Android CameraApp 介绍

文档修订记录

版本编号	日期	变更人	简要说明
V1.0	2015-11-24	路永瑚	初版整理
V1.1	2015-12-15	路永瑚	1.增加 Camera API 1/API 2 区别和介绍 2.修改部分图片模糊问题。 3.修改部分排版异常问题。

目录

前　言

本文主要基于 Android 系统介绍 CameraApp 的业务实现，以及一些 Camera 前沿技术探讨以及相关应用介绍，以便读者能对 CameraApp 有一个基本的了解。

1. Camera 基础知识介绍

1.1 手机 Camera 外观

目前手机 Camera 大致外观如图 1-1 所示：

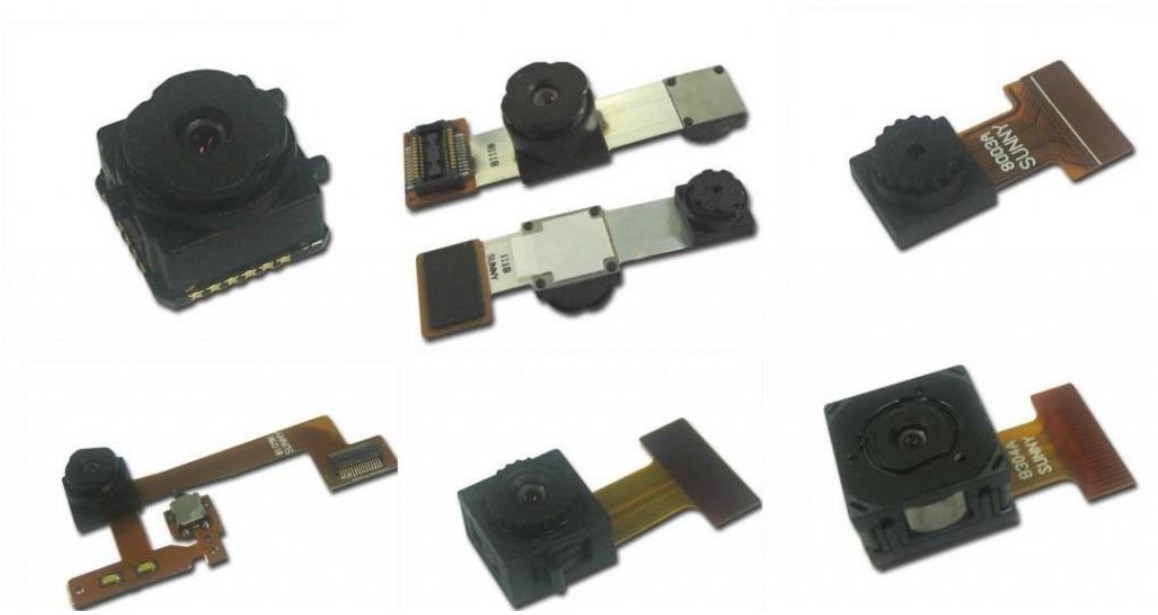


图 1-1

Camera 模组（CCM）介绍

CCM 包含四大件： 镜头(lens)、传感器（sensor）、软板（FPC）、图像处理芯片（DSP）， 如图 1-2：

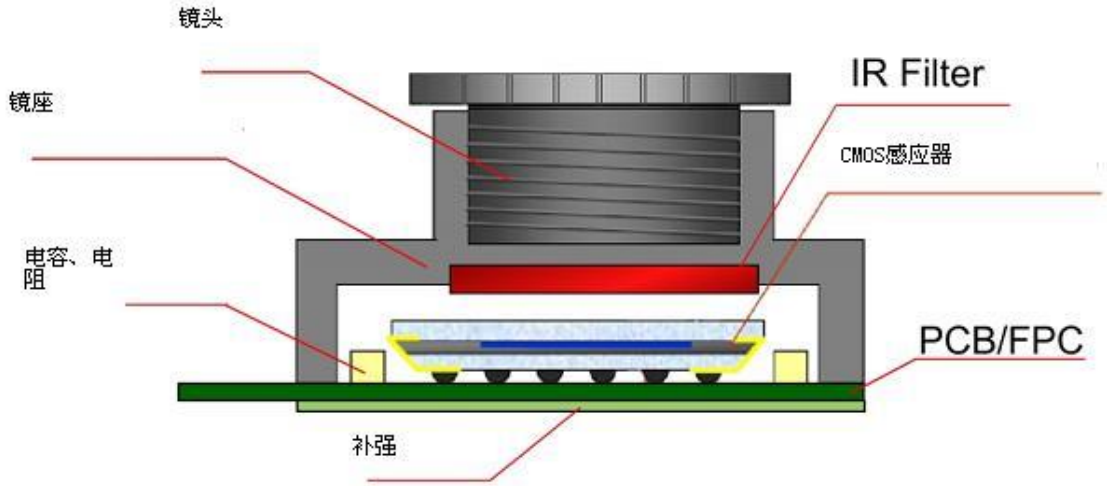


图 1-2

1.2 Camera 的成像原理

照相机(Camera)主要是由四到六片透镜组合的镜头来成像。当外部光线穿过 lens 后，经过 color filter 滤波后照射到 sensor 面上，sensor 将从 lens 上传到过来的光线转换成电信号，再通过内部的 AD（模数转换）转换为数字信号，如果 sensor 没有集成 DSP，则通过 DVP 的方式传输到 baseband，此时的数据格式是 RAW DATA。必须通过平台的 isp 来处理。如果集成了 DSP，这 RAW DATA 数据经过 AWB，color matrix，lens shading，gamma，sharpness，AE 和 de-noise 处理后输出 YUV 或者 RGB 格式的数据。最后会由 CPU 送到 framebuffer 中进行显示。

1）CCD/CMOS 将被摄体的光信号转变为电信号—电子图像（模拟信号）

2）由模/数转换器（ADC）芯片来将模拟信号转化为数字信号

3）数字信号形成后，由 DSP 或编码库对信号进行压缩并转化为特定的图像文件格式（RGB、YUV 等格式）储存。

1.3 Camera 三大基本功能

Camera 最基本的业务应该包含取景器（Preview）和拍摄照片（Capture），通过取景器用户可以实时掌握当前拍照、录像的内容。

- 拍照

原理：通过 sensor 获取一帧指定分辨率的数据，并保存为特定的图像格式，手机上常见的格式包括：jpeg，jpg，raw 等。

- 录像

原理：通过 sensor 不断的获取预览数据，然后将数据封装为特定的视频格式，手机上常见的视频格式包括：mp4,3gp 等。

- 预览

原理：不断的将 sensor 的预览数据显示到手机屏幕上，形成实时的图像预览。

2. Android Camera 应用开发基础

2.1 Camera 应用 UI 界面介绍

图 2-1 是 Android 原生 Camera UI 界面，主要包含四组控件：

1. 预览界面（整个屏幕）
2. 拍照按键（中间的蓝色按钮）
3. 菜单按钮（右侧的按钮）
4. 模式选择按钮（左侧的按钮）

图 2-1

当点击左侧模式选择按钮时如图 2-2， 点击右侧菜单按钮时如图 2-3

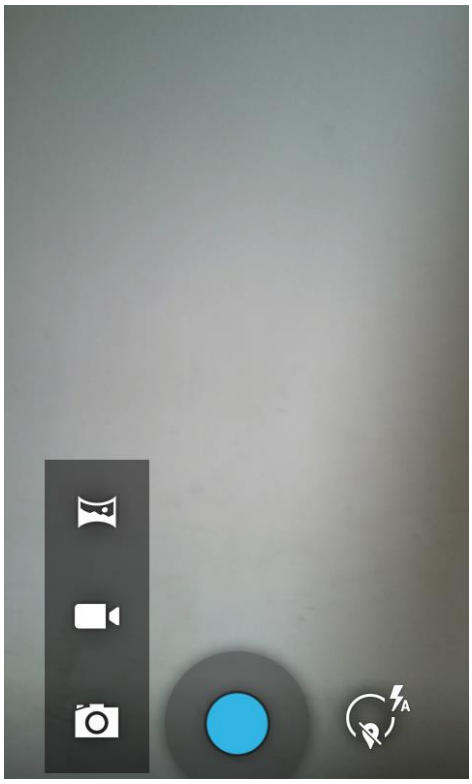


图 2-2

图 2-3

2.2 Camera 基本功能以及常用接口

Camera 的功能及设置是非常丰富的， 常见的功能及设置项如下：

- 录像
- 闪光灯
- 闪光灯
- 白平衡
- EV/EC
- ISO
- HDR
- Self-timer
- Scene/AutoSceneRecognition
- FaceDetection
- 降噪
- 连拍、ZSL 功能

- 分辨率(preview-size、photo-size、video-size) 4:3 16:9

当然丰富的功能也对应有准多的 framework 接口：

open
release
getNumberOfCameras
getCameraInfo
setPreviewDisplay
startPreview
stopPrevie
autoFocus
takePicture
setParameters
getParameters
setShutterCallback
setPreviewCallback
setOneShotPreviewCallback
setPreviewCallbackWithBuffer
addCallbackBuffer
startFaceDetection
……

interface
AutoFocusCallback
ShutterCallback
PictureCallback
PreviewCallback
OnZoomChangeListener
FaceDetectionListener
ErrorCallback
……

以上就是 Camera 中常用的一些接口，具体的使用方法和介绍大家可以去看 GoogleApi 介绍。

2.3 拍照功能的简单实现

因为 Camera 的功能实现涉及 App、framework、HAL，与 HAL 的交互主要通过 framework,所以 Camera 应用开发主要用到的类为 android.hardware.Camera,用该类拍照的执行步骤：

1. 通过 open(int)函数，获得一个 Camera 实例。

其中 int 参数一般可选值为 0 和 1,0 表示后置摄像头，1 表示前置摄像头，当然如果一个手机只有后置摄像头，当参数为 1 时，会发生打开错误，当手机的摄像头为 3 个时，可以传入 2 来打开第三个摄像头。但是需要注意的是，目前多数手机是不支持同时打开两个摄像头的，所以当我们打开摄像头 0 后，再去打开摄像头 1 时，先要释放摄像头 0，然后才能打开摄像头 1，否则会出现打开失败的错误。

此外，由于 open 函数可能需要消耗比较长的时间，所以最好把 open 函数放到工作线程里面，例如使用 AsyncTask 类。

2. 通过 getParameters()函数，获得已有的设置（默认）。

我们打开 camera 后，通过 getParameters 方法获取的 camera 参数是默认参数，如果我们想在退出 camera 前，保存当前的设置参数，然后再下次使用 camera 的时候，自动应用上次的 camera 参数，那么，在我们退出 camera 的时候，就需要手动去保存当前的 camera 参数。

3. 如果需要的话，调用 setParameters(Camera.Parameters)函数，修改第二步返回的 Camera.Parameters 对象。

在日常使用中，我们一般都会对 camera 做一些设置，比如说拍摄照片的分辨率，ISO 值等等，这些设置中，有些值有独立的接口供我们使用，但是有些值是没有独立的接口调用，此时就需要通过 setParameters 接口来将我们设置的值传给 camera。

4. 如果需要的话，调用 setDisplayOrientation(int)函数，设置屏幕水平或垂直。

在我们的应用开发中，首先我们需要确定我们开发的 camera 是否支持横竖屏切换，或者强制横屏或者强制竖屏。对于这个接口，谷歌官方给出了代码，我们再开发应用的时候，可以直接使用这个事例代码。

目前的情况是，大多数的 camera 都是设置的强制竖屏，这样可以保证比较好的用户体验，我们在设计 camera 应用的时候，最好也保持强制竖屏，但是也要保证在横屏的时候，界面图标也能横屏显示，并拍摄出正确角度的照片，所以，我们再 camera 应用内部需要调用加速度 sensor 来自动调整 UI 显示和照片拍摄角度。

5. 重要：传递一个已初始化好的 SurfaceHolder 对象给 setPreviewDisplay(SurfaceHolder)函数。如果没有 surface，camera 将无法打

开预览。

这一步是为了能够显示取景界面，我们需要在 app 里定义一个 SurfaceView 控件，然后获取 SurfaceHolder 并传递给 camera 用于预览界面的显示。

Android 4.0 以后的版本中，我们可以不再使用 setPreviewDisplay(SurfaceHolder)，而使用

setPreviewTexture(SurfaceTexture)，使用哪个方式来作为显示预览数据的载体取决于我们的功能，如果我们想让预览数据像普通的其它 view 控件一样，可以随意的对其进行平移，旋转等等操作的时候，我们就要选择后者来显示预览数据，大家可以去网上 baidu 一下具体的用法。

6. 重要：调用 startPreview()函数开始更新预览 surface。预览必须在你照相前启动。

7. 在你想要捕获一张照片时，调用 takePicture(ShutterCallback shutter, PictureCallback raw,PictureCallback jpeg)函数。等待回调函数中给出的实际图像数据。

其中 Camera.ShutterCallback 主要用于拍照发声和拍照动作回调，Camera.PictureCallback 用于接收拍照后返回的 jpg 数据。

8. 在照相后，预览显示将会停止。如果要照更多相片，需要再次调用 startPreview()函数（回至第六步）。

当前的手机 Camera 很多都支持 ZSL 模式，在这种模式下面就不需要重新调用 startPreview 来打开预览，而且拍照过程中，预览界面也不会卡一下。

9. 调用 stopPreview()函数，停止更新预览 surface。

10. 重要：调用 release()函数，释放 camera 引用，以便其他应用使用。应用应当在 onPause()调用时立刻释放 camera 引用(在 onResume()时重新打开它)。

有时我们需要在 App 里获取 camera 的预览数据（常用于实时特效显示，全景拍等应用场景），对于这个需求，android 提供了两个原生的 api 接口供开发者使用：

1. setPreviewCallback(Camera.PreviewCallback cb)

2. setOneShotPreviewCallback(Camera.PreviewCallback cb)

它们都可以获取实时的 camera 预览帧数据，但是第二个方法只会获取一帧数据，所以要看具体的需求选择不同的接口。

下面就是拍照命令和数据回调时序图，从这两张图中大家可以形象的理解拍照的流程。

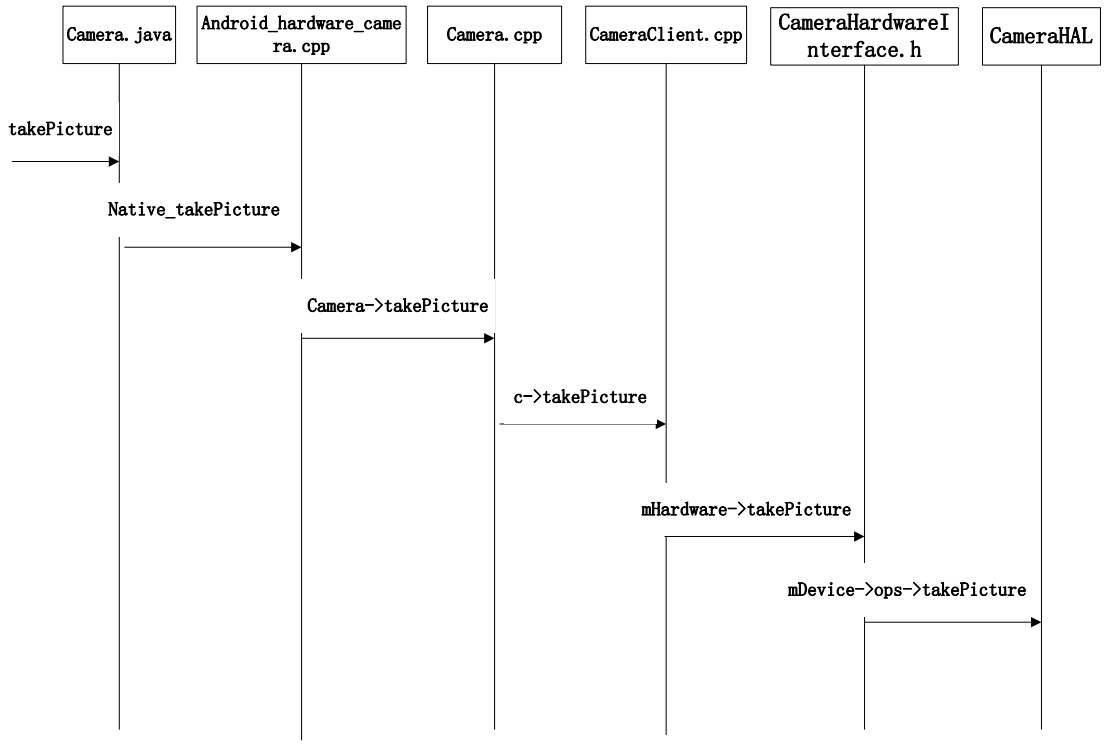


图 2-4 拍照命令时序图

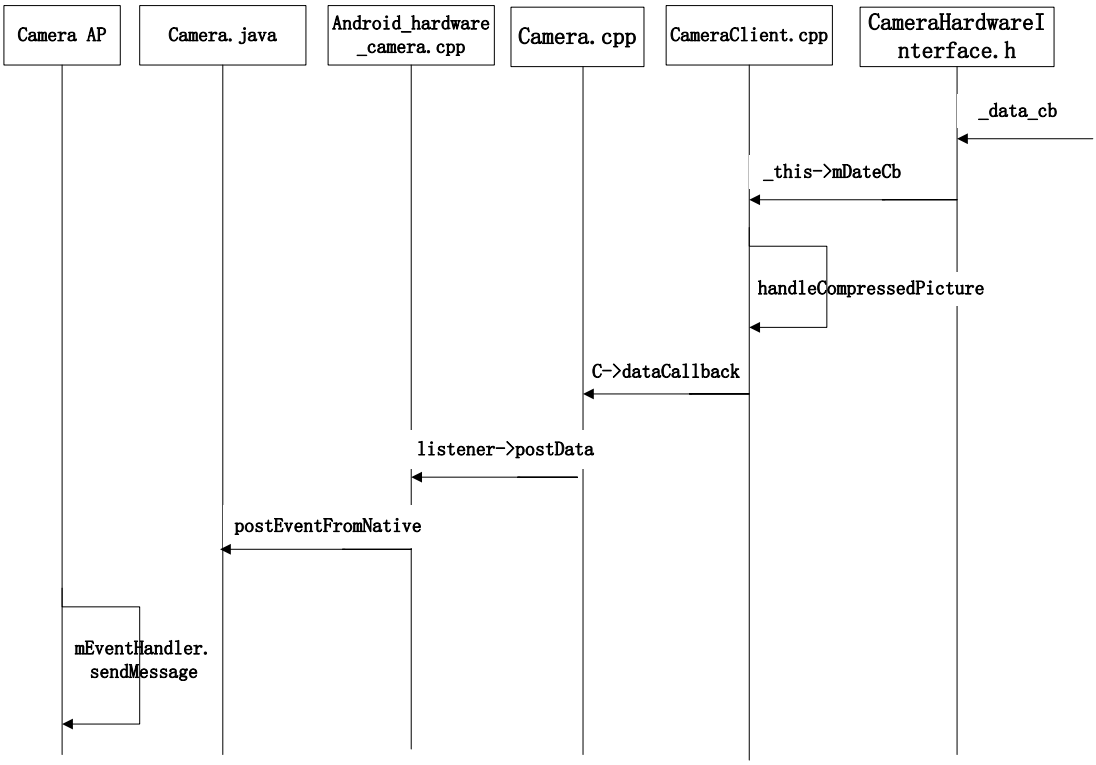


图 2-5 拍照数据回调时序图

2.4 录像功能的简单实现

录像的实现中主要使用以下步骤：

1. 如上所述，获取一个初始化好的 Camera 对象并开启预览。

2. 调用 unlock()函数，以允许 media 进程得以访问 camera。这一步是必须要做的，不然进行第五步操作的时候，会报错。

3. 传递当前 camera 对象至 setCamera(Camera)函数。请查看 MediaRecorder 关于视频录制的信息。

4.调用 MediaRecorder 的 setProfile()方法，设置录像参数。

5.调用 MediaRecorder 的 start()方法，开始录像。

6. 调用 MediaRecorder 的 stop()方法，停止录像。

7. 当结束录制时，调用 reconnect()函数重新取和加锁 camera 对象。

8. 如上所述，调用 stopPreview()和 release()函数，结束拍摄。

不同的 Android 设备有不同的硬件规格，如照片分辨率和自动对焦性能。为

了使你的应用和更多设备兼容，最好不要限制 camera 规格，比如你要显示照片尺寸选择菜单，最好是通过 getSupportedPictureSize()方法获取手机支持的照片尺寸来显示可选菜单，而不是在程序中固定设置几个 size。

2.5 Camera API1/API2 的区别

从 Android 5.0 开始（API Level 21），可以完全控制安卓设备相机的新 api Camera2(android.hardware.Camera2)被引入了进来。在以前的 Camera api 中，对相机的手动控制需要更改系统才能实现，而且 api 也不友好。不过老的 Camera API 在 5.0 上已经过时，在未来的 app 开发中推荐的是 Camera2 API。

L 版本之前的 Camera API

- 有限的图片数据流获取方式
- 有限相机状态信息
- 没有手动捕获控制

Camera2 API

- 支持 30fps 的全高清连拍
- 支持帧之间的手动设置
- 支持 RAW 格式的图片拍摄
- 支持快门 0 延迟以及电影速拍
- 支持相机其他方面的手动控制包括噪音消除的级别

因为新 API 换了架构，让开发者用起来更难了。 先来看看 camera2 包架构示意图 2-6：

camera2 流程示意图

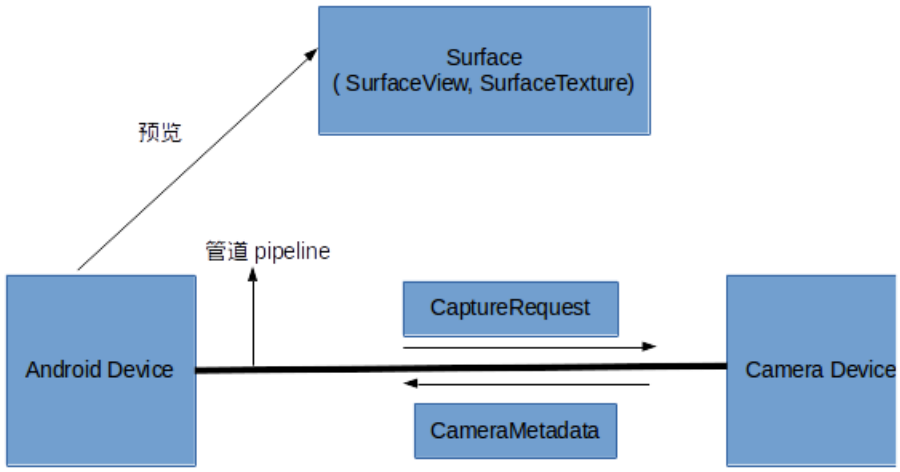


图 2-6

这里引用了管道的概念将安卓设备和摄像头之间联通起来，系统向摄像头发 Capture 请求，而摄像头会返回 CameraMetadata。

虽然推荐在未来的 app 中使用 Camera2 API，但是只有 Lollipop 的设备商才可用，也不大可能会出现兼容老设备的包。因此在最小版本（minSdkVersion）升到 21 之前，大家还是需要继续使用 Camera API(android.hardware.Camera)。

至于 Camera2 API 的具体介绍大家可以查看 google 介绍。

3. Android Camera 相关架构

前面也说到了，Camera 从 App 到 HAL 都有涉及到，所以 App 的实现其实就是对 framework 接口的应用，对下控制流的实现， 具体的框架如图 3-1:

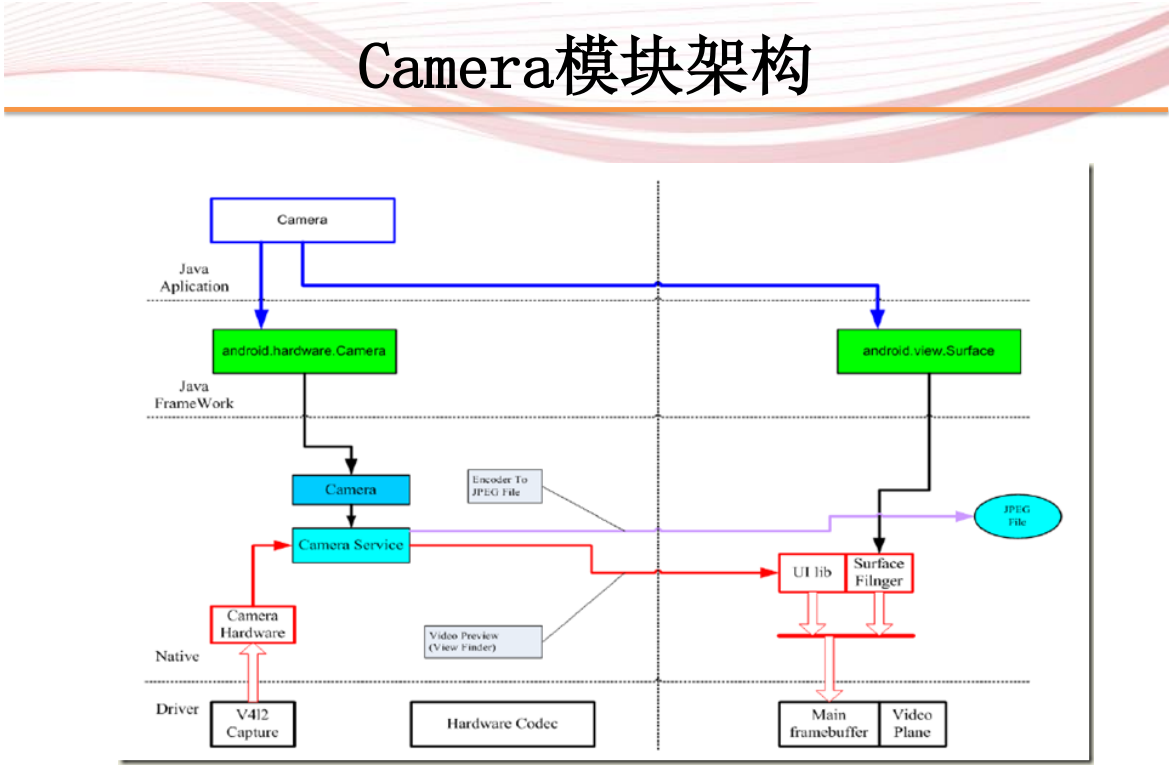


图 3-1

3.1 Media Recoder 模块架构

如图 3-2：

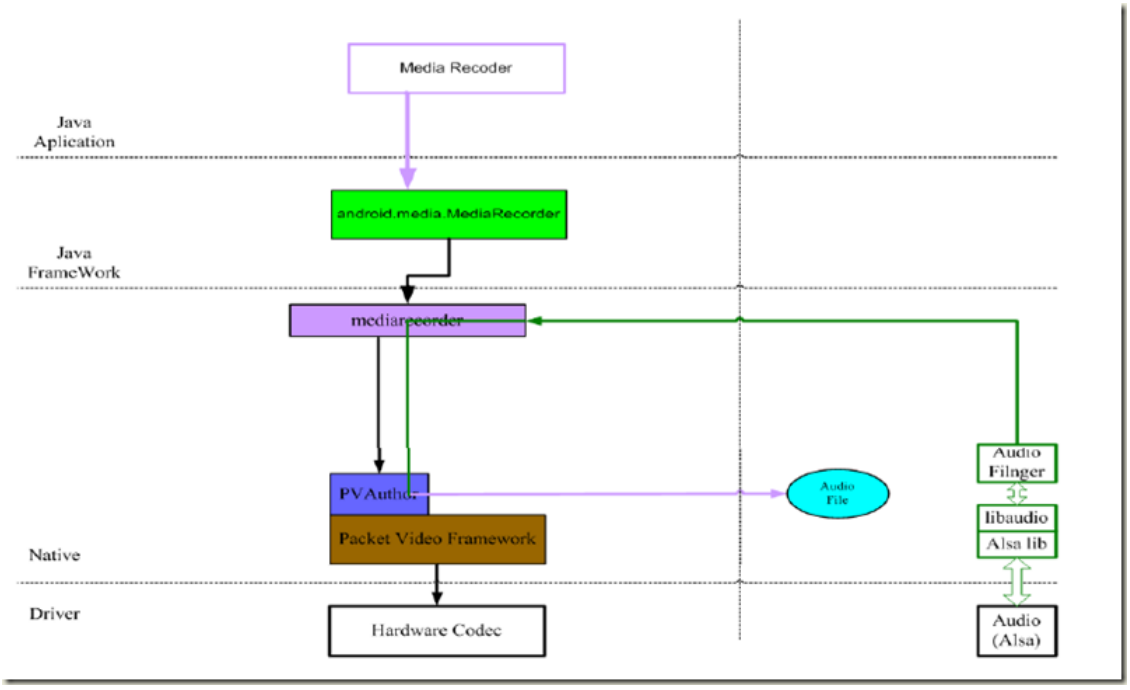


图 3-2

3.2 Camera 调用流程：

如图 3-3：

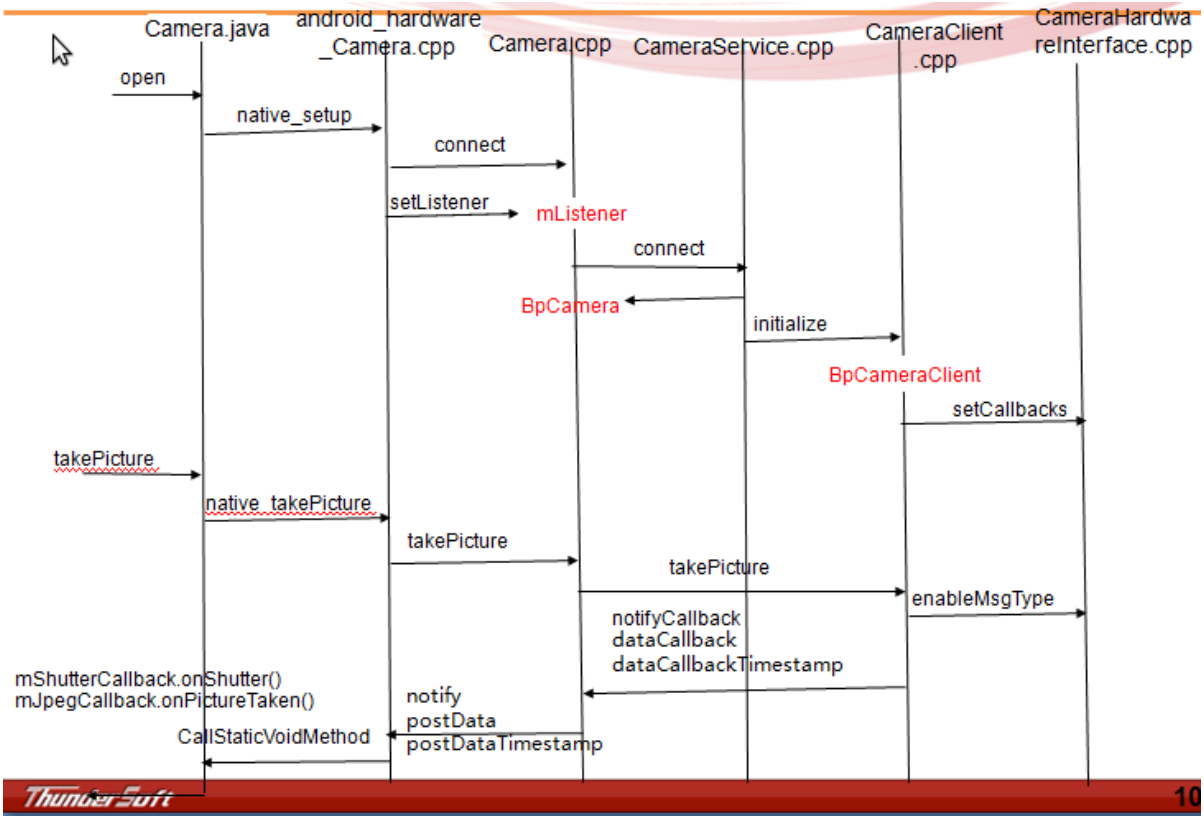


图 3-3

从图 3-3 以及 Camera 的接口可以总结出 Camera 的宏观处理逻辑：

- 上层通过调用接口来控制下层，并设置回调函数，实现控制流；
- 下层通过回调函数向上层传递各种类型的数据，实现数据流。

4. 一些 Camera 应用的介绍、分析

4.1 UCam 全能相机

- 其中最大的特色应该是各种特效（包括标准相机里面的特效和莫魔法美颜）
- 支撑特效依靠的是图像的算法，所以对于图片的效果处理，算法是核心技术。

- 对于特效的实时显示，使用的是预览帧数据返回接口，这种实时渲染的效果比较消耗系统资源，所以一般来说，实时特效的显示分辨率都比较低，否则容易出现卡顿或者内存使用过高的问题。

4.2 彩漫相机

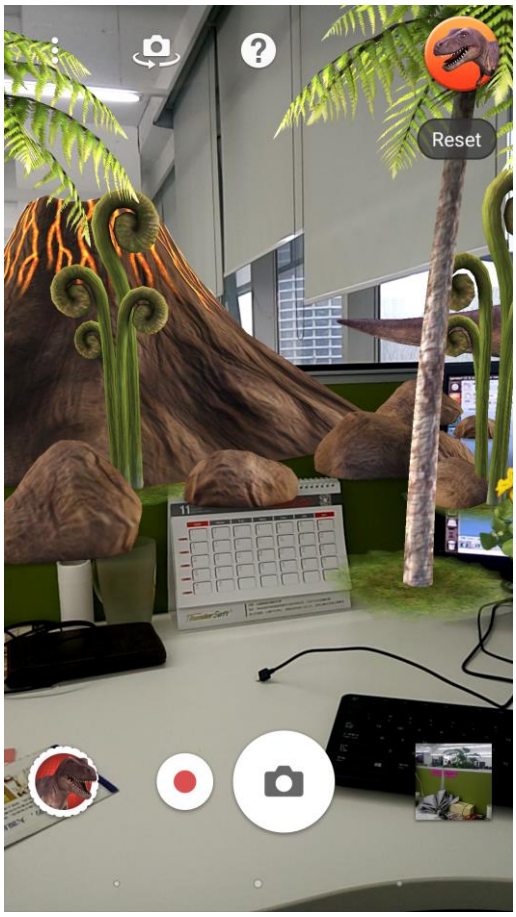
其主要使用了人脸识别技术，非常有创意， 采用了类似“移花接木”的效果，非常迎合年轻人，图片文字等很接地气。

应用基本原理：

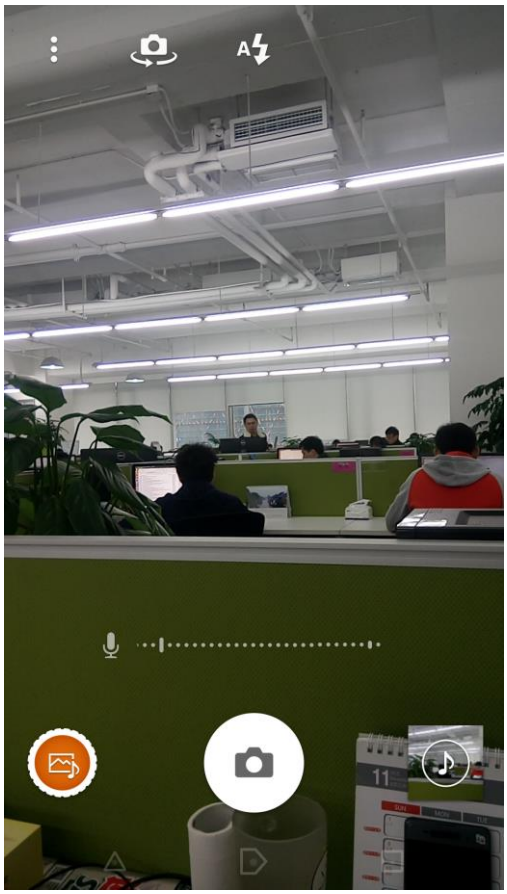
通过面部识别算法，截取出面部部分图形， 然后经过一定的处理，再和预设图片进行 合成

4.3 Sony Camera

Sony Camera 中的 Ar-effect app:



Sony SoundPhoto:



SoundPhoto 是 Sony 非常有特色的一款 Camera 应用，用户可以拍一副带有 10s 时长声音背景的图片， 图片会带有一个音乐的符号。在打开图片时可以同时播放录制的背景声音。

SoundPhoto 的实现中主要应用了 Camera + 数据库 + Audio/picture 混合编码的算法。

5．Camera 性能优化

Camera 的性能优化相当重要，一个性能较好、拍照图片质量不错的手机总有很多卖点，但是 Camera 的性能优化又比较困难，因为 Camera 应用涉及到上层和底层几乎所有部分开发，所以 Camera 的优化也会涉及到从上到下。

一般情况下主要涉及到这些方面：

- 在 open Camera 时尽量减少非必要参数的加载，同时可以在开机启动后预加载 Camera 底层参数。
- 使用 PDAF 等技术以减少聚焦时间提升 Camera 拍照速度。
- 使用 ZSL 模式，减少拍照时 stopPreview 和 startPreview 的时间。
- 将拍照过程中的图片保存以及动画加载放在线程中执行，并且尽量不要通让这些线程的状态 block 拍照。
- 在 onResume 中不要执行耗时太多的动作。
- 在打开多个 Camera 时可以使用线程同步执行，不必按顺序一个个来打开。
- 提升 startPreview 速度和 Focus 速度。
- 在 takepicture encoder 时尽量优化降噪耗时和性能。