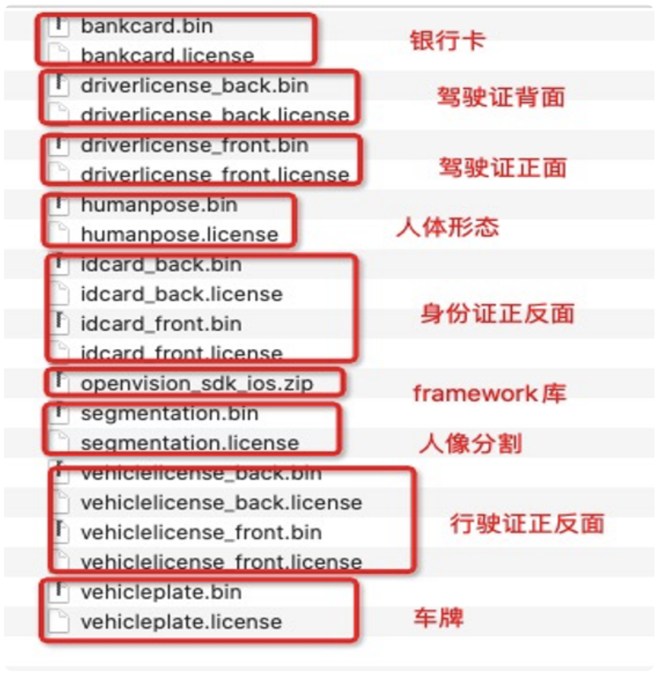
IOS端openVisionSdk集成说明

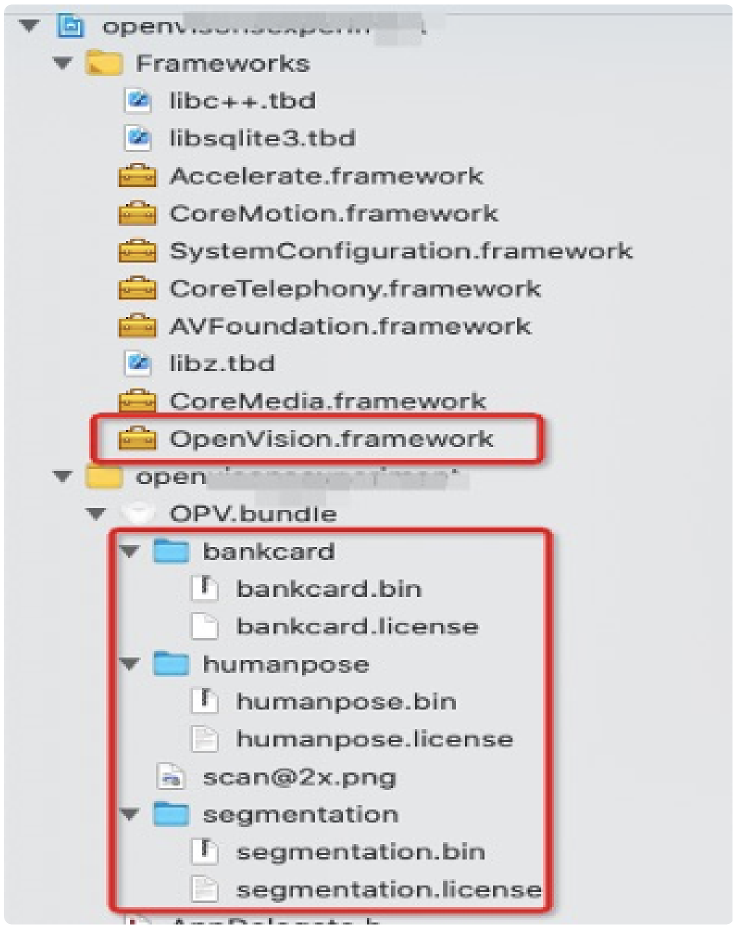
一、Xcode配置工程

1、获取相关资源压缩包（由阿里云相关人员提供下载链接）后，解压压缩包，可看到如下资源文件framework包及支持相关能力的bin、license文件。如下图：



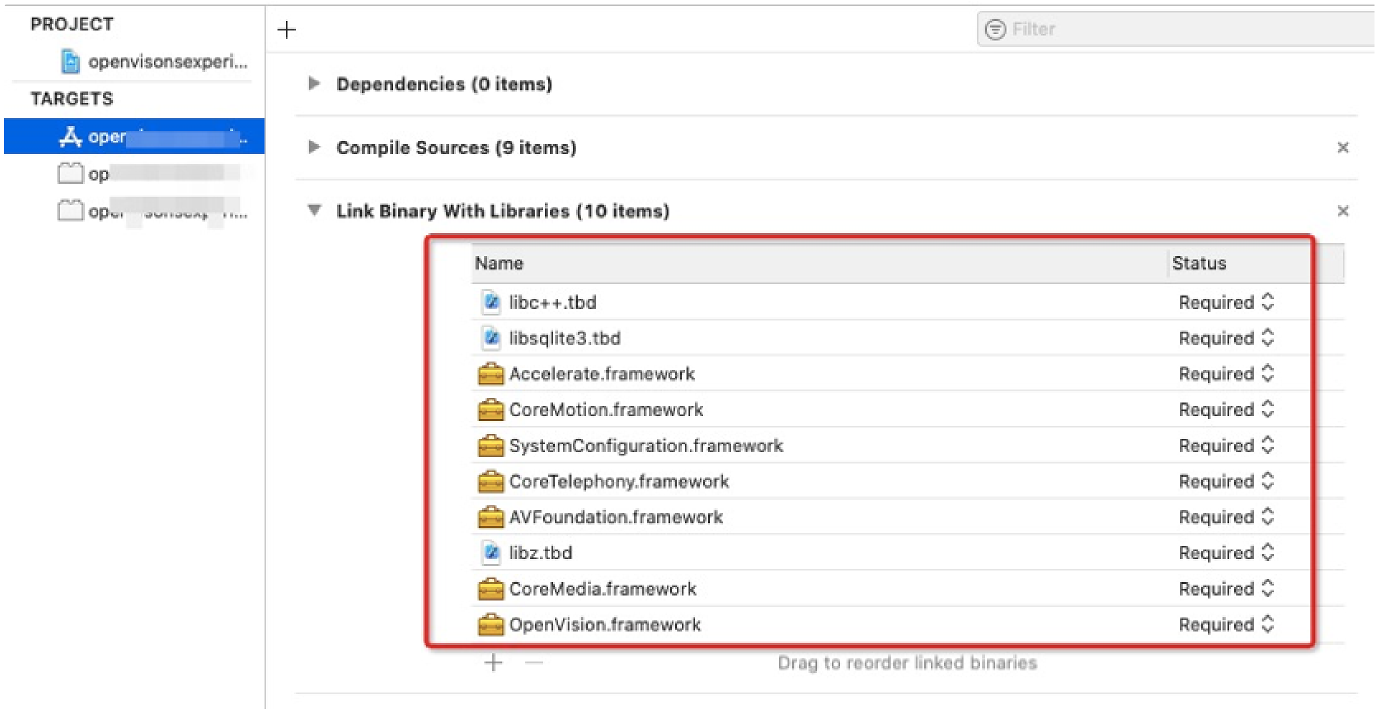
bankcard.bin
bankcard.license
driverlicense_back.bin
driverlicense_back.license
driverlicense_front.bin
driverlicense_front.license
humanpose.bin
humanpose.license
idcard_back.bin
idcard_back.license
idcard_front.bin
idcard_front.license
openvision_sdk_ios.zip
segmentation.bin
segmentation.license
vehiclelicense_back.bin
vehiclelicense_back.license
vehiclelicense_front.bin
vehiclelicense_front.license
vehicleplate.bin
vehicleplate.license

2、解压openvison_sdk_ios.zip,得到OpenVision.framework。项目下新建Frameworks，放入解压得到的OpenVision.framework，在OPV.bundle文件夹下放入bin、license文件（需要集成什么能力导入对应的bin、license文件即可）。如下图：



Frameworks
libc++.tbd
libsqlite3.tbd
Accelerate.framework
CoreMotion.framework
SystemConfiguration.framework
CoreTelephony.framework
AVFoundation.framework
libz.tbd
CoreMedia.framework
OpenVision.framework
OPV.bundle
bankcard
bankcard.bin
bankcard.license
humanpose
humanpose.bin
humanpose.license
scan@2x.png
segmentation
segmentation.bin
segmentation.license

3、需要集成一些系统的库，项目设置target –> 选项卡Build Phases –>Linked Binary With Libraries如下图：



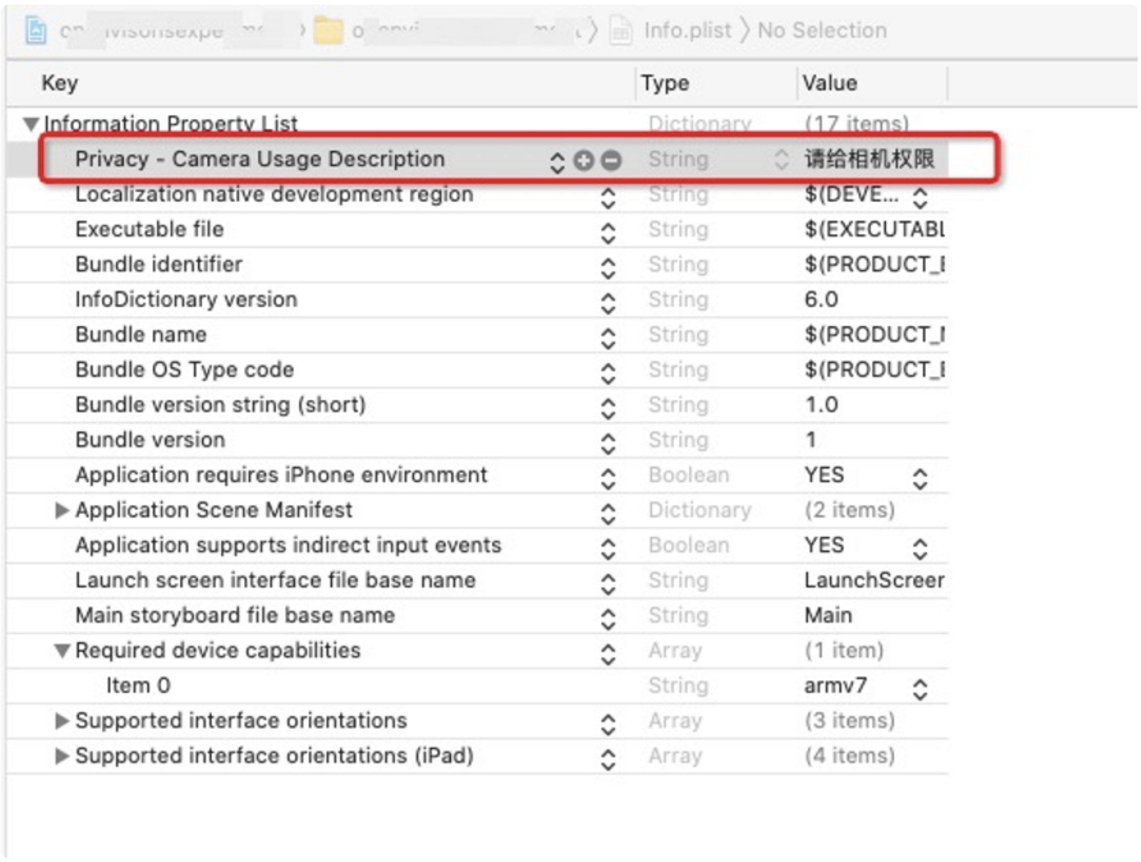
PROJECT
openvisonsexperi...
TARGETS

Dependencies (0 items)
Compile Sources (9 items)
Link Binary With Libraries (10 items)

Name	Status
libc++.tbd	Required
libsqlite3.tbd	Required
Accelerate.framework	Required
CoreMotion.framework	Required
SystemConfiguration.framework	Required
CoreTelephony.framework	Required
AVFoundation.framework	Required
libz.tbd	Required
CoreMedia.framework	Required
OpenVision.framework	Required

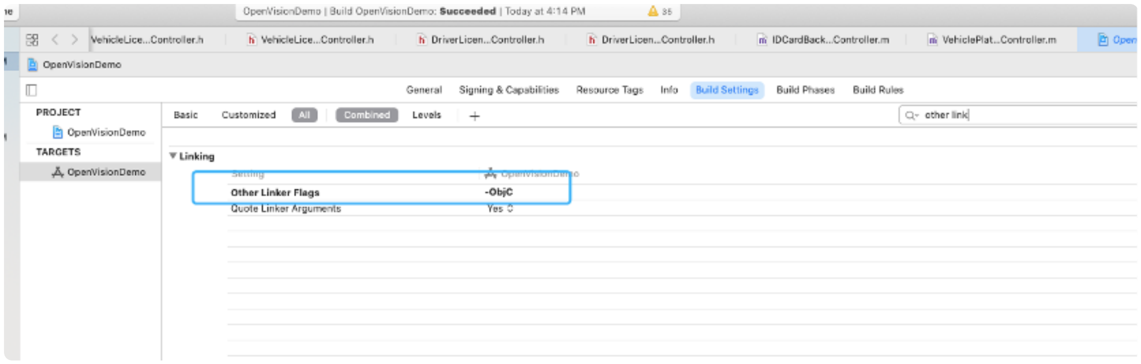
Drag to reorder linked binaries

4、需要配置相机的权限，项目下的Info.plist文件并在target 中添加如下flag，如下图：



Info.plist 〉 No Selection

Key	Type	Value
▼Information Property List	Dictionary	(17 items)
Privacy - Camera Usage Description	String	请给相机权限
Localization native development region	String	$(DEVE...
Executable file	String	$(EXECUTABL
Bundle identifier	String	$(PRODUCT_I
InfoDictionary version	String	6.0
Bundle name	String	$(PRODUCT_I
Bundle OS Type code	String	$(PRODUCT_I
Bundle version string (short)	String	1.0
Bundle version	String	1
Application requires iPhone environment	Boolean	YES
▶Application Scene Manifest	Dictionary	(2 items)
Application supports indirect input events	Boolean	YES
Launch screen interface file base name	String	LaunchScreer
Main storyboard file base name	String	Main
▼Required device capabilities	Array	(1 item)
Item 0	String	armv7
▶Supported interface orientations	Array	(3 items)
▶Supported interface orientations (iPad)	Array	(4 items)



OpenVisionDemo | Build OpenVisionDemo: **Succeeded** | Today at 4:14 PM

▼Linking

Setting	OpenVisionDemo
Other Linker Flags	**-ObjC**
Quote Linker Arguments	Yes

二、功能实现：

1、通过相机预览实现银行卡、身份证（正、反面）、驾驶证（正、反面）、行驶证（正、反面）、车牌的信息识别。（注：相关功能的实现依赖于framework库中是否包含该功能及项目中是否有对应的bin和license文件）。集成过程如下：

在调用证件扫描功能的类中引入下面的头文件：

BaseViewController　　Plain Text　复制代码

```
#import <OpenVision/OpenVerison.h>
// 声明相机类对象、算法类对象、扫描位置框，遵循OPVCameraDelegate 代理,
//卡类识别除了Engine声明和加载文件license、bin文件不同,
//其他都相同（可封装基类统一处理，详见Demo）:

@interface ViewController () <OPVCameraDelegate>
//声明相机类对象
@property (nonatomic, strong) OPVCamera    *opvCamera;
//声明算法类对象（银行卡）
@property (nonatomic, strong) OPVBankCardEngine *ocrEngine;
//添加扫描位置区域，可根据需求自行设置位置、大小，一般设置为身份证大小比例。
@property (nonatomic, strong) UIImageView *imageView;
@end

//视图添加相机View，设置扫描区域框:
//添加相机View
CGRect frame = self.view.bounds;
OPVCamera *opvCamera = [[OPVCamera alloc] initWithCameraFrame:frame
  cameraHandler:^(OPVCameraRunningStatus status, NSError *error) {
      NSLog(@"相机启动状态码%ld",status);
}];
[self.view addSubview:opvCamera.cameraView];
opvCamera.delegate = self;
_opvCamera = opvCamera;
//扫描区域框设置
CGRect rect = self.view.bounds;
//位置大小可根据需求自行设置
CGRect scanFrame = CGRectMake(0, rect.size.height*0.15,
  rect.size.width, rect.size.height*0.3516);
//资源文件为一个蓝色的框，可自行设置，也可查看Demo文件设置
NSString *bundlePath = [[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"OPV.bundle"];
NSString * scanImagePath = [bundlePath stringByAppendingPathComponent:@"scan"];
UIImage *scanImage = [UIImage imageNamed:scanImagePath];
_imageView = [[UIImageView alloc] initWithFrame:scanFrame];
[self.view addSubview:_imageView];
_imageView.image = scanImage;
_imageView.backgroundColor = [UIColor clearColor];

//加载算法相关license、bin文件，配置算法相关参数，初始化算法。
- (void)crateXMediaEngine {
  //create xmedia 相关文件加载
  NSString *bundlePath = [[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"OPV.bundle"];
  OPVConfig *config = [[OPVConfig alloc] init];
  config.license =  [bundlePath stringByAppendingPathComponent:@"bankcard.license"];
  config.model = [bundlePath stringByAppendingPathComponent:@"bankcard.bin"];
  //kOPVProcessFullPictureOutput: 算法处理结果中是否输出处理成功当前帧图片 0不输出 1输出
  //kOPVProcessTimeInterval: 50ms处理一次算法
  config.options = @{kOPVProcessFullPictureOutput:@(1),kOPVProcessTimeInterval:@(0.05)};

  NSError *error;
  _ocrEngine = [[OPVBankCardEngine alloc] initWithConfig:config error:&error];
  if (!error) {
    //扫描区域设置, 位置即为上面设置扫描框的位置。不设置的话扫描区域为整个相机View, 为提高扫描准确度
    //建议设置
    [_opvCamera attachEngine:_ocrEngine options:@{kOPVOptionsROI:@[@(0),@(0.15),@(1.0),@(0.3516)]}];
  } else {
    NSLog(@"ocrEngine init failed!");
  }
}
```

2.证件:

Plain Text | 复制代码

```
//身份证正面:
@property (nonatomic, strong) OPVIDCardEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"IDCard/idcard_front.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"IDCard/idcard_front.bin"];
//身份证反面:
@property (nonatomic, strong) OPVIDCardEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"IDCard/idcard_back.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"IDCard/idcard_back.bin"];
```

3.车牌:

Plain Text 复制代码

```
//车牌:
@property (nonatomic, strong) OPVPlateNumberEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"VehiclePlate/vehicleplate.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"VehiclePlate/vehicleplate.bin"];
```

4.行驶证

Plain Text 复制代码

```
//行驶证正面:
@property (nonatomic, strong) OPVVehicleLicenseEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"VehicleLicense/vehiclelicense_front.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"VehicleLicense/vehiclelicense_front.bin"];
//行驶证背面:
@property (nonatomic, strong) OPVVehicleLicenseEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"VehicleLicense/vehiclelicense_back.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"VehicleLicense/vehiclelicense_back.bin"];
```

5.驾驶证

Plain Text 复制代码

```
//驾驶证正面:
@property (nonatomic, strong) OPVDrivingLicenseEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"DriverLicense/driverlicense_front.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"DriverLicense/driverlicense_front.bin"];
//驾驶证背面:
@property (nonatomic, strong) OPVDrivingLicenseEngine  *opvOCREngine;
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"DriverLicense/driverlicense_back.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"DriverLicense/driverlicense_back.bin"];
```

6.调用算法:

Plain Text 复制代码

```
//调用算法:
[self.opvCamera attachEngine:self.ocrEngine options:@{kOPVOptionsROI:@[@(0),@(0.15),@(1.0),@(0.3516)]}];//

//算法结果回调:
- (void)OPVResultWithEngine:(OPVBaseEngine *)engine result:(OPVResult *)result {
if (!result.error && result.cvResult.count > 0) {
NSString *resultText = @"";
for (int i = 0; i < result.cvResult.count; i++) {
    if (result.cvResult[i].key.length > 0) {
resultText = [resultText stringByAppendingString:[NSString stringWithFormat:@"%@: ",result.cvResult[i].key]];
} else {
resultText = [resultText stringByAppendingString:@"label:"];
}
resultText = [resultText stringByAppendingString:result.cvResult[i].label];
resultText = [resultText stringByAppendingFormat:@"\n"];
resultText = [resultText stringByAppendingString:@"conf:"];
resultText = [resultText stringByAppendingString:[NSString stringWithFormat:@"%f ",result.cvResult[i].conf]];
}
NSLog(@"%@",resultText);
__weak typeof(self) wself = self;
//通知主线程刷新
dispatch_async(dispatch_get_main_queue(), ^{
__strong typeof(self) sSelf = wself;
//result.image、resultText为结果返回，可自定义控件接收展示
sSelf.roiImageResult.image = result.image;
sSelf.recResult.text = resultText;
});
}
}
//生命周期管理:
- (void)viewWillDisappear:(BOOL)animated {
if (self.opvCamera) {
//关闭相机，关闭后想再次打开调startCamera
[self.opvCamera stopCamera];
//销毁整个算法引擎
[self.opvCamera removeEngine:_ocrEngine];
}
}
```

7.实现人像抠图的效果。集成过程如下:

在调用证件扫描功能的类中引入这个头文件

Plain Text | 复制代码

```
#import <OpenVision/OpenVerison.h>
基类BaseViewController可以用最上面的代码
加载分割算法相关license、bin文件，配置算法相关参数，初始化算法:
@property (nonatomic, strong) OPVSegmentationEngine *segEngine;

//create xmedia 加载相关文件
NSString *bundlePath = [[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"OPV.bundle"];
OPVConfig *config = [[OPVConfig alloc] init];
config.license =  [bundlePath stringByAppendingPathComponent:@"segmentation.license"];
config.model = [bundlePath stringByAppendingPathComponent:@"segmentation.bin"];
config.options = @{kOPVProcessTimeInterval:@(0.07)};//70ms处理一次算法

_segEngine = [[OPVSegmentationEngine alloc] initWithConfig:config error:nil];
//设置可识别区域，不传的话识别整个相机View
[_opvCamera attachEngine:_segEngine options:@{kOPVOptionsROI:@[@(0),@(0),@(1.0),@(1.0)]}];
```

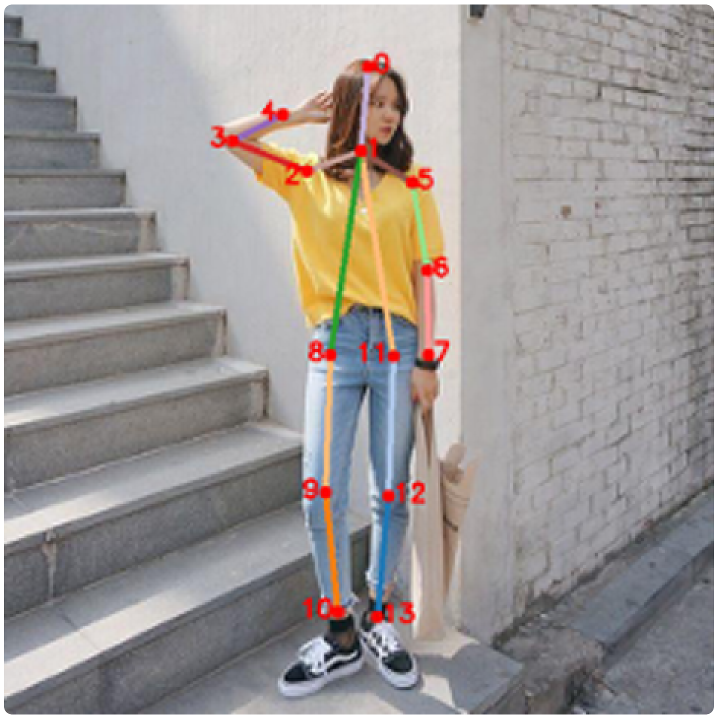
分割算法结果回调

Plain Text 复制代码

```
- (void)OPVResultWithEngine:(OPVBaseEngine *)engine result:(OPVResult *)result {
if (!result.error) {
result = (OPVSegmentationResult *)result;
unsigned char *data = ((OPVSegmentationResult *)result).data;
int pixFormat = ((OPVSegmentationResult *)result).format;
int width = ((OPVSegmentationResult *)result).width;
int height = ((OPVSegmentationResult *)result).height;
UIImage *rstImg = [OPVUtils convertToImageWithPixelData:data format:(OPVPixelFomat)pixFormat width:width height:height];
__weak typeof(self) wself = self;
//通知主线程刷新
dispatch_async(dispatch_get_main_queue(), ^{
__strong typeof(self) sSelf = wself;
//rstImg为结果图片，可自定义控件接收展示
sSelf.imageView.image = rstImg;
});
}
}
```

8.人体姿态，获取人体描边的位置点集成过程如下：

人体骨架连线规则，每组固定返回14个点，排序即为下标，按照固定点连线即可（每个坐标点都有参数conf， conf大于0 再画这个点和对应连线）如图：



在调用证件扫描功能的类中引入这个头文件：

Plain Text | 复制代码

```
#import <OpenVision/OpenVerison.h>
声明相机类对象、算法类对象，遵循OPVCameraDelegate 代理：
@interface ViewController () <OPVCameraDelegate>
//声明相机类对象
@property (nonatomic, strong) OPVCamera    *opvCamera;
//声明算法类对象
@property (nonatomic, strong) OPVHumanPoseEngine *poseEngine;
@end

//视图添加相机View：
//添加相机View
CGRect frame = self.view.bounds;
OPVCamera *opvCamera = [[OPVCamera alloc] initWithCameraFrame:frame cameraHandler:^(OPVCameraRunningStatus status, NSError *error) {
NSLog(@"相机启动状态码%ld",status);
}];
[self.view addSubview:opvCamera.cameraView];
opvCamera.delegate = self;
_opvCamera = opvCamera;
//加载算法相关license、bin文件，配置算法相关参数，初始化算法：
- (void)crateHumanposeEngine {
    //create xmedia 加载相关文件
    NSString *bundlePath = [[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"OPV.bundle"];
    OPVConfig *config = [[OPVConfig alloc] init];
    config.license =  [bundlePath stringByAppendingPathComponent:@"humanpose/humanpose.license"];
    config.model = [bundlePath stringByAppendingPathComponent:@"humanpose/humanpose.bin"];
    config.options = @{kOPVProcessTimeInterval:@(0.07)};//算法70ms处理一次
    _poseEngine = [[OPVHumanPoseEngine alloc] initWithConfig:config error:nil];
    [self.opvCamera attachEngine:_poseEngine options:@{}];
}

```

人体姿态，获取人体描边算法结果回调

Plain Text 复制代码

```
- (void)OPVResultWithEngine:(OPVBaseEngine *)engine result:(OPVResult *)result {
if (!result.error) {
OPVHumanPoseResult *poseResult = (OPVHumanPoseResult *)result;
//poseResult.humanPoseResult 为坐标点数据数组，多个人像的话会有多个数组，每个数组有
//14个人体点，可根据需求自行画出各点，并连线（可参考Demo）进行其他操作
if ([poseResult.humanPoseResult count]==0) {
__weak typeof(self) wself = self;
dispatch_async(dispatch_get_main_queue(), ^{
//无数据返回
[wself.humanPoseLineView setPosePointArray:nil];
});
return;
}

NSMutableArray *pointArrays = [[NSMutableArray alloc] init];
for (int i = 0; i < poseResult.humanPoseResult.count; i++) {
NSArray *items = poseResult.humanPoseResult[i];

NSMutableArray *tempPointArr = [[NSMutableArray alloc] init];
for (int j = 0; j < items.count; j ++) {
CVResult *item = items[j];
CGPoint point = CGPointMake(([item.pos[0] floatValue]) * poseResult.pixelWidth, ([item.pos[1] floatValue]) * poseResult.pixelHeight);
if (i==0 && j==0) {
NSLog(@"human head pos:[%f,%f]",point.x,point.y);
}
// frame -> view 坐标转换 _viewWidth _viewHeight为视图宽高
if (poseResult.pixelWidth !=0 && poseResult.pixelHeight != 0) {
CGPoint newPoint = CGPointMake(point.x * _viewWidth / poseResult.pixelWidth, point.y * _viewHeight / poseResult.pixelHeight);
NSValue *pointValue = [NSValue valueWithCGPoint:newPoint];
[tempPointArr addObject:pointValue];
} else {
NSValue *pointValue = [NSValue valueWithCGPoint:point];
[tempPointArr addObject:pointValue];
}
}
//pointArrays为所有转换完的点数据，pointArrays[i]为每组数据，每组有固定14个点
[pointArrays addObject:tempPointArr];
}
}
}

```

9.Vin识别功能实现,集成过程如下：

在调用Vin扫描功能的类中引入这个头文件：

Plain Text | 复制代码

```
#import <OpenVision/OpenVerison.h>
//声明相机类对象、算法类对象，遵循OPVCameraDelegate 代理：
@interface ViewController () <OPVCameraDelegate>
//声明相机类对象
@property (nonatomic, strong) OPVCamera    *opvCamera;
//声明算法类对象
@property (nonatomic, strong) OPVVinCodeEngine *vinEngine;
@end
//视图添加相机view
OPVCamera *opvCamera = [[OPVCamera alloc] initWithCameraFrame:frame   cameraHandler:^(OPVCameraRunningStatus status, NSError *error) {
        NSLog(@"相机启动状态码%ld",status);
    }];
[self.view addSubview:opvCamera.cameraView];
opvCamera.delegate = self;
_opvCamera = opvCamera;
//加载vin模块能力
- (void)crateVinSpotEngine {
    //create xmedia 加载相关文件
    NSString *bundlePath = [[[NSBundle mainBundle] bundlePath] stringByAppendingPathComponent:@"OPV.bundle"];
    OPVConfig *config = [[OPVConfig alloc] init];
    config.license =  [bundlePath stringByAppendingPathComponent:@"VinCodeSpot/vincode.license"];
    config.model = [bundlePath stringByAppendingPathComponent:@"VinCodeSpot/vincode.bin"];
    config.options = @{kOPVProcessTimeInterval:@(0.05)};//算法50ms处理一次

    _vinEngine = [[OPVVinCodeEngine alloc] initWithConfig:config error:nil];
      [self.opvCamera attachEngine:_poseEngine options:@{}];
}

```

Vin算法结果回调：

Plain Text | 复制代码

```
- (void)OPVResultWithEngine:(OPVBaseEngine *)engine result:(OPVResult *)result {
    if (!result.error) {
        result = (OPVSegmentationResult *)result;
        unsigned char *data = ((OPVSegmentationResult *)result).data;
        int pixFormat = ((OPVSegmentationResult *)result).format;
        int width = ((OPVSegmentationResult *)result).width;
        int height = ((OPVSegmentationResult *)result).height;
        UIImage *rstImg = [OPVUtils convertToImageWithPixelData:data format:(OPVPixelFomat)pixFormat width:width height:height];
        __weak typeof(self) wself = self;
        //通知主线程刷新
        dispatch_async(dispatch_get_main_queue(), ^{
            __strong typeof(self) sSelf = wself;
            //rstImg为结果图片，可自定义控件接收展示
            sSelf.imageView.image = rstImg;
        });
    }
}

```

其他通用设置

相机前后摄像头切换

Plain Text | 复制代码

```
[self.opvCamera switchCamera:nil];
```

生命周期管理

Plain Text | 复制代码

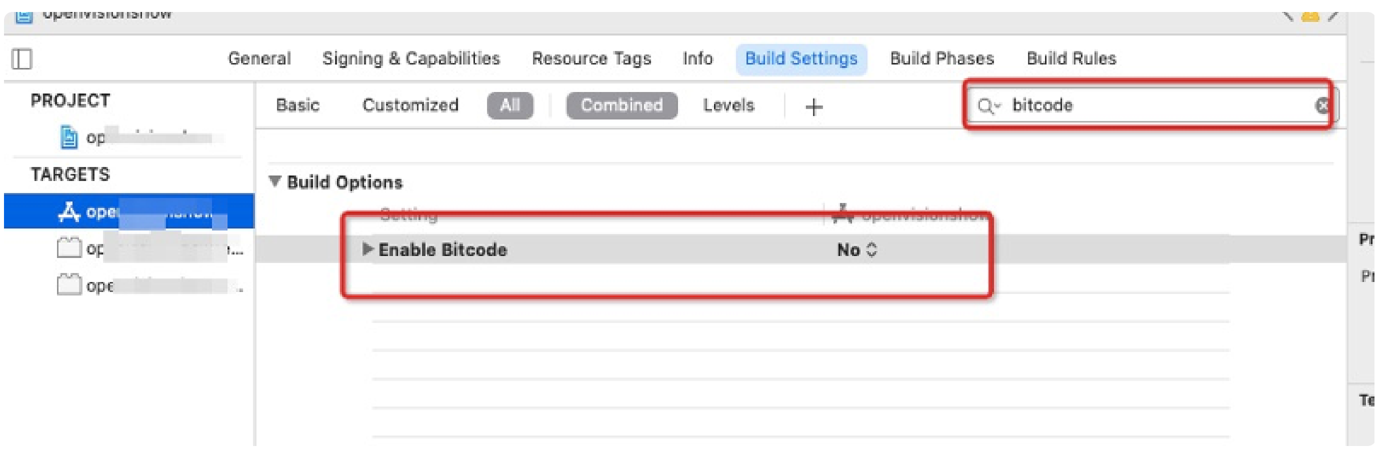
```
- (void)viewWillDisappear:(BOOL)animated {
if (self.opvCamera) {
//关闭相机，关闭后想再次打开调startCamera
[self.opvCamera stopCamera];
//销毁整个算法引擎  _segEngine 分割能力，也可以是其他任何能力
[self.opvCamera removeEngine:_segEngine];
}
}

```

三、其他：

1、证件检测必须为所选证件项进行检测，包括正、反面的区别，否则会检测没反应或检测失败。

2、封装好的相机OPVCamera相关操作请参考OPVCamera.h文件。

3、报错 You must rebuild it with bitcode enabled (Xcode setting ENABLE_BITCODE) 解决方法。



Enable Bitcode No