ArcSoft 虹软

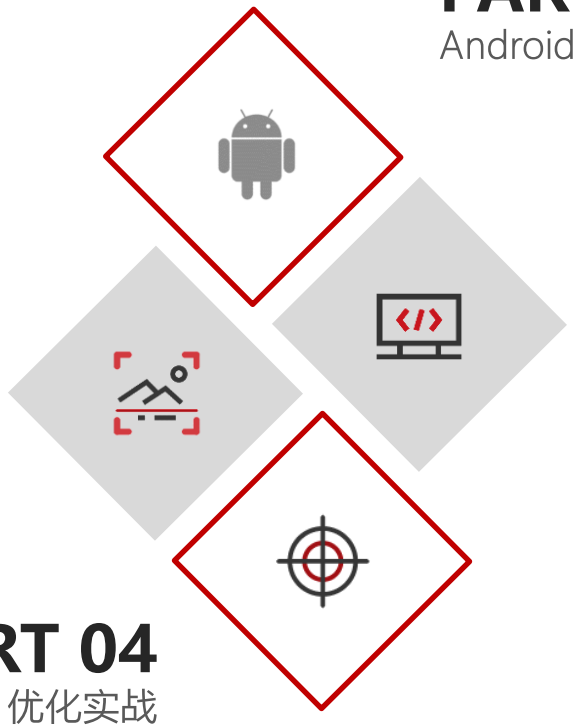
ArcSoft IT大咖说 知识共享平台

Android 平台
ArcFace 人脸识别的开发与优化提升

ArcSoft 虹软

ArcSoft | IT大咖说 知识共享平台

PART 01
Android工程配置及注意事项



PART 02
SDK介绍

PART 03
图像格式说明

PART 04
优化实战

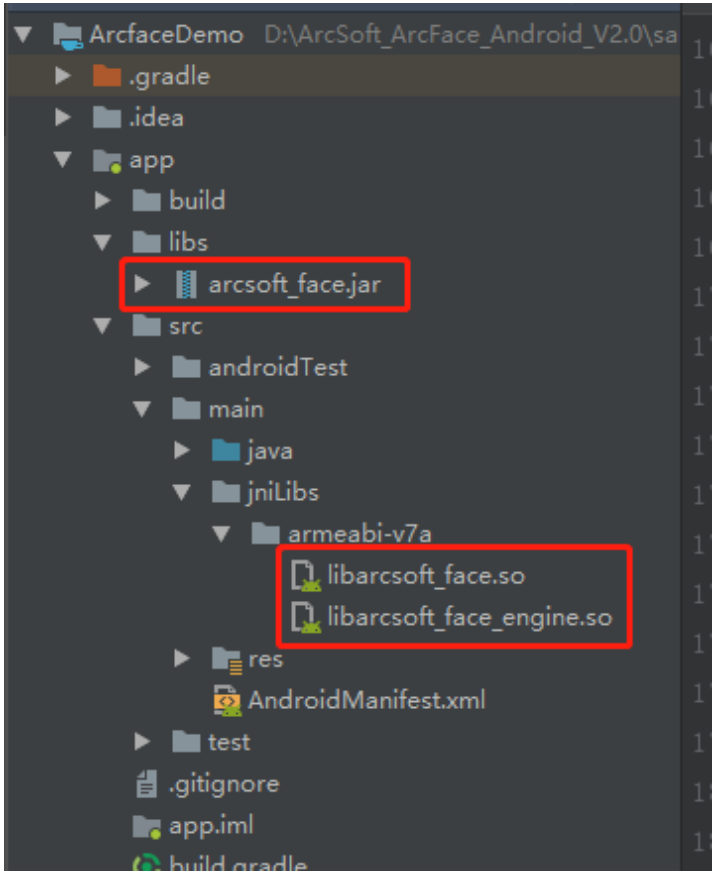
PART 01
Android工程配置及注意事项

Android工程配置

配置内容：

1、添加jar至{project}->{module}->libs目录，

2、添加so至{project}->{module}-> src->main->jniLibs目录



Android工程注意事项

注意事项：

动态库文件目录下相关动态库不全、或有多个ABI子目录，但是子目录中的文件列表不同很可能会导致java.lang.UnsatisfiedLinkError从而导致crash。同时，在打release包时，确保SDK相关的jar不被混淆。

```
jniLibs
├─arm64-v8a
│      a.so
│      b.so
└─armeabi-v7a
       a.so
       b.so
```

```
jniLibs
├─arm64-v8a
│      a.so
└─armeabi-v7a
       a.so
       b.so
```

```
jniLibs
└─armeabi-v7a
       a.so
       b.so
```

```
jniLibs
└─armeabi-v7a
       a.so
```

PART 02
SDK介绍

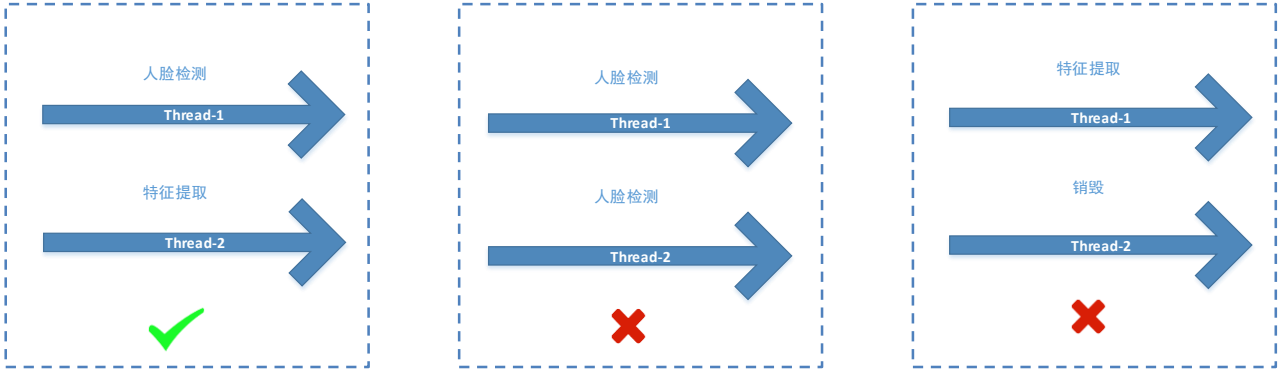
- 接口功能介绍
- 引擎的多线程正确使用方式

接口功能介绍

引擎的多线程正确使用方式

单个引擎的同一功能模块中的算法功能函数不支持多线程调用，且调用过程中不能进行销毁。
若需多线程调用，需要创建多个引擎。

例如：



PART 03
图像格式说明

- 理解图像格式
- 相关图像颜色空间介绍
- 各类图像格式介绍
- 图像格式转换

理解图像格式

Android平台的ArcFace 2.0 SDK目前支持2种图像格式：NV21,BGR24。而一般图片文件都是以压缩形式存储的，例如jpg。

一般各种平台都会有提供相关的RGB提取函数。例如Android平台，可以通过jpg文件 -> Bitmap对象 -> 提取ARGB的方式获取ARGB数据。有了ARGB数据，再了解下图像格式，我们即可自己编写各种图像转换方法。例如通过调整RGB的位置以获取BGR24数据；通过R、G、B值计算每个像素点的Y、U、V值再存放到NV21数组的对应位置可生成NV21数据。

注意：

颜色空间内的位置更改实现的格式转换不会导致失真（如RGB24修改为BGR24，只是更换B和R的位置）；

但是颜色空间间的转换会失真。因此在做图像格式转换时应以颜色空间内转换优先。

相关图像颜色空间介绍

RGB颜色空间

RGB颜色空间以Red、Green、Blue三种基本色为基础，进行不同程度的叠加，产生丰富而广泛的颜色，所以俗称三基色模式。

常见的RGB格式有：RGB_565、RGB_888、ARGB_8888、ARGB_4444等。

YUV颜色空间

在YUV颜色空间中，Y用来表示亮度，U和V用来表示色度。

常见的YUV格式有以下几大类：

planar: Y、U、V全部连续存储，如I420、YV12

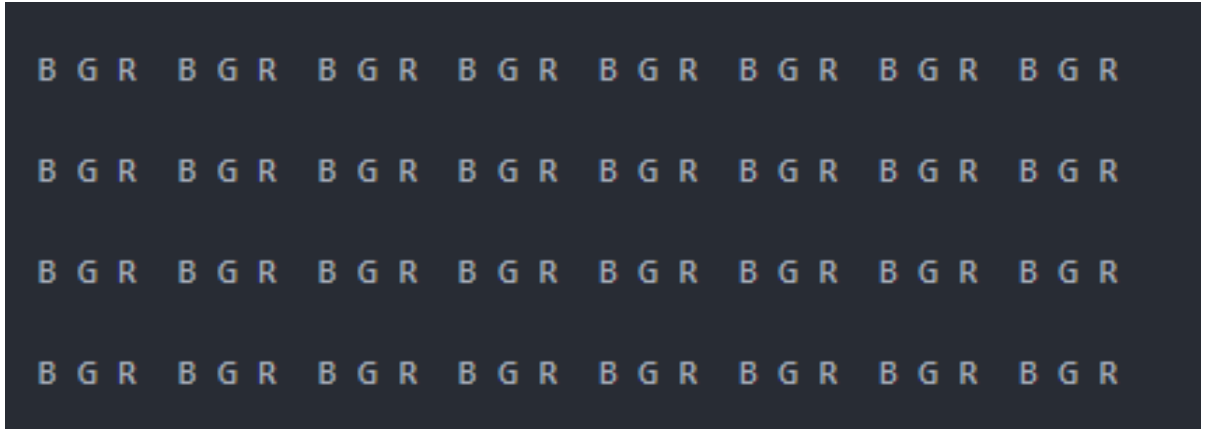
packed: Y、U、V交叉存储，如YUYV

semi-planar: Y连续存储，U、V交叉存储，如NV21、NV12

BGR24图像格式

BGR24图像格式是一种采用24bpp(bit per pixel)的格式。每个颜色通道B、G、R各占8bpp。

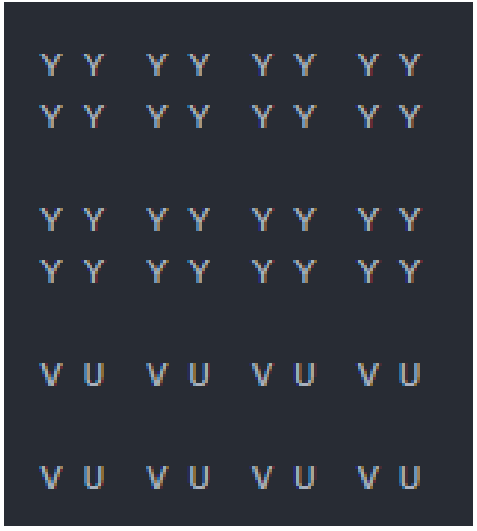
排列方式如：



NV21图像格式

NV21图像格式属于 YUV颜色空间中的YUV420SP类，每四个Y分量共用一组U分量和V分量，Y连续排序，U与V交叉排序。

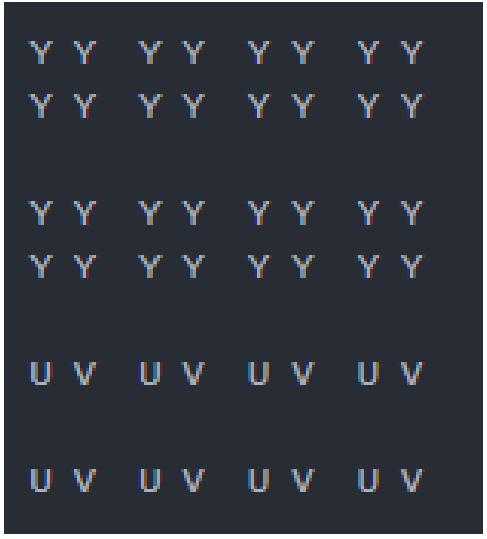
排列方式如：



NV12图像格式

NV12图像格式属于 YUV颜色空间中的YUV420SP类，每四个Y分量共用一组U分量和V分量，Y连续排序，U与V交叉排序（NV21和NV12只是U与V的位置相反）。

排列方式如：



图像格式转换介绍

1.ARGB32->BGR24

这是一个32位转24位的过程，需要舍弃透明度数据。
举个例子，对于4x2的图片，ARGB32格式内容为：

A1 R1 G1 B1 A2 R2 G2 B2 A3 R3 G3 B3 A4 R4 G4 B4
A5 R5 G5 B5 A6 R6 G6 B6 A7 R7 G7 B7 A8 R8 G8 B8

那么若需要转化为BGR24，内容将变成：

B1 G1 R1 B2 G2 R2 B3 G3 R3 B4 G4 R4
B5 G5 R5 B6 G6 R6 B7 G7 R7 B8 G8 R8

转换过程：
舍弃A，再将R和B的位置互换。

图像格式转换介绍

2.NV12->NV21

NV12的数据排列：

Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y

Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y

U V U V U V U V

U V U V U V U V

NV21的数据排列：

Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y

Y Y Y Y Y Y Y Y
Y Y Y Y Y Y Y Y

V U V U V U V U

V U V U V U V U

转换过程：

仅需替换U和V的位置即可

PART 04
优化实战

- trackID妙用
- 优化点

trackID妙用 - trackID的作用

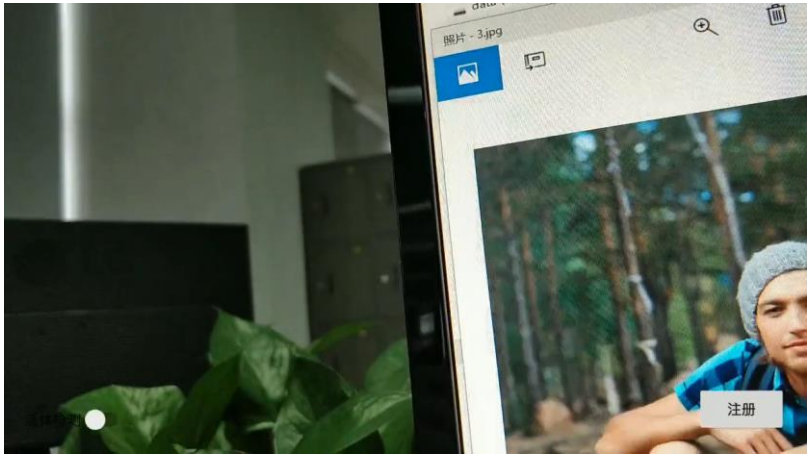
trackID ? 人的编号

5个人 在画面中停留2s

无trackID 有trackID

FR 次数？(fps = 25)

5*2*25 5



trackID - 上下帧相同人脸的判断

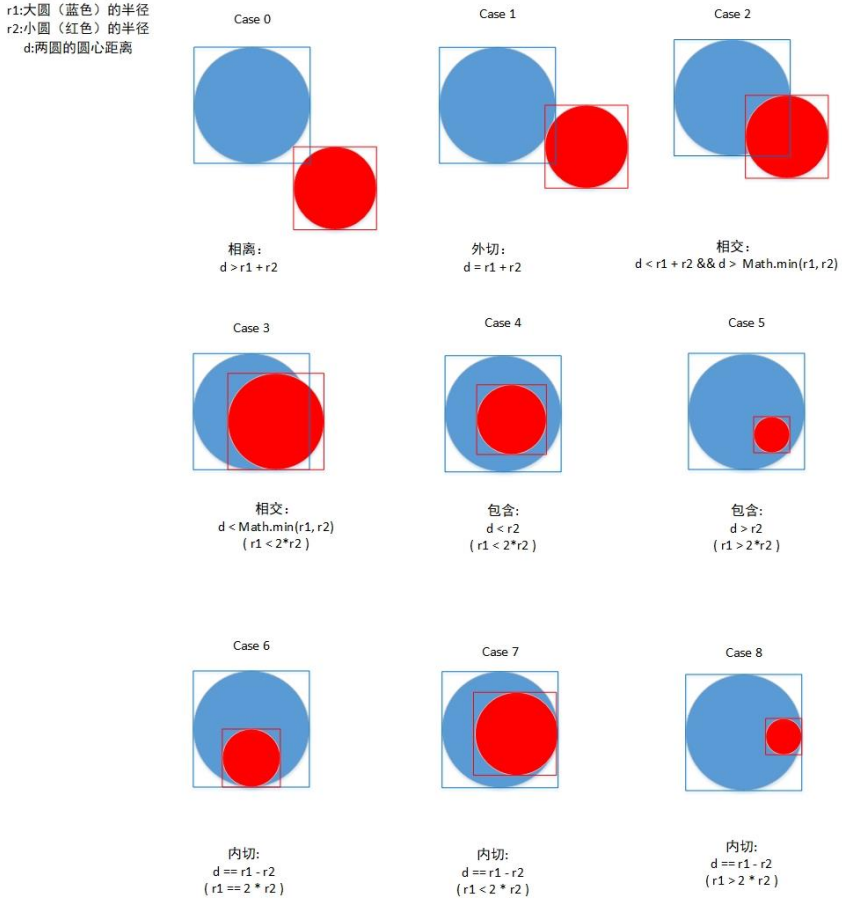
一般情况下，我们可以通过上下帧的重合度来判断两个人脸框是否属于同一个人，那么重合度的依据是什么？这里提供一种思路：通过距离判断。若当前帧和上一帧的所有人脸框中最接近的那个人脸框的中心距离小于某个值validDistance（该值越小，对人脸框的接近程度要求越高），则我们认为这两个框属于同一个人脸。由于人脸框基本都是接近正方形的矩形，我们可以将其当做正方形处理，使用其内切圆的关系便于说明。

trackID - 上下帧相同人脸的判断

右图列出了可能存在的一些情况。

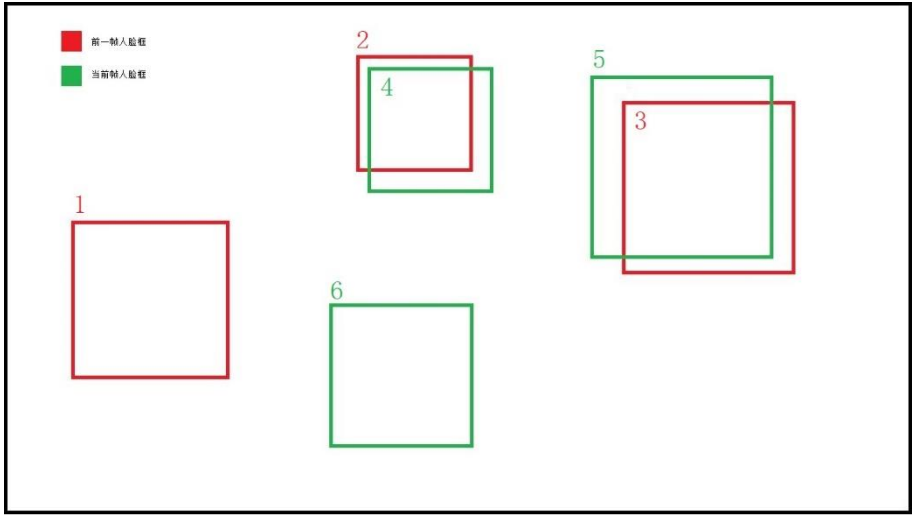
若我们将validDistance的值设为较大人脸框内切圆的半径的一半，则根据该图，我们可以判定：

- case 0, case 1, case 2, case 5, case 8不是同一个人
- case 3, case 4, case 7 是同一个人
- case 6 是临界情况，自行决定



trackID - 上下帧相同人脸的判断

- 例如以下情况



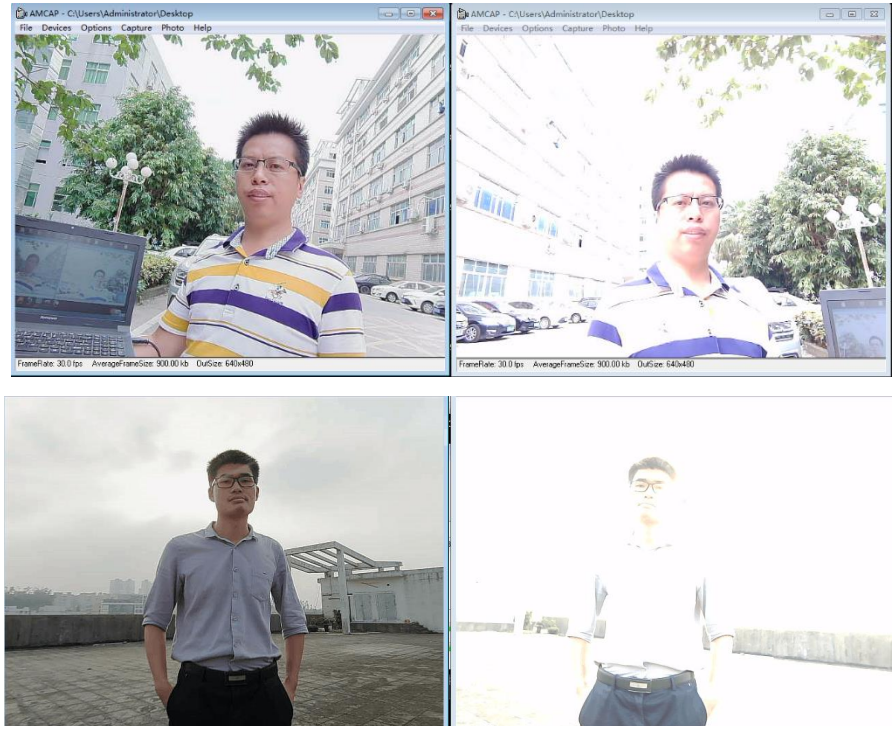
红色框代表前一帧人脸框，绿色框代表当前帧人脸框。那么我们可以判断得：

- 前一帧的1号人脸框在当前帧消失
- 前一帧的2号人脸框和当前帧的4号人脸框对应
- 前一帧的3号人脸框和当前帧的5号人脸框对应
- 6号人脸框是当前帧出现的新的人脸框

优化实战 - 人脸检测

人脸检测是人脸识别的基础。可以通过**控制环境光线**、**提升相机硬件**、**使用WDR技术**等方法来提升画面质量以提升人脸检测的速度，以及**设置人脸检测角度为单方向**的方式以提升人脸检测速度。

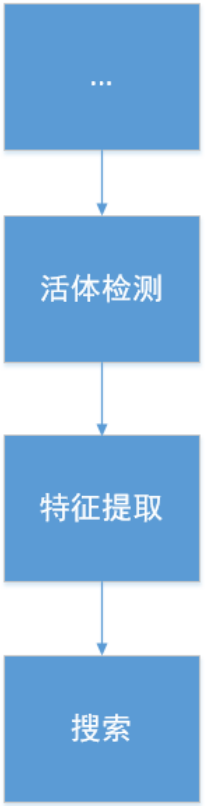
右图是使用WDR技术前后的图像效果比对。



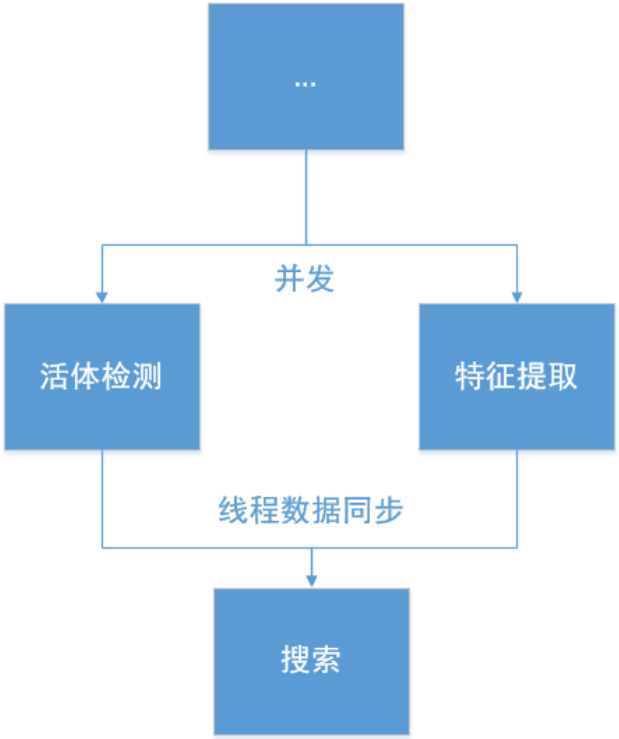
优化实战 - 活体检测

活体检测是一种安全性验证，其结果并不影响人脸特征提取，因此可和人脸特征提取并行。该项相对于人脸检测是异步操作。对于n:m识别的场景，需要多人脸活体识别，因此可以使用图片模式的异步活体检测。

活体检测和特征提取串行



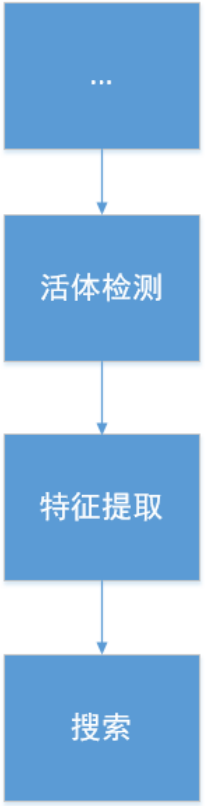
活体检测和特征提取并行



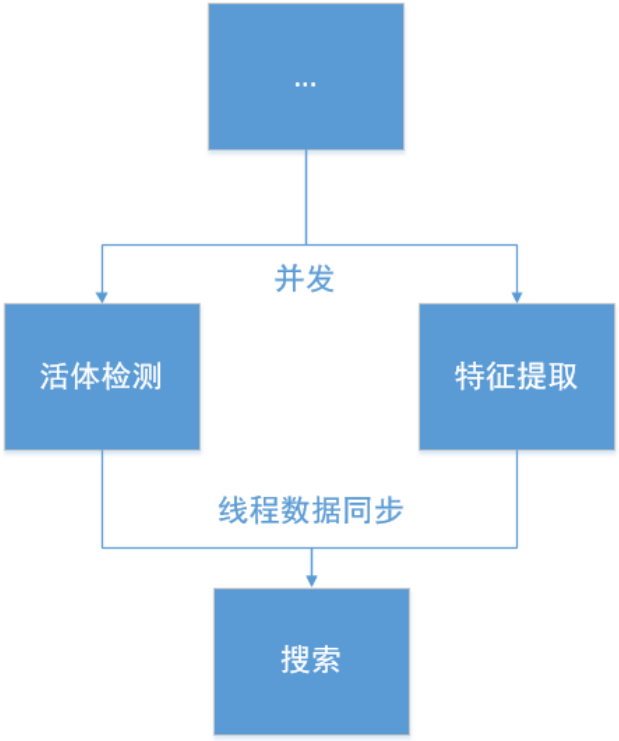
优化实战 - 特征提取

在特征提取后才可进行人脸搜索。该项可与活体检测并行，若特征提取成功后活体检测结果为未知，则等待活体检测结果，否则可根据活体检测结果做对应操作。该项相对于人脸检测是异步操作。

活体检测和特征提取串行

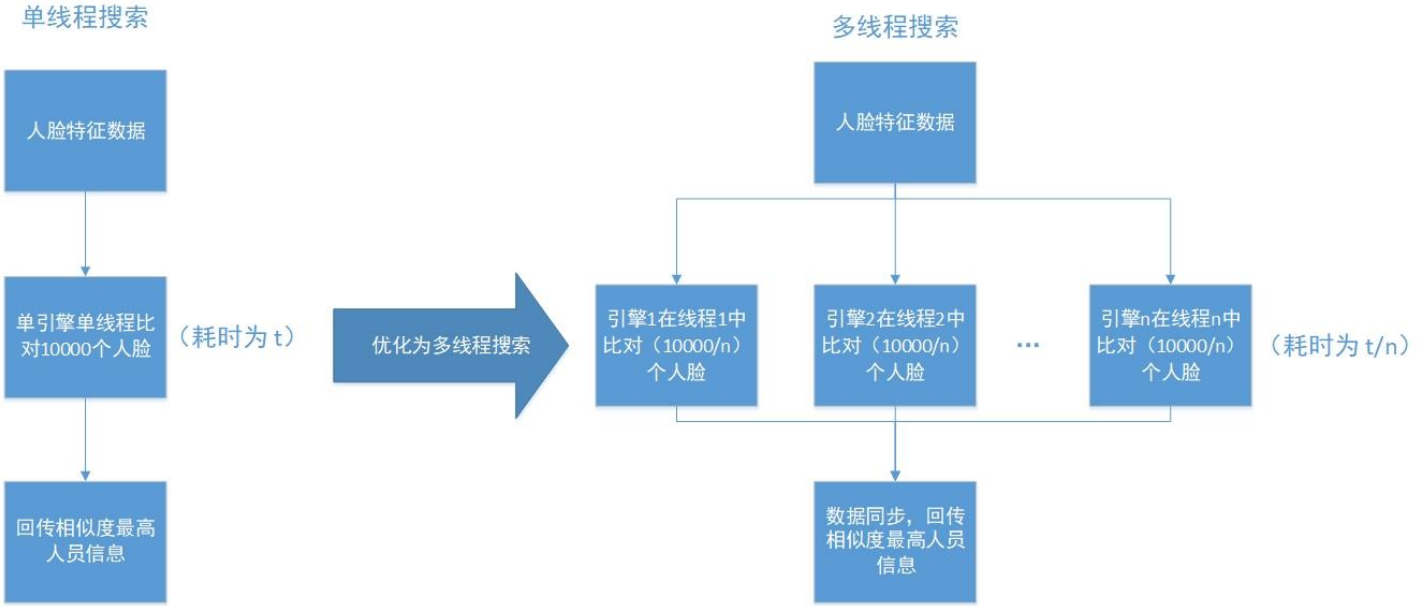


活体检测和特征提取并行



优化实战 - 人脸搜索

人脸搜索建议在服务器端完成，在1000人以下耗时极短，若人脸库过大，可通过**多引擎多线程**比对。



优化实战 - 活体检测的选择

在1:n的场景下，可以使用视频流模式的活体检测。

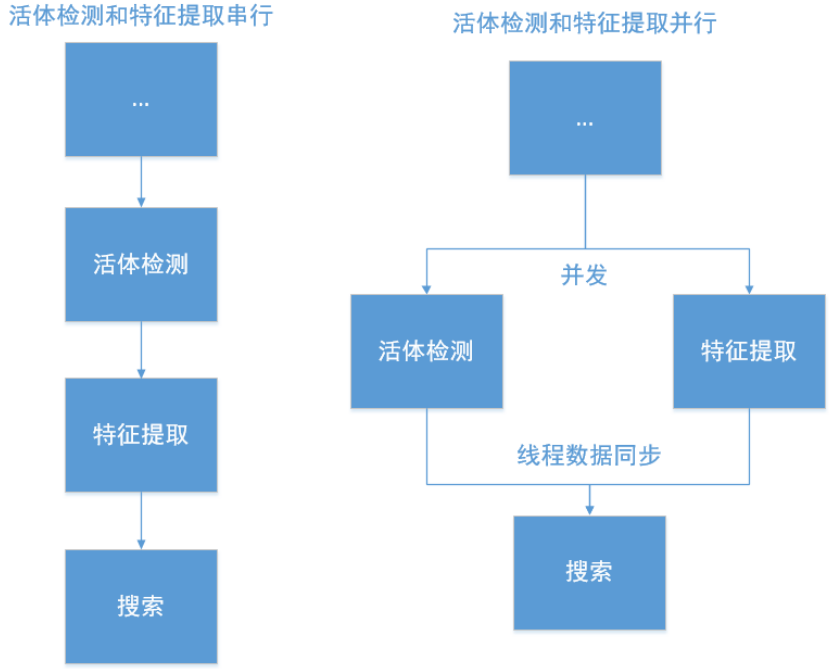
在n:m的场景下，可以通过异步图片模式活体检测。效果可见下图。

视频流模式
单人脸活体判断，另一个人活体值是未知

静态图模式
结合trackID实现多人脸活体判断

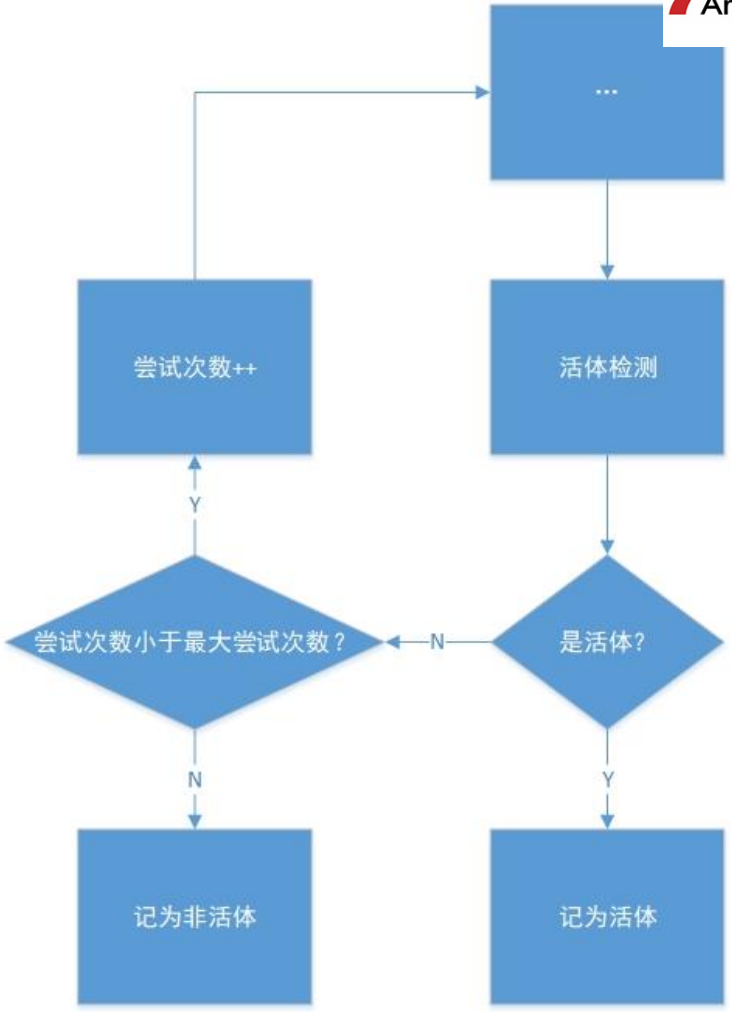
优化实战 - 多项并发

活体检测和特征提取十分耗时，两者可同步进行，在使用特征进行搜索前需要验证人脸的活体检测结果，若活体检测未出结果，则等待。



优化实战 - 重试机制

大角度、模糊人脸等因素对活体检测和特征提取的影响较大，可通过添加重试机制来确保活体和特征提取顺利进行。



优化实战 – 引擎池、线程池

例如在做异步特征提取时，可以通过特征解析引擎池+特征解析线程池实现并发异步特征解析，在画面中有多个人脸的情况下，使用该方案可以大幅提升画面中有多个人脸的情况下的识别速度，但是对设备性能要求较高。（异步图片模式活体检测方案同理）

注意:

若CPU核心数较少，不建议开过多线程，因为核心数不足的情况下，并发过多的线程还是等待CPU调度，不仅速度没提升，还浪费了内存资源

扫描上方二维码
获取更多虹软人脸识别技术

THANKS

谢 谢