

自定义相机预览 iOS SDK说明文档 v1.0.0.20170510

• 文档目录

一、新建项目准备工作

- 1) 系统 framework 导入
- 2) IntSig framework 导入

二、AVFoundation 说明

- 1) 相机基本实现步骤
- 2) 初始化AVFoundation自定义相机预览

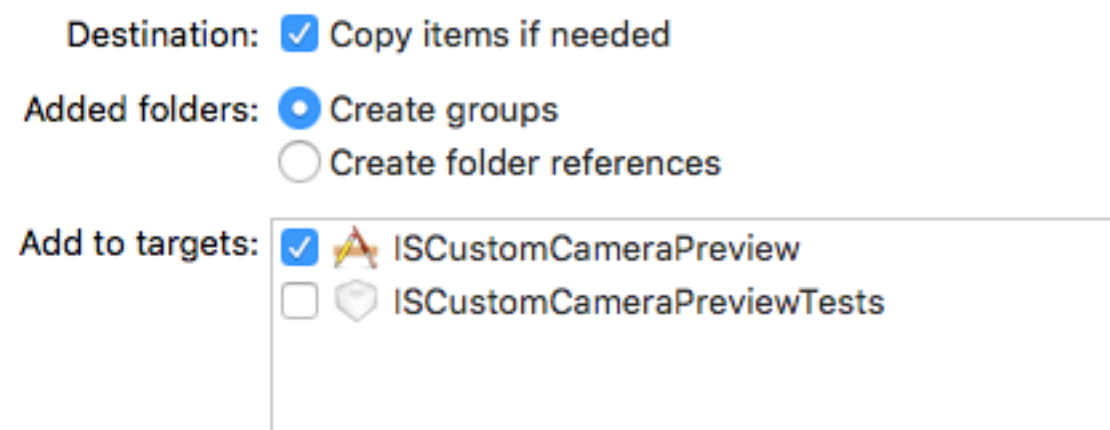
一、新建项目准备工作

1) 系统framework导入

- a) libc++.tbd
- b) CoreMedia.framework
- c) AssetsLibrary.framework
- d) AVFoundation.framework

2) IntSig framework导入

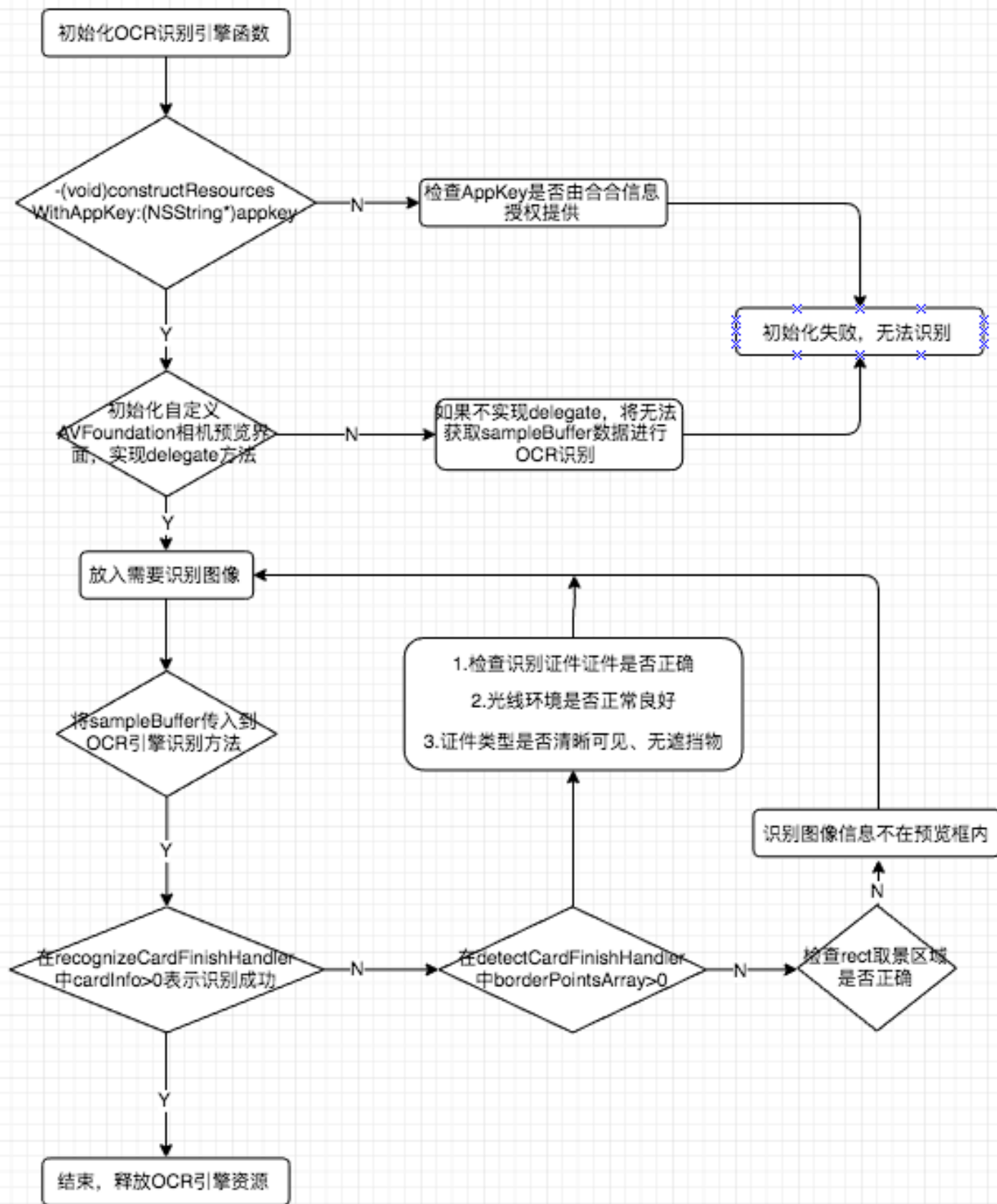
- a) 请根据提供的对应的SDK进行导入，将Release文件中的内容导入到Project中
- b) 在导入的时候需要将"Copy items if needed"进行勾选



二、AVFoundation 说明

1) 相机基本实现步骤

- 1.初始化OCR识别引擎
- 2.自定义AVFoundation相机预览
- 3.捕捉输出信息传入OCR识别引擎中
- 4.输出识别结果，释放OCR引擎资源



二、AVFoundation 说明

2) 初始化AVFoundation自定义相机预览

- ① 捕捉会话——AVCaptureSession
- ② 捕捉输入——AVCaptureDeviceInput
- ③ 捕捉预览——AVCaptureVideoPreviewLayer
- ④ 捕捉连接——AVCaptureConnection
- ⑤ 视频——AVCaptureVideoDataOutput
- ⑥ 方向问题——Orientation
- ⑦ 手动/自动聚焦
- ⑧ 建立预览区域
- ⑨ 将获取的每一帧数据传入到OCR识别引擎中

备注：具体完整实现过程请参考”ISIDReaderPreview”中ViewController.m中代码实现

① 捕捉会话——AVCaptureSession

AVCaptureSession(捕捉会话管理): 它从物理设备得到数据流, 比如摄像头和麦克风, 输出到一个或多个目的地。

capture session 可以通过会话预设值(session preset), 来控制捕捉数据的格式和质量, 下面是创建一个 session 的代码:

```
//建立视频捕获
- (void)setupAVCapture
{
    //创建AVCaptureSession对象
    AVCaptureSession *session = [AVCaptureSession new];
    [self configSetSessionPreset:session];
}

//配置会话预设, 用于获取不同型号的像素, 以便于进行比例转换
- (void)configSetSessionPreset:(AVCaptureSession *)session
{
    if ([session canSetSessionPreset:AVCaptureSessionPreset1920x1080])
    {
        NSLog(@"AVCaptureSessionPreset1920x1080");
        [session setSessionPreset:AVCaptureSessionPreset1920x1080];
        self.imageFrameWidth = 1920;
        self.imageFrameHeight = 1080;
    }
    else if ([session canSetSessionPreset:AVCaptureSessionPreset640x480])
    {
        NSLog(@"AVCaptureSessionPreset640x480");
        [session setSessionPreset:AVCaptureSessionPreset640x480];
        self.imageFrameWidth = 640;
        self.imageFrameHeight = 480;
    }
    else if ([session canSetSessionPreset:AVCaptureSessionPreset352x288])
    {
        NSLog(@"AVCaptureSessionPreset352x288");
        [session setSessionPreset:AVCaptureSessionPreset352x288];
        self.imageFrameWidth = 352;
        self.imageFrameHeight = 288;
    }
    else
    {
        NSLog(@"failed setSessionPreset");
        return;
    }
    self.captureSession = session;
    [self configCaptureDevice:session];
}
```

② 捕捉输入——AVCaptureDeviceInput

AVCaptureDeviceInput(捕捉设备): 它实际上是为摄像头和麦克风等物理设备定义的接口, 我们可以通过它来访问或控制这些硬件设备。比如控制摄像头的对焦、曝光等。

```
//配置相机设备属性
- (void)configCaptureDevice:(AVCaptureSession *)session
{
    //获取视频输入设备, 该方法默认返回iPhone的后置摄像头
    AVCaptureDevice *device = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];

    /*
    我们需要使用AVCaptureDeviceInput来让设备添加到session中,
    AVCaptureDeviceInput负责管理设备端口。我们可以理解它为设备的抽象。
    一个设备可能可以同时提供视频和音频的捕捉。
    我们可以分别用AVCaptureDeviceInput来代表视频输入和音频输入。
    */
    AVCaptureDeviceInput *deviceInput = [AVCaptureDeviceInput deviceInputWithDevice:device error:nil];

    if ( [session canAddInput:deviceInput] )
        [session addInput:deviceInput];

    [self configCaptureVideoDataOutput:session];
}
```

③ 捕捉预览——AVCaptureVideoPreviewLayer

AVCaptureVideoPreviewLayer(捕捉预览): 它是CALayer的子类, 可被用于自动显示相机产生的实时图像。

preview layer可以控制视频内容渲染的缩放和拉效果

```
//配置预览视图层
- (void)configPreviewLayer:(AVCaptureSession *)session
{
    self.previewLayer = [[AVCaptureVideoPreviewLayer alloc] initWithSession:session];
    [self.previewLayer setBackgroundColor:[[UIColor blackColor] CGColor]];
    /*
     定义一个字符串如何视频是在一个avcapturevideopreviewlayer边界矩形显示。
     */
    [self.previewLayer setVideoGravity:AVLayerVideoGravityResizeAspect];

    CALayer *rootLayer = [self.view layer];
    //阴影效果失效(阴影部分超出父视图, 被剪掉了)
    [rootLayer setMasksToBounds:YES];

    CGFloat screenWidth = [[UIScreen mainScreen] bounds].size.width;
    CGFloat screenHeight = [[UIScreen mainScreen] bounds].size.height;

    if (screenWidth / self.imageFrameHeight > screenHeight / self.imageFrameWidth)
    {
        CGFloat height = self.imageFrameWidth * screenWidth / self.imageFrameHeight;
        CGRect frameRect = CGRectMake(0, (screenHeight - height) / 2, screenWidth, height);
        [self.previewLayer setFrame:frameRect];
        self.expandOrientation = ISPreviewExpandOrientationPortrait;
    }
    else
    {
        CGFloat width = self.imageFrameHeight * screenHeight / self.imageFrameWidth;
        CGRect frameRect = CGRectMake((screenWidth - width) / 2, 0, width, screenHeight);
        [self.previewLayer setFrame:frameRect];
        self.expandOrientation = ISPreviewExpandOrientationLandscape;
    }

    [rootLayer addSublayer:self.previewLayer];

    //开始捕捉
    [session startRunning];
}
```

注意:

1. 它看起来有点像输出, 但其实不是, 它仅用来预览摄像头捕捉的画面。真正用于输出的是AVCaptureSession (previewLayer拥有session, session拥有outputs);
2. 它的坐标系和屏幕的坐标系不同, 如果点击某区域实现对焦时, 我们需要将设备的坐标系转换为实时预览图的坐标;
3. 它的坐标原点永远都在右上角, 这和我们手机的坐标系不同, 手机坐标系的原点是不变的。因此拍照或录制视频是, 要先得得到设备方向(关于方向问题, 后面会详解), 计算输出的旋转角度。

④ 捕捉连接——AVCaptureConnection

捕捉连接负责将捕捉会话接收的媒体类型和输出连接起来，AVCaptureVideoDataOutput可以接受视频数据。会话通过捕捉连接，确定输入视频。通过对捕捉连接的访问，可以对信号流进行底层控制，比如禁用某些特定的连接。

```
//AVCaptureOutput delegate, 分析视频流
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer fromConnection:
(AVCaptureConnection *)connection
{
    CFRetain(sampleBuffer);
    /*
    CMSampleBufferRef
    在取得 CMSampleBufferRef 之后，还必须透过一连串转换才能够得到
    UIImage, CMSampleBufferRef -> CVImageBufferRef -> CGContextRef -> CGImageRef -> UIImage
    你可以将以下程序代码任意实作于上述两个内部函数中来取得连续影像片段中的 UIImage。

    cardRect 预览区域范围，用于SDK引擎进行核验
    */
    [[ISBankCardController sharedISOpenSDKController] detectCardWithOutputSampleBuffer:sampleBuffer cardRect:self.borderRectInImage
    detectCardFinishHandler:^(int result, NSArray *borderPointsArray)
    {
        } recognizeCardFinishHandler:^(NSDictionary *cardInfo)
    {
        if ([cardInfo count] > 0)
        {
            [self.captureSession stopRunning];
            dispatch_async(dispatch_get_main_queue(), ^{
                ISBankCardResultViewController *resultVC = [[ISBankCardResultViewController alloc] initWithInfo:cardInfo];
                [self presentViewController:resultVC animated:YES completion:^
                {
                    }]);
            });
        }
    }];
    CFRelease(sampleBuffer);
}
```


⑤ 视频——AVCaptureVideoDataOutput

AVCaptureVideoDataOutput(视频数据输出): 它输出硬件实时捕捉的视频数字样本

```
//配置捕获视频数据输出
- (void)configCaptureVideoDataOutput:(AVCaptureSession *)session
{
    //为了从session中取得数据,我们需要创建一个AVCaptureOutput
    self.videoDataOutput = [AVCaptureVideoDataOutput new];

    // we want YUV, both CoreGraphics and OpenGL work well with 'YUV'
    //其中重要的属性就一个, kCVPixelBufferPixelFormatTypeKey, 指定解码后的图像格式, 必须指定成NV12, 苹果的硬解码器只支持NV12。
    NSDictionary *rgbOutputSettings = [NSDictionary dictionaryWithObject:
                                        [NSNumber numberWithInt:kCVPixelFormatType_420YpCbCr8BiPlanarFullRange] forKey:
    (id)kCVPixelBufferPixelFormatTypeKey];

    [self.videoDataOutput setVideoSettings:rgbOutputSettings];
    [self.videoDataOutput setAlwaysDiscardsLateVideoFrames:YES];

    /*
    为了分析视频流,我们需要为output设置delegate,并且指定delegate方法在哪个线程被调用。需要主要的是,线程必须是串行的,确保视频帧按序到达。
    设置output delegate,将output添加至session,在代理方法中分析视频流
    */
    self.videoDataOutputQueue = dispatch_queue_create("VideoDataOutputQueue", DISPATCH_QUEUE_SERIAL);
    [self.videoDataOutput setSampleBufferDelegate:self queue:self.videoDataOutputQueue];

    if ([session canAddOutput:self.videoDataOutput])
    {
        [session addOutput:self.videoDataOutput];
    }
    //视频连接
    [[self.videoDataOutput connectionWithMediaType:AVMediaTypeVideo] setEnabled:YES];

    [self configPreviewLayer:session];
}
```

⑥ 方向问题——Orientation

设备方向device orientation

```
// 设备方向
UIDevice *device = [UIDevice currentDevice] ;
switch (device.orientation) {
    case UIDeviceOrientationFaceUp:
        NSLog(@"屏幕朝上平躺");
        break;
    case UIDeviceOrientationFaceDown:
        NSLog(@"屏幕朝下平躺");
        break;
    case UIDeviceOrientationUnknown:
        NSLog(@"未知方向");
        break;
    case UIDeviceOrientationLandscapeLeft:
        NSLog(@"屏幕向左横置");
        break;
    case UIDeviceOrientationLandscapeRight:
        NSLog(@"屏幕向右横置");
        break;
    case UIDeviceOrientationPortrait:
        NSLog(@"屏幕直立");
        break;
    case UIDeviceOrientationPortraitUpsideDown:
        NSLog(@"屏幕直立, 上下颠倒");
        break;
}
```

⑦ 手动/自动聚焦

```
//建立视图手势
- (void)setUpGesture
{
    UITapGestureRecognizer *tapGestureRecognizer = [[UITapGestureRecognizer alloc] initWithTarget:self action:@selector(focusGesture:)];
    [self.view addGestureRecognizer:tapGestureRecognizer];
}

/* #pragma mark Add TapGestureRecognizer */
- (void)focusGesture:(UITapGestureRecognizer*)gesture{
    CGPoint point = [gesture locationInView:gesture.view];
    [self focusAtPoint:point];
}

/* AVCaptureFlashMode 闪光灯 ; AVCaptureFocusMode 对焦 ; AVCaptureExposureMode 曝光 ; AVCaptureWhiteBalanceMode 白平衡
闪光灯和白平衡可以在生成相机时候设置 ; 曝光要根据对焦点的光线状况而决定,所以和对焦一块写 ; point为点击的位置 */
- (void)focusAtPoint:(CGPoint)point{
    //手动点击对焦
    CGSize size = self.view.bounds.size;
    CGPoint focusPoint = CGPointMake( point.y /size.height ,1-point.x/size.width );
    NSError *error;
    AVCaptureDevice *deviceFocusAtPoint = [AVCaptureDevice defaultDeviceWithMediaType:AVMediaTypeVideo];
    if ([deviceFocusAtPoint lockForConfiguration:&error]) {
        //对焦模式和对焦点
        if ([deviceFocusAtPoint isFocusModeSupported:AVCaptureFocusModeAutoFocus]) {
            [deviceFocusAtPoint setFocusPointOfInterest:focusPoint];
            [deviceFocusAtPoint setFocusMode:AVCaptureFocusModeAutoFocus];
        }
        //曝光模式和曝光点
        if ([deviceFocusAtPoint isExposureModeSupported:AVCaptureExposureModeAutoExpose ]) {
            [deviceFocusAtPoint setExposurePointOfInterest:focusPoint];
            [deviceFocusAtPoint setExposureMode:AVCaptureExposureModeAutoExpose];
        }
        [deviceFocusAtPoint unlockForConfiguration];
        //设置对焦动画
        self.focusView.center = point;
        self.focusView.hidden = NO;
        [UIView animateWithDuration:0.3 animations:^(
            /*
             CGAffineTransformMakeScale 两个参数, 代表x和y方向缩放倍数。 (缩放:设置缩放比例) 仅通过设置缩放比例就可实现视图扑面而来和缩进频幕的效果。
             CGAffineTransformMakeTranslation (平移:设置平移量) CGAffineTransformMakeRotation (旋转:设置旋转角度)
            */
            self.focusView.transform = CGAffineTransformMakeScale(1.25, 1.25);
        )completion:^(BOOL finished) {
            [UIView animateWithDuration:0.5 animations:^(
                self.focusView.transform = CGAffineTransformIdentity;
            ) completion:^(BOOL finished) {
                self.focusView.hidden = YES;
            }];
        }];
    }
}
```

⑧ 建立预览区域

```
//建立预览区域边界视图
- (void)setupBorderView
{
    if (self.previewLayer == nil)
    {
        return;
    }
    CGSize screenSize = [[UIScreen mainScreen] bounds].size;
    CGFloat screenWidth = screenSize.width;
    CGFloat screenHeight = screenSize.height;
    UIDeviceOrientation orientation = [UIDevice currentDevice].orientation;
    BOOL isLandscape = YES;
    //angle 用于旋转角度
    CGFloat angle = 0;

    //orientaion根据屏幕Home方位进行判断
    if (orientation == UIDeviceOrientationPortrait || orientation ==
        UIDeviceOrientationPortraitUpsideDown)
    {
        isLandscape = NO;
    }
    else if (orientation == UIDeviceOrientationFaceUp && self.isFirstLaunch)
    {
        isLandscape = NO;
    }
    else if (orientation == UIDeviceOrientationLandscapeLeft || orientation ==
        UIDeviceOrientationLandscapeRight)
    {
        isLandscape = YES;
        if (orientation == UIDeviceOrientationLandscapeLeft)
        {
            angle = M_PI_2;
        }
        else
        {
            angle = M_PI_2 * 3;
        }
    }
    else
    {
        if (self.isLandscape != nil)
        {
            isLandscape = [self.isLandscape boolValue];
        }
        else
        {
            isLandscape = YES;
        }
    }
    self.isFirstLaunch = NO;
    if (self.isLandscape != nil && [self.isLandscape boolValue] == isLandscape)
    {
        return;
    }
    if (self.backgroundColor != nil)
    {
        [self.backgroundColor removeFromSuperlayer];
        self.backgroundColor = nil;
    }
    self.isLandscape = @(isLandscape);
    //先将borderRect范围设为空，后面根据比例重新计算赋值
    self.borderRectInImage = CGRectZero;
    //根据像素大小和预览区域大小计算视频取景框内比例，用于识别框内信息
    CGFloat scale = self.imageFrameWidth / self.previewLayer.frame.size.height;
    if (isLandscape)
    {
        //GoldenSectionRation 为黄金比例0.63，用于等比例转换
        if (screenWidth / screenHeight < GoldenSectionRation)
        {
            self.borderWidth = (int)(screenWidth * 7 / 8);
            self.borderHeight = self.borderWidth / GoldenSectionRation;
        }
        else
        {
            self.borderHeight = (int)(screenHeight * 7 / 8);
            self.borderWidth = self.borderHeight * GoldenSectionRation;
        }
    }
}
```

```
else
{
    if (screenHeight / screenWidth < GoldenSectionRation)
    {
        self.borderHeight = (int)(screenHeight * 7 / 8);
        self.borderWidth = self.borderHeight / GoldenSectionRation;
    }
    else
    {
        self.borderWidth = (int)(screenWidth * 7 / 8);
        self.borderHeight = self.borderWidth * GoldenSectionRation;
    }
}
//在视图(1920*1080)上取景区域x、y坐标轴,
CGFloat originX = (self.imageFrameWidth - self.borderHeight * scale) / 2;
CGFloat originY = (self.imageFrameHeight - self.borderWidth * scale) / 2;

//用于IS 识别引擎预览图片范围区域尺寸,
self.borderRectInImage = CGRectMake(originX, originY, self.borderHeight * scale, self.borderWidth *
scale);

//计算出预览区域的rect
CGRect borderRect = CGRectMake((screenSize.width - self.borderWidth)/2, (screenSize.height -
self.borderHeight)/2, self.borderWidth, self.borderHeight);
//进行super绘制
UIBezierPath *superPath = [UIBezierPath bezierPathWithRoundedRect:self.view.bounds cornerRadius:0];
[superPath setUsesEvenOddFillRule:YES];

//进行border绘制
UIBezierPath *borderPath = [UIBezierPath bezierPathWithRoundedRect:borderRect cornerRadius:10];
[borderPath setUsesEvenOddFillRule:YES];

[superPath appendPath:borderPath];

/*
渲染快速。CAShapeLayer使用了硬件加速，绘制同一图形会比用Core Graphics快很多。
*/

CAShapeLayer *fillLayer = [CAShapeLayer layer];
fillLayer.path = superPath.CGPath;
fillLayer.fillRule = kCAFillRuleEvenOdd;
fillLayer.fillColor = [UIColor colorWithRed:0 green:0 blue:0 alpha:1].CGColor;
fillLayer.opacity = 0.3;
[self.view.layer addSublayer:fillLayer];
self.backgroundColor = fillLayer;
if (self.imageView == nil)
{
    //添加取景框内默认图片显示
    UIImageView *imageView = [[UIImageView alloc] init];
    NSString *imageName = @"card";
    if ([NSLocale preferredLanguages] count > 0)
    {
        NSString *currentLanguage = [[NSLocale preferredLanguages] objectAtIndex:0];
        if ([currentLanguage hasPrefix:@"zh-Hans"])
        {
            imageName = @"card_cn";
        }
        else if ([currentLanguage hasPrefix:@"zh-Hant"] || [currentLanguage hasPrefix:@"zh-TW"] ||
[currentLanguage hasPrefix:@"zh-HK"])
        {
            imageName = @"card_tw";
        }
    }
    imageView.image = [UIImage imageNamed:imageName];
    [self.view addSubview:imageView];
    self.imageView = imageView;
}
if (isLandscape)
{
    //根据Home对默认显示图片进行旋转
    self.imageView.frame = CGRectMake(0, 0, borderRect.size.height, borderRect.size.width);
    self.imageView.center = CGPointMake(screenWidth / 2, screenHeight / 2);
    self.imageView.transform = CGAffineTransformMakeRotation(angle);
}
else
{
    self.imageView.transform = CGAffineTransformMakeRotation(angle);
    self.imageView.frame = borderRect;
}
}
```

⑨ 将获取的每一帧数据传入到OCR识别引擎中

```
//AVCaptureVideoDataOutput delegate
- (void)captureOutput:(AVCaptureOutput *)captureOutput didOutputSampleBuffer:(CMSampleBufferRef)sampleBuffer
fromConnection:(AVCaptureConnection *)connection
{
    CFRetain(sampleBuffer);
    //sampleBuffer 为预览是获取的视频流信息，用于生成图片
    //self.borderRectInImage 自定义拍摄取景框的尺寸大小，该尺寸的大小是根据像素和频幕尺寸比例进行转换
    [[ISOpenSDKStatus sharedISOpenSDKController] detectCardWithOutputSampleBuffer:sampleBuffer
cardRect:self.borderRectInImage detectCardFinishHandler:^(int result, NSArray *borderPointsArray)
    {

    } recognizeCardFinishHandler:^(NSDictionary *cardInfo)
    {
        if ([cardInfo count] > 0)
        {
            [self.captureSession stopRunning];
            dispatch_async(dispatch_get_main_queue(), ^{
                //将cardInfo中的内容就是当前OCR识别信息
            });
        }
    }];
    CFRelease(sampleBuffer);
}
```