境

硬件环境：智能手机，具有摄像头

支持环境：安卓系统

图 9：Sight 运行环境

3.1.3 Sight 系统数据流图

Sight 系统的数据流图如下所示：

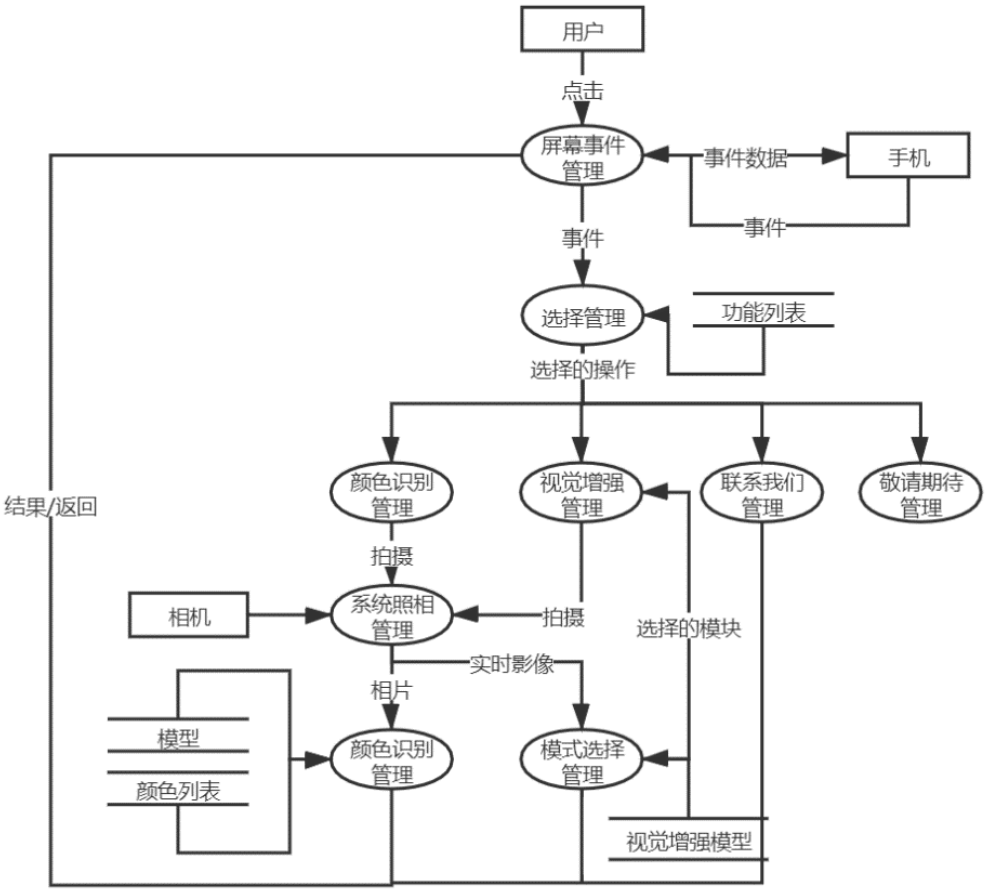

图 10：Sight 系统数据流图

可以看到，Sight 系统是事件驱动的软件，依据用户的相应动作，产生相应的系统事件并反馈给用户。

3.2 Sight 系统组成

Sight 系统组成图如下所示：

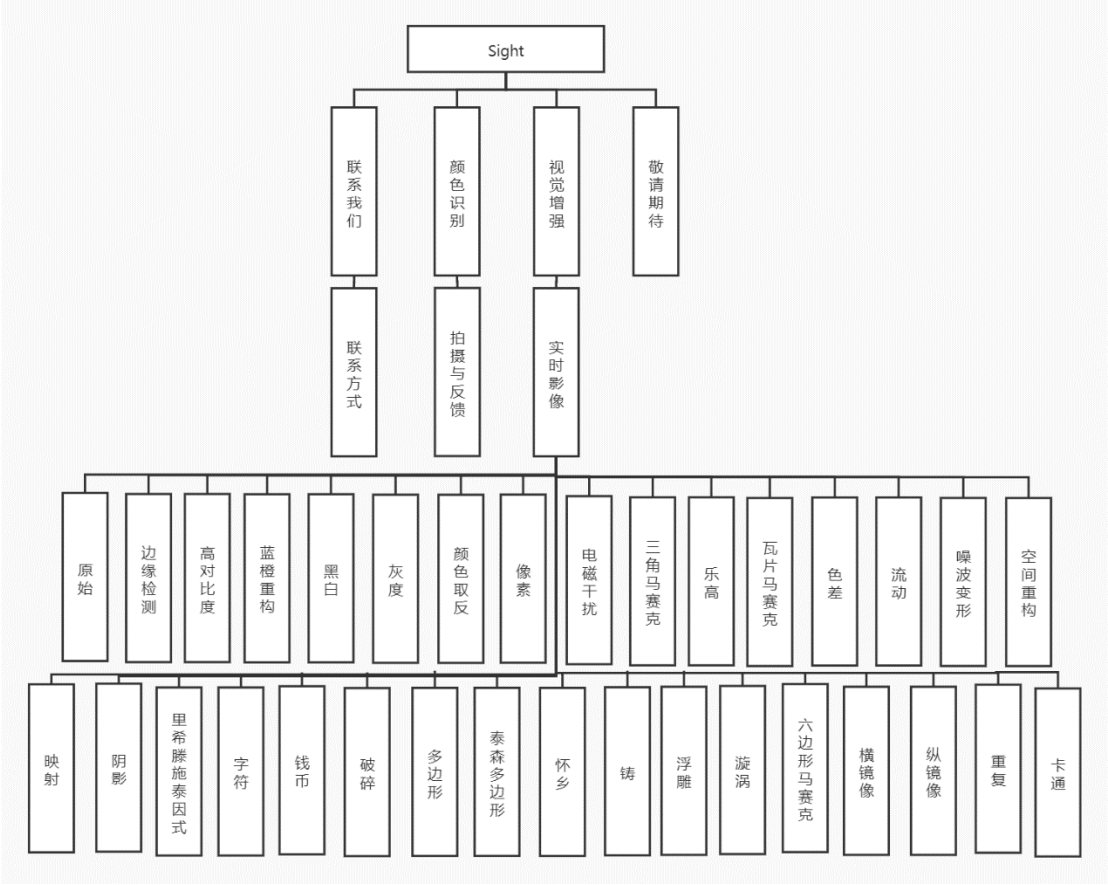

图 11：Sight 系统组成图

3.3 Sight 各模块设计与效果展示

Sight 基于面向对象的编程语言开发，程序高度模块化，按以下几个模块分别从模块逻辑设计，模块 ui 设计，模块程序设计，以及实现效果四方面进行阐述。

3.3.1 Sight 主模块

- **模块逻辑设计**

主模块是程序启动后遍调用的第一个模块，负责初始化初始界面，等待用户进行一些屏幕点击的操作，对用户的操作产生相应的系统事件，同时还负责初始化其他模块。

- **模块 ui 设计**

模块 ui 设置了四个 button，用户点击这四个 button 可以启动相应的分模块，进行进一步的操作，具体实现可以看实现效果的描述。

- **模块程序设计**

程序设计中使用 MainActivity 负责主模块的初始化，在 xml 文件编写号主模块的布局，对 onCreate 函数进行重写，可以在启动 MainActivity 的同时对页面初始化，获取程序所需的相应的权限。

- **实现效果**

Sight 主模块布局如下图所示：

图 12：Sight 主模块实现效果

3.3.2 Sight 颜色识别模块

- **模块逻辑设计**

用户在主模块选择颜色识别功能后，主模块会启动 ColorRecognitionActivity，这个 Activity 负责处理用户在颜色识别模块下的相关操作，ColorRecognitionActivity 初始化一个新的页面，专门服务用户使用颜色识别功能，ColorRecognitionActivity 会监控用户操作，一旦用户选择了拍摄按钮，则调用系统相机，用户可以进行拍摄。

此外，ColorRecognitionActivity 在此还负责导入模型，可以对用户拍摄的相片进行处理和分析得出颜色的结果反馈给用户。

- **模块 ui 设计**

此模块在颜色识别页面初始化一个 button，供给用户使用系统相机拍摄相片。

- **模块程序设计**

此模块由 MainActivity 创建一个新的 ColorRecognitionActivity，在 activity 目录下（具体程序代码架构可以查看 3.4 节关键代码解释，介绍了代码架构以及关键代码的详细解释），同样会调用一个编写好的 xml 布局文件对画面进行初始化。

- **实现效果**

Sight 颜色识别模块布局如下图所示：

图 13：Sight 颜色识别模块实现效果 1

点击按钮后的系统相机调用实现效果如下图所示：

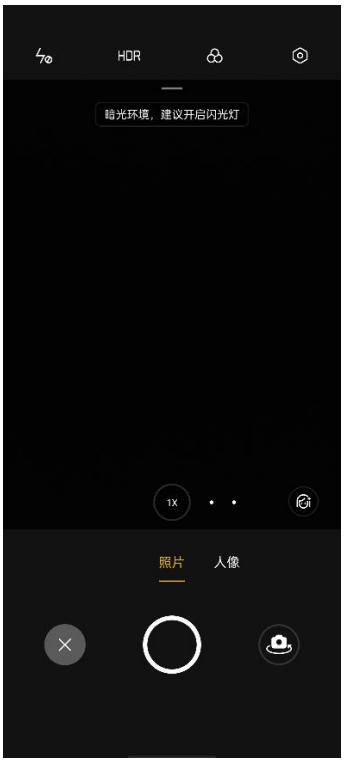

图 14：系统相机调用实现效果

对用户拍摄相片进行分析可以得出结果反馈给用户，如下图所示，识别出了红色：

图 15：结果反馈实现效果

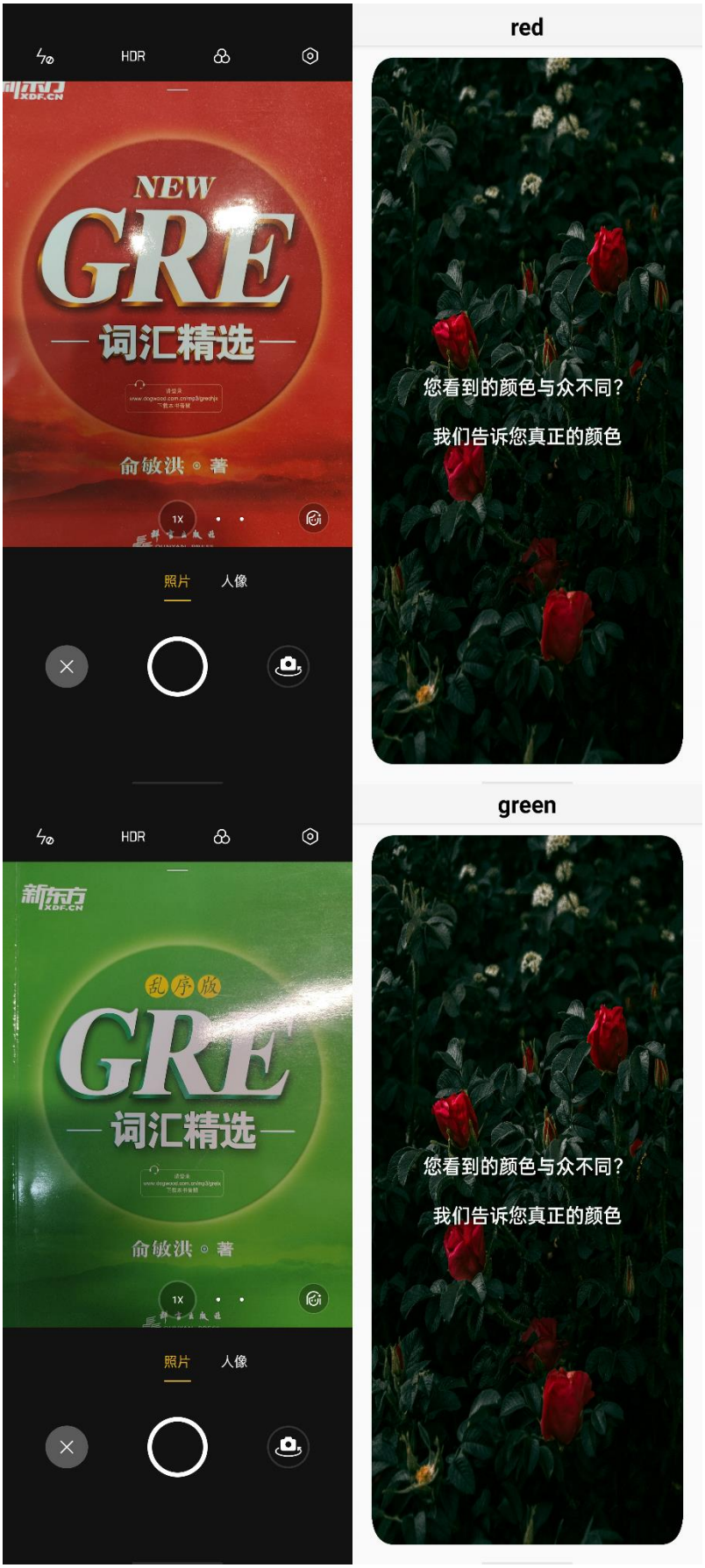
HDR
NEW
GRE
词汇精选
俞敏洪 著
照片
人像
HDR
乱序版
GRE
词汇精选
俞敏洪 著
照片
人像
red
您看到的颜色与众不同?
我们告诉您真正的颜色
green
您看到的颜色与众不同?
我们告诉您真正的颜色

图 16：一个使用示例

3.3.3 Sight 联系我们模块

- **模块逻辑设计**

用户在主模块选择联系我们功能后，主模块会启动 ContactActivity，这个 Activity 负责处理用户在联系我们模块下的相关操作，ContactActivity 初始化一个新的页面，专门服务用户使用联系我们功能，ContactActivity 会显示出我们的联系方式，地址，工作事件等信息。

- **模块 ui 设计**

此模块使用 TextView 以及 LinearLayout 完成。

- **模块程序设计**

此模块由 MainActivity 创建一个新的 ContactActivity，在 activity 目录下（具体程序代码架构可以查看 3.4 节关键代码解释，介绍了代码架构以及关键代码的详细解释），同样会调用一个编写好的 xml 布局文件对画面进行初始化。

- **实现效果**

联系我们模块实现效果如下所示：

图 17：联系我们模块实现效果

3.3.4 Sight 视觉增强模块

- 模块逻辑设计

用户在主模块选择视觉增强功能后，主模块会启动 VisualEnhanceActivity，这个 Activity 负责处理用户在视觉增强模块下的相关操作，VisualEnhanceActivity 初始化一个新的页面，专门服务用户使用视觉增强功能。

视觉增强功能提供给用户 30+种视觉增强模式，用户可以通过右上角的 menu 菜单进行选择，也可以直接滑动屏幕切换模式，VisualEnhanceActivity 会将摄像机拍摄的到实时画面显示到屏幕上，用户可以根据自己的需求进行使用。

- 模块 ui 设计

Sight 视觉增强模块 ui 设计使用 FrameLayout 完成，保存了 30+种 Frame，对拍摄到的图像进行处理，实时显示到用户手机屏幕上。与以上所述两种 ui 设计不同，鉴于此模块界面操作与切换的灵活性，此界面完全由代码生成，相关属性的设置也在相应的系统事件的相关反应下进行设置。

- 模块程序设计

此模块由 MainActivity 创建一个新的 VisualEnhanceActivity，在 activity 目录下（具体程序代码架构可以查看 3.4 节关键代码解释，介绍了代码架构以及关键代码的详细解释），同时会实例化 CameraController，CameraFilter，MyGLBuffer，MyOpenGL，分别用于对摄像机进行控制，以及实现相关的视觉增强效果操作，还有服务于特定视觉增强模式所需的 buffer 缓冲区相关控制功能，以及和图形库 OpenGL 的相关接口实现的 MyOpenGL。

具体的所有的视觉增强效果的实现存储在 filters 目录下，fileters 目录下的类均继承自 CameraFilter，通过简单的设置即可完成视觉增强的效果。

- 实现效果

原图效果如下：

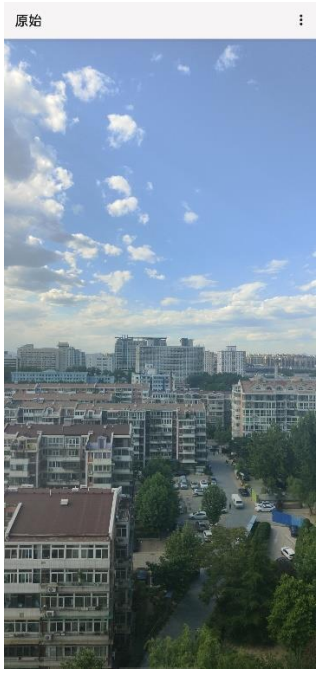

图 18：视觉增强模块下的原图

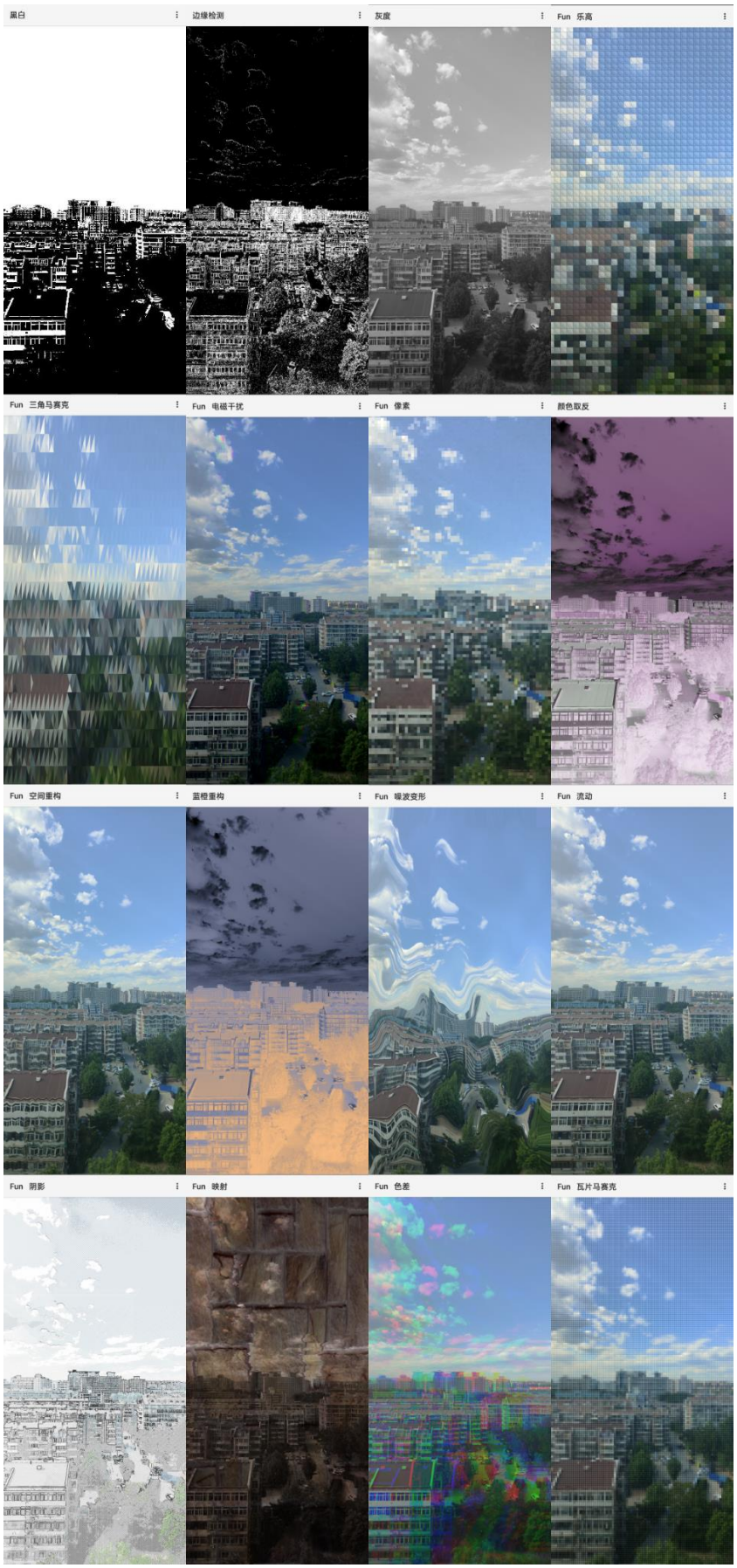

图 19：部分视觉增强效果展示

3.4 Sight 关键代码解释

3.4.1 Sight 代码架构

Sight 代码架构总览图如下所示（非常的庞大，因此接下来会拆解并解释）：

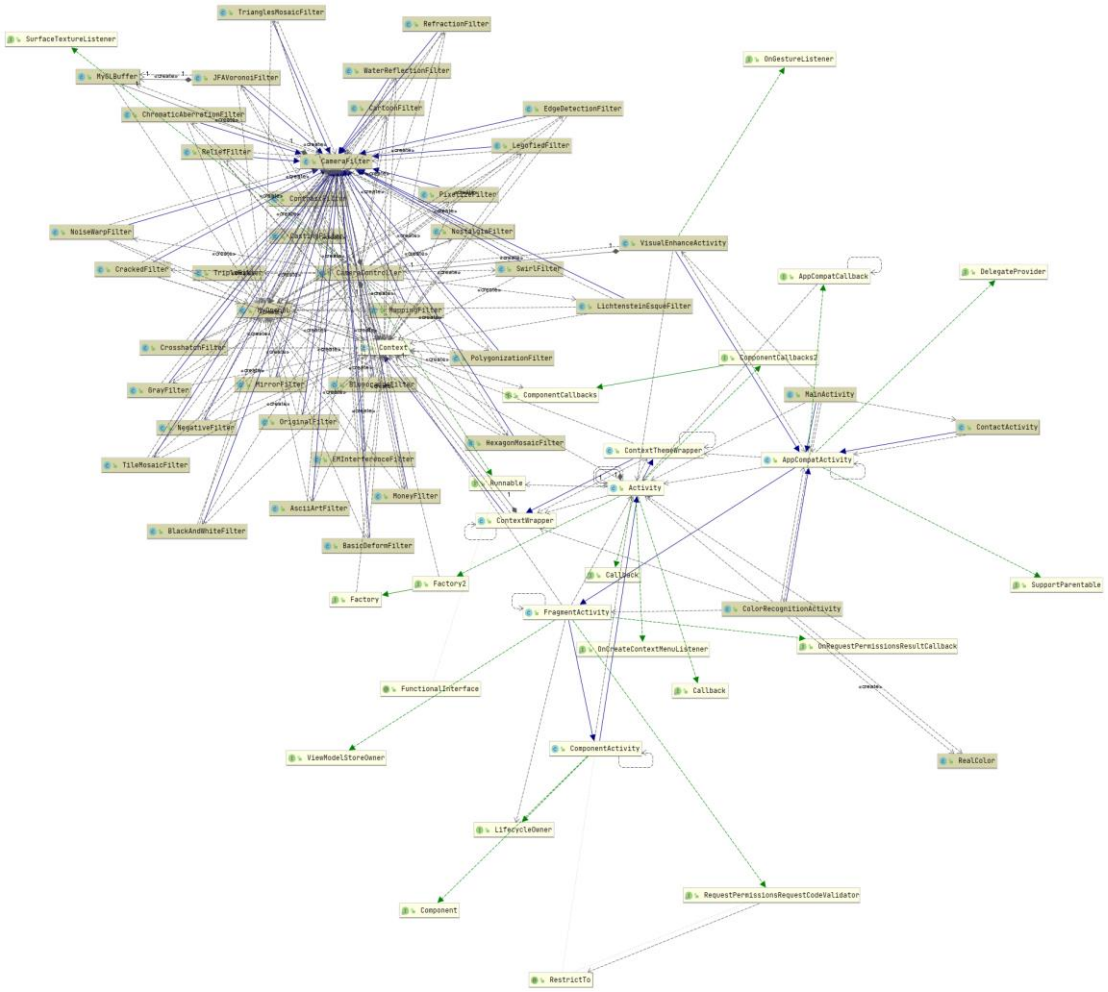

图 20：代码设计 UML 架构图

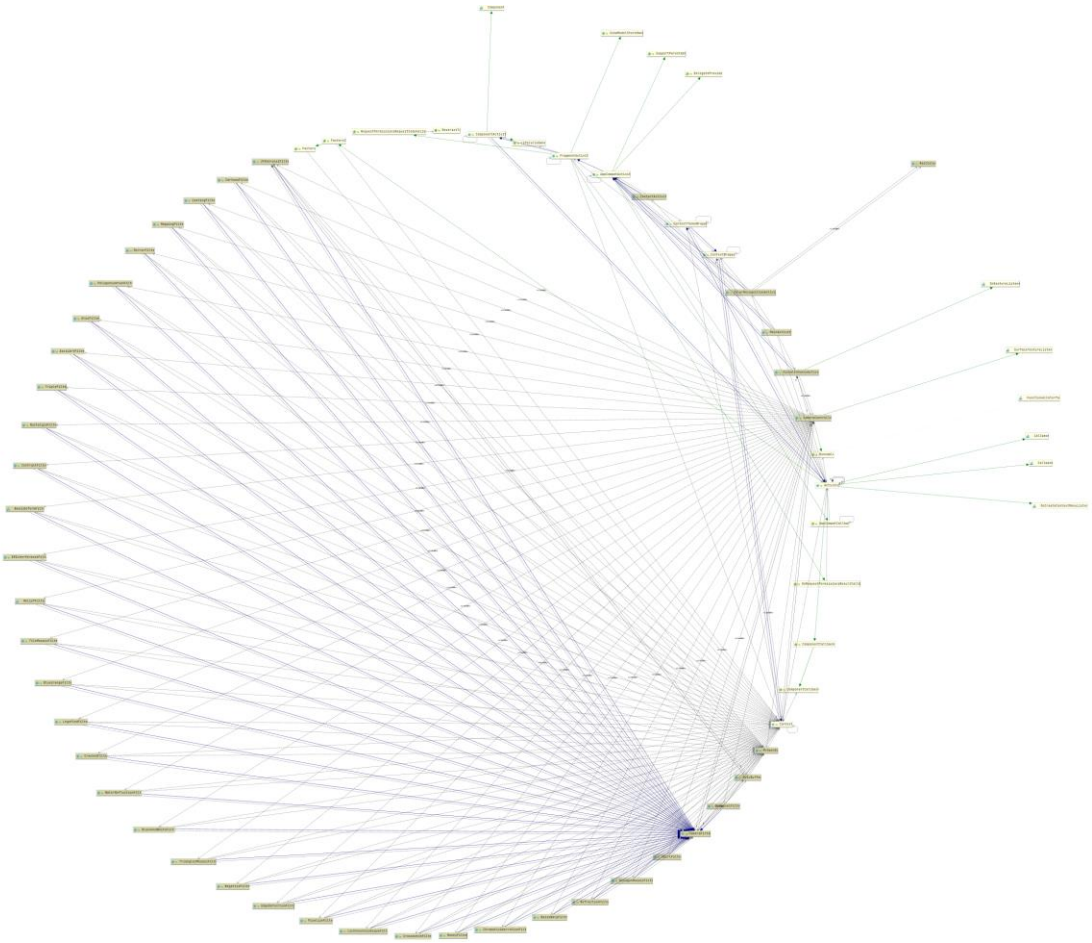

图 21：代码设计 UML 架构图（另一种布局）

3.4.2 Sight 主模块代码解释

主模块代码架构图如下：

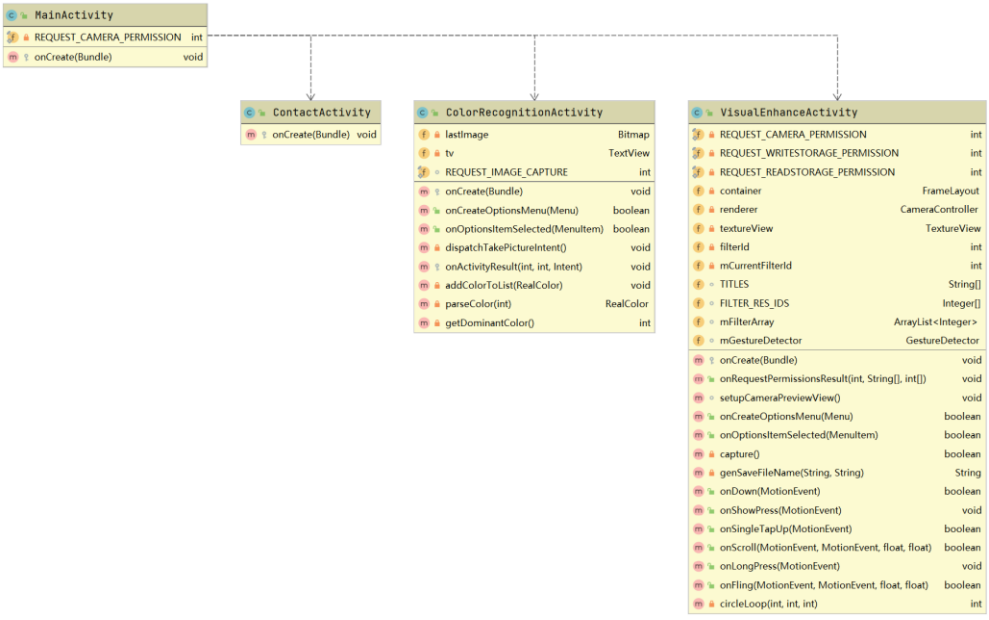

图 22：主模块代码架构

由图可知，主模块的 MainActivity 通过调用 startActivity 方法，在主模块可以启动其余的 Activity。

相关代码如下：

```
// 获取 button 的点击事件，开启颜色识别 ColorRecognitionActivity
findViewById(R.id.CR).setOnClickListener(v -> {
      startActivity(new Intent(MainActivity.this, ColorRecognitionActivity.class));
});

// 获取 button 的点击事件，开启视觉强化 VisualEnhanceActivity
findViewById(R.id.VE).setOnClickListener(v -> {
      startActivity(new Intent(MainActivity.this, VisualEnhanceActivity.class));
});

// 获取 button 的点击事件，开启联系我们 ContactActivity
findViewById(R.id.ACK).setOnClickListener(v -> {
      startActivity(new Intent(MainActivity.this, ContactActivity.class));
});

// TODO: more function of program
//findViewById(R.id.MORE).setOnClickListener(v -> {
//    startActivity(new Intent(MainActivity.this, MoreToComeActivity.class));
//});
```

以上代码在重写 onCreate 函数种实现，可以看到会根据用户的操作开启相关 Activity 的生命周期。

3.4.3 Sight 颜色识别模块代码解释

颜色识别模块代码架构如下图所示：

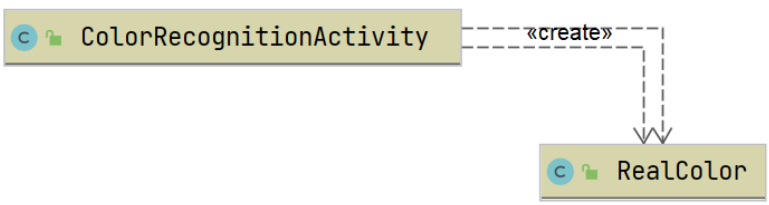

图 23：Sight 颜色识别模块代码架构图

从图中可以看到 RealColor 依赖于 ColorRecognitionActivity，实际上 Realcolor 是 CV 的模型类，用户拍摄的相片会在 ColorRecognitionActivity 类中转换成像供应的 Bitmap，然后输入到模型里，最后产生相应的结果。

相关代码如下：

```
1. static final int REQUEST_IMAGE_CAPTURE = 1;
2.
3.
4. private void dispatchTakePictureIntent() {
5.     Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE
   _CAPTURE);
6.     if (takePictureIntent.resolveActivity(getPackageManager()) !=
    null) {
7.         startActivityForResult(takePictureIntent, REQUEST_IMAGE_C
   APTURE);
8.     }
9. }
10.
11.
12.@Override
13.protected void onActivityResult(int requestCode, int resultCode,
   Intent data) {
14.     if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RES
   ULT_OK) {
15.         Bundle extras = data.getExtras();
16.         Bitmap imageBitmap = (Bitmap) extras.get("data");
17.         this.lastImage = imageBitmap;
18.         int pixel = getDominantColor();
19.         RealColor realColor = parseColor(pixel);
20.         addColorToList(realColor);
21.     }
22.}
23.
```

上面代码块的两个方法分别用于获取用户拍摄的相片，以及将相片输入到模型中，得到相应的结果反馈给用户。

dispatchTakePictureIntent() 负责获取用户拍摄的相片。

onActivityResult() 负责将相片输入到模型中，得到相应的结果。

以上代码是在 VisualEnhanceActivity 类中实现的。

3.4.4 Sight 视觉增强模块代码解释

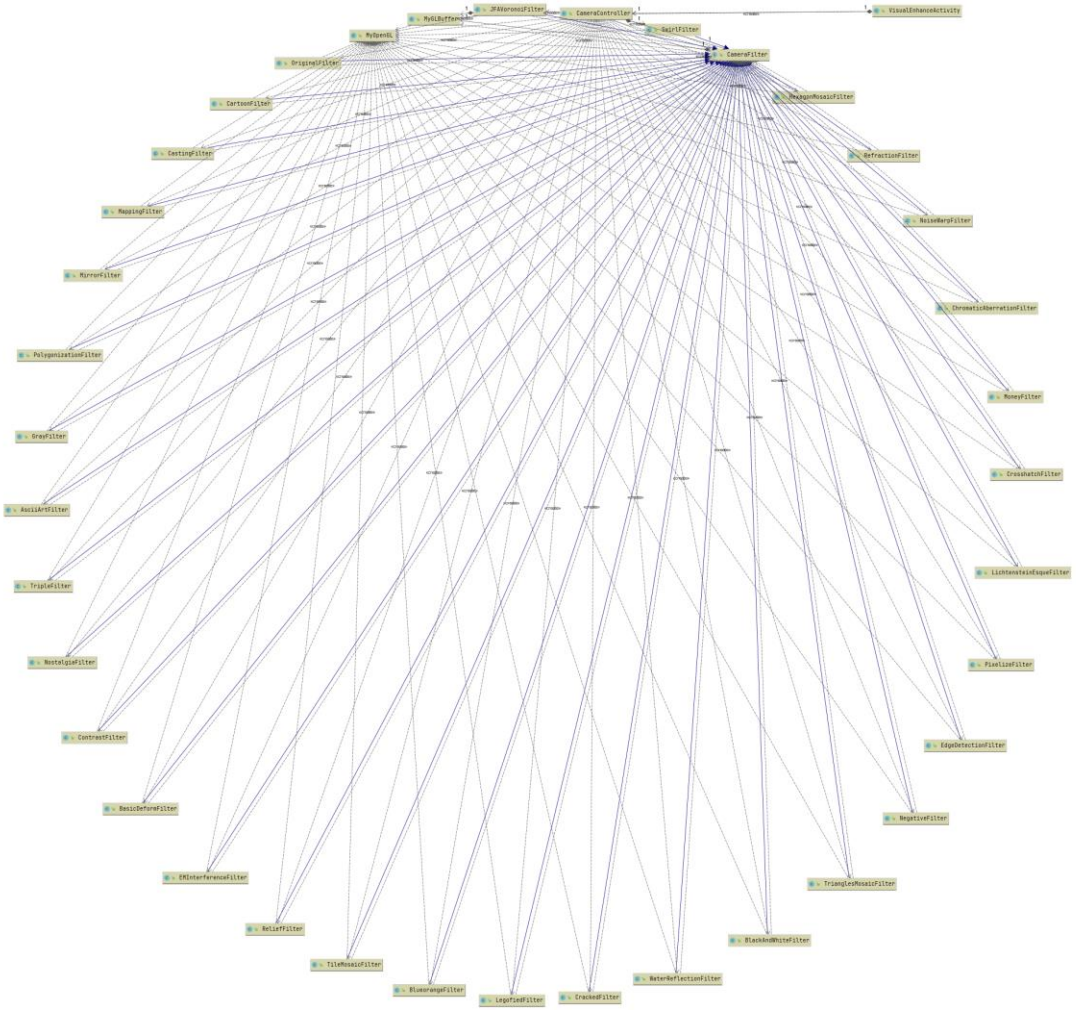

图 24：Sight 视觉增强模块代码架构图

可以看到图中关联最密集的类是 CameraFilter，CameraController，MyOpenGL 这三个，分别是所有 Filter 的基类，控制相机事件的类，以及用于和 OpenGL 库交互的接口类。

相关代码如下：

```
1. @Override
2. public boolean onFling(MotionEvent e1, MotionEvent e2, float velo
   cityX, float velocityY) {
3.     float velocity = Math.abs(velocityX) > Math.abs(velocityY) ?
   velocityX : velocityY;
4.     int step = velocity > 0 ? -1 : 1;
5.     mCurrentFilterId = circleLoop(TITLES.length, mCurrentFilterId
   , step);
6.     setTitle(TITLES[mCurrentFilterId]);
```

```
7.      if (renderer != null) {
8.          renderer.setSelectedFilter(FILTER_RES_IDS[mCurrentFilterI
    d]);
9.      }
10.
11.     return true;
12.}
13.
14.
15.private int circleLoop(int size, int currentPos, int step) {
16.
17.     if (step == 0) {
18.         return currentPos;
19.     }
20.
21.     if (step > 0) {
22.         if (currentPos + step >= size) {
23.             return (currentPos + step) % size;
24.         } else {
25.             return currentPos + step;
26.         }
27.     } else {
28.         if (currentPos + step < 0) {
29.             return currentPos + step + size;
30.         } else {
31.             return currentPos + step;
32.         }
33.     }
34.
35.}
36.
```

上面的 onFling 方法和 circleLoop 方法用于实现滑动切换 Frame。
上面的代码是在 VisualEnhanceActivity 中实现的。

```
1. public class AsciiArtFilter extends CameraFilter {
2.     private int program;
3.
4.     public AsciiArtFilter(Context context) {
5.         super(context);
6.
7.         program = MyOpenGL.buildProgram(context, R.raw.vertext, R
    .raw.ascii_art);
```

```
8.      }
9.
10.
11.     @Override
12.     public void onDraw(int cameraTexId, int canvasWidth, int canv
    asHeight) {
13.
14.         setupShaderInputs(program,
15.                 new int[]{canvasWidth, canvasHeight},
16.                 new int[]{cameraTexId},
17.                 new int[][]{});
18.         GLES20.glDrawArrays(GLES20.GL_TRIANGLE_STRIP, 0, 4);
19.
20.     }
21.
22.}
23.
```

上面的是一个视觉增强模块的使用示例，继承 CameraFilter 类，后面只需显示的调用 OpenGL 库提供的 buildProgram 方法即可完成模块切换。

3.4.5 Sight 工程结构图

Sight 工程结构图如下所示：

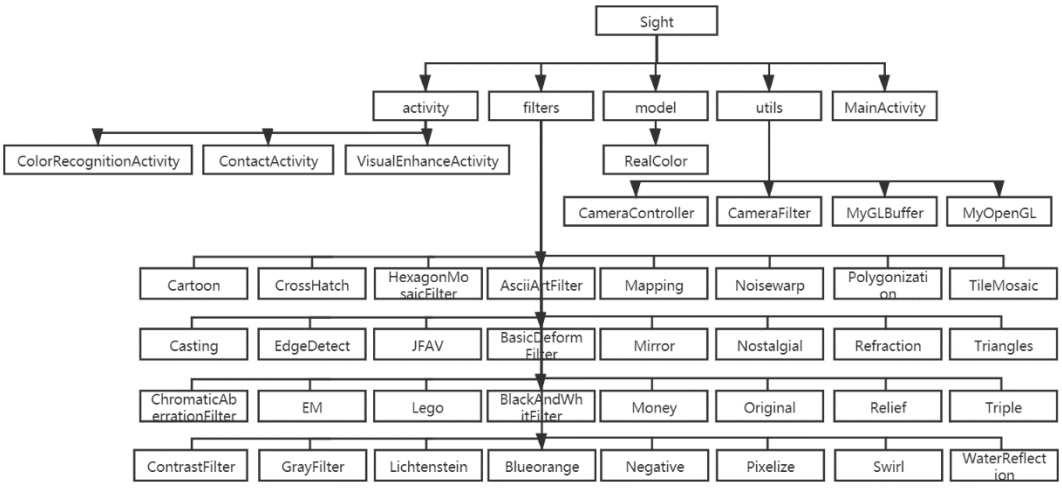

图 25：Sight 工程结构图

4. 系统可能的扩展

Sight 可能的扩展其实种类繁多，下面只列举目前我本人想到的几点 idea。

4.1 Sight 扩展一：超分辨率

CV 领域的技术可谓多种多样，SR 即为其中之一，我们的应用上，如果可以加入超分辨率机制，其实是有多种用处的。

一是可以对现有功能进行改善，如进一步提升颜色识别的准确率等

二是可以针对超分辨率，加入望远镜功能等，可以让手机清晰的观察到超远距离的物体，对物体的成像进行优化达到可以使得用户满意的效果等。

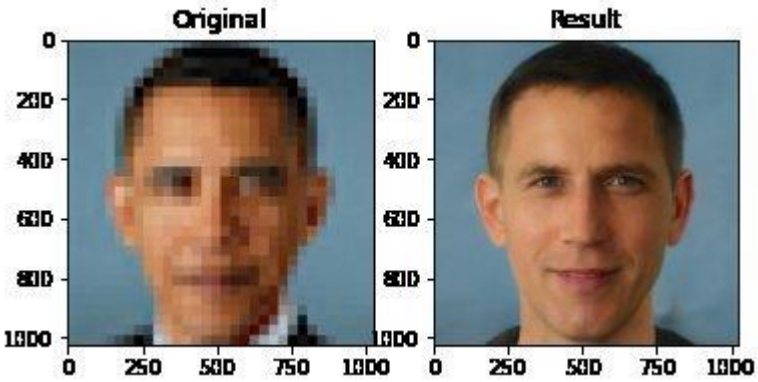

图 26：超分辨率应用示例

4.2 Sight 扩展二：三维建模

3D 的 CV 领域现在是研究的一大热门，针对 3D 的建模，我们的应用可以有很好的应用，如通过拍摄的视频或者相片进行测距，或者进行三维的场景复现。

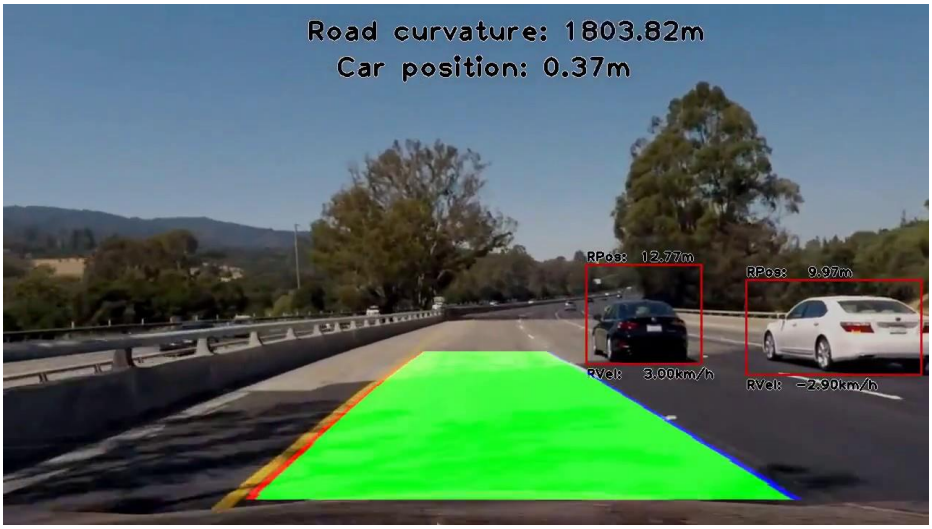

图 27：计算机视觉的 3D 建模在驾驶中的应用

4.3 Sight 扩展三：专类化

我们可以将 Sight 和每个独立的用户进行绑定，因为此 app 的服务对象是视力不正常人群，虽然 Sight 功能多样，但是也不能完美的符合所有用户的要去，因此，应该将 Sight 与每个用户进行绑定。

具体的一些可能的扩展如：针对客户使用相关的软件的数据，或者用户本身的提示，可以对现有的模型进行持续的优化，使模型不断的靠近用户的需求方向，必要时也可以进行重置，返回到用户想返回到的模型状态，这样 Sight 就有了更专类化的应用。也更加契合不同用户的需求。