常见问题FAQ

1.鉴权方式

答： SDK通过License(文件)来控制版本，想要使用SDK提供各项能力(人脸检测、美颜、美妆、贴纸、滤镜等等)要先获得商汤提供的授权文件，获取方式：1.通过线下方式获取 2.通过网络加载。

2.检测句柄handle已经创建成功，但是检测不到人脸

答：遇到这个问题可以采取一下步骤进行排查
(1) 查看输入图像数据是否正确；
(2) 查看输入的图像数据格式与设置是否相同，比如我输入的数据是YUV格式，但是格式设置的RGBA；
(3) 查看输入的旋转方向与输入书的数据方向是否相同，获取手机旋转方向的方法可查看Demo中getRotate方法的具体实现，还有就是要看一下手机是否开启了禁止旋转功能；
(4) 查看输入的图像宽高是否正确；

3.设置贴纸后没有效果

答：遇到这个问题可以采取一下步骤进行排查
(1) 查看是否有error log 输出，如果有error log可查看对象 error code来进一步错误信息；
(2) 查看是否正确的检测到人脸，可以输出一下当前的face_count；

4.生成纹理是报错-6683/-6661

答：遇到这个问题可以采取一下步骤进行排查
这两个错误是都是由如下方法报出

```
CVOpenGLESTextureCacheCreateTextureFromImage(kCFAllocatorDefault,
                                             _cvTextureCache,
                                             *pixelBufferOut,
                                             NULL,
                                             GL_TEXTURE_2D,
                                             GL_RGBA,
                                             width,
                                             height,
                                             GL_BGRA,
                                             GL_UNSIGNED_BYTE,
                                             0,
                                             cvTexture);
```

(1) 查看创建纹理时输入的纹理宽高(width,height)是否正确
(2) 查看创建纹理时纹理缓存时候创建(_cvTextrueCache)

5.调用 texture_process 接口后黑屏

答：遇到这个问题可以采取一下步骤进行排查
(1) 在图像处理时，需要使用OpenGL API，在使用OpenGLAPI时，需要注意的问题时我们要保证当前OpenGL在统一哥OpenGLContext中，因此在使用texture_process、render接口前，要设置一下OpenGLContext，

```
if ([EAGLContext currentContext] != self.glContext) {
    [EAGLContext setCurrentContext:self.glContext];
}
```

(2) 查看输出纹理是否成功创建，可以通过

```
glIsTexture(outputTexture);
```

如何没有创建成功就要使用例如如下的代码两创建

```
- (BOOL)setupTextureWithPixelBuffer:(CVPixelBufferRef *)pixelBufferOut
                                  w:(int)iWidth
                                  h:(int)iHeight
                         glTexture:(GLuint *)glTexture
                         cvTexture:(CVOpenGLESTextureRef *)cvTexture {
    CFDictionaryRef empty = CFDictionaryCreate(kCFAllocatorDefault,
                                               NULL,
                                               NULL,
                                               0,
                                               &kCFTypeDictionaryKeyCallBacks,

&kCFTypeDictionaryValueCallBacks);

    CFMutableDictionaryRef attrs =
CFDictionaryCreateMutable(kCFAllocatorDefault,
                                                                  1,

&kCFTypeDictionaryKeyCallBacks,

&kCFTypeDictionaryValueCallBacks);

    CFDictionarySetValue(attrs, kCVPixelBufferIOSurfacePropertiesKey, empty);

    CVReturn cvRet = CVPixelBufferCreate(kCFAllocatorDefault,
                                         iWidth,
                                         iHeight,
                                         kCVPixelFormatType_32BGRA,
                                         attrs,
                                         pixelBufferOut);

    if (kCVReturnSuccess != cvRet) {
```

```
        NSLog(@"CVPixelBufferCreate %d" , cvRet);
    }

    cvRet = CVOpenGLESTextureCacheCreateTextureFromImage(kCFAllocatorDefault,
                                                         _cvTextureCache,
                                                         *pixelBufferOut,
                                                         NULL,
                                                         GL_TEXTURE_2D,
                                                         GL_RGBA,
                                                         self.imageWidth,
                                                         self.imageHeight,
                                                         GL_BGRA,
                                                         GL_UNSIGNED_BYTE,
                                                         0,
                                                         cvTexture);

    CFRelease(attrs);
    CFRelease(empty);

    if (kCVReturnSuccess != cvRet) {

        NSLog(@"CVOpenGLESTextureCacheCreateTextureFromImage %d" , cvRet);

        return NO;
    }

    *glTexture = CVOpenGLESTextureGetName(*cvTexture);
    glBindTexture(CVOpenGLESTextureGetTarget(*cvTexture), *glTexture);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
    glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
    glBindTexture(GL_TEXTURE_2D, 0);

    return YES;
}
```

6.如何获取数据格式

6.1 如果判断当前的数据格式

答： 对于iOS，我们可以设置两个数据格式 1.NV12(2个plane) 2.RGBA(1个plane)
因此可以通过通道数当前的数据格式:

```
int plane = CVPixelBufferGetPlaneCount(pixelBuffer);
if(plane > 1){
    //数据格式为NV12
}else{
    //数据格式为RGBA
}
```

6.2 如果获取图像的宽高

```
NV12:

int width = CVPixelBufferGetWidthOfPlane(pixelBuffer, 0);
int height = (int)CVPixelBufferGetHeightOfPlane(pixelBuffer, 0);

RGBA:
int width = (int)CVPixelBufferGetWidth(pixelBuffer);
int height = (int)CVPixelBufferGetHeight(pixelBuffer);
```

6.3 如果获取图像数据

```
NV12:

unsigned char* yImageData = (unsigned
char*)CVPixelBufferGetBaseAddressOfPlane(pixelBuffer, 0);
unsigned char* uvImageData = (unsigned
char*)CVPixelBufferGetBaseAddressOfPlane(pixelBuffer, 0);

RGBA:
unsigned char* pRGBAImageData = (unsigned
char*)CVPixelBufferGetBaseAddress(pixelBuffer);
```