RTC美颜接入

美颜美型接入说明

人脸识别接入接口

美颜接入接口

美颜接入流程

美颜美型接入说明

1.13版本RTC SDK只支持移动端标准版本美颜（不带人脸识别功能及针对处理），也就是利用opengl进行纹理的图像处理。

针对美颜参数调节，需要客户在对接第三方美颜算法时，调用第三方算法库接口调节美颜参数。RTC Sdk保障透传opengl的纹理数据和opengl的线程环境给到第三方美颜库进行处理。

1.14版本RTC SDK新增人脸识别的高级美颜， 利用第三方算法库对图像进行人脸识别， 将人脸结构化数据传入opengl管道中进行指定的图像处理。

人脸识别接入接口

人脸识别接入时，需要订阅采集之后的前处理buffer数据，所以在startPreview之后需要调用以下接口获取采集前处理数据并处理：

【iOS】：

```
///@brief 订阅采集视频前处理裸数据
///@param videoSource 订阅本地采集的哪一路流，移动端场景填写AliRtcVideosourceCameraLargeType
- (void)subscribeVideoPreprocessData:(AliRtcVideoSource)videoSource;
```

订阅成功后，通过以下delegate回调本地采集数据

```
///@brief RTC采集视频数据前处理回调
///@param type video source type
///@param videoFrame the video data frame
///@return 人脸识别结构体指针（第三方定义结构体），SDK只做传递
- (long)onVideoDetectCallback:(AliRtcVideoSource)type videoFrame:(AliRtcVideoDataSample *)videoFrame;
```

【Android】：

```
///@brief register preprocess observer
///@param observer 回调接口
void RegisterPreprocessVideoObserver(AliDetectObserver observer);
```

订阅成功后，将会回调AliDetectObserver的onData

```
///@brief 采集前处理回调接口
///@param dataFrameY Y分量指针
///@param dataFrameU U分量指针
///@param dataFrameV V分量指针，NV12和NV21该指针为null
///@param format， 图像数据格式
///@param width， 图像宽度
///@param height， 图像高度
///@param strideY， 图像Y分量stride
///@param strideU， 图像U分量stride
///@param strideV， 图像V分量stride
///@param rotate， 图像旋转角度
///@param extraData，附加字段（非定制化可忽略）
long onData(long dataFrameY, long dataFrameU, long dataFrameV, AliRTCImageFormat format, int width, int height, int strideY, int strideU, int strideV, int rotate, long extraData);
```

美颜接入接口

美颜接入时，需要订阅视频的纹理数据，所以在startPreview之后需要调用以下接口获取opengl的纹理数据和opengl的线程环境：

【iOS】：

```
///@brief 订阅opengl的纹理数据
///@param[in] uid 订阅的用户，通常本地需要美颜，填写""或者本地uid都可以
///@param[in] videoSource 订阅本地pub的哪一路流，移动端场景填写AliRtcVideosourceCameraLargeType
///@param[in] videoTextureType 美颜的场景选择AliRtcVideoTextureTypePre
- (void)subscribeVideoTexture:(NSString *)uid videoSource:(AliRtcVideoSource)videoSource videoTextureType:(AliRtcVideoTextureType)videoTextureType;
```

订阅成功之后，将会回调三个接口：

```
///@brief 表示本地视频流纹理创建
///@param[in] uid 订阅的用户，就是subscribeVideoTexture所填写的uid
///@param[in] videoTextureType，就是subscribeVideoTexture所填写的videoTextureType
///@param[in] context opengl的上下文EAGLContext指针
```

– (**void**)onVideoTextureCreated:(NSString *)uid videoTextureType:
(AliRtcVideoTextureType)videoTextureType context:(**void** *)context

///@**brief** 表示每一帧视频流处理
///@**param**[in] uid 订阅的用户，就是subscribeVideoTexture所填写的uid
///@**param**[in] videoTextureType，就是subscribeVideoTexture所填写的videoTextureType
///@**param**[in] textureId 视频流纹理的输入texture id
///@**param**[in] width 视频流纹理的宽度
///@**param**[in] height 视频流纹理的高度
///@**param**[in] extraData 检测的人脸数据， **带人脸识别时：【为onVideoDetectCallback返回的人脸结构数据指针】；不带人脸识别时：为0；**
///@**param**[out] **处理之后的texture id，如果不需要美颜，请把textureId输入的texture id传回。【非常重要】**
– (**int**)onVideoTexture:(NSString *)uid videoTextureType:
(AliRtcVideoTextureType)videoTextureType textureId:(**int**)textureId width:(**int**)width height:
(**int**)height extraData:(**long**)extraData

///@**brief** 表示本地视频流纹理销毁
///@**param**[in] uid 订阅的用户，就是subscribeVideoTexture所填写的uid
///@**param**[in] videoTextureType，就是subscribeVideoTexture所填写的videoTextureType
– (**void**)onVideoTextureDestory:(NSString *)uid videoTextureType:
(AliRtcVideoTextureType)videoTextureType

【Android】：
///@**brief** 订阅opengl的纹理数据
///@**param**[in] uid 订阅的用户，通常本地需要美颜，填写""或者本地uid都可以
///@**param**[in] observer 回调接口
void RegisterTexturePreObserver(String callId, AliTextureObserver observer);

订阅成功之后，将会在用户所传的observer中回调三个接口函数：

///@**brief** 表示本地视频流纹理创建
///@**param**[in] callId 订阅的用户，就是RegisterTexturePreObserver所填写的callId
///@**param**[in] context opengl的上下文EGLContext指针
void onTextureCreate(String callId, **long** context);

///@**brief** 表示每一帧视频流处理
///@**param**[in] callId 订阅的用户，就是RegisterTexturePreObserver所填写的callId
///@**param**[in] textureId 视频流纹理的输入texture id

///@param[in] width 视频流纹理的宽度
///@param[in] height 视频流纹理的高度
///@param[in] stride 视频流纹理的stride
///@param[in] rotate 视频流纹理的rotate角度
///@param[in] extraData 检测的人脸数据， **带人脸识别时：【为AliDetectObserver的OnData返回的人脸结构数据指针】；不带人脸识别时：为0；**
///@param[out] **处理之后的texture id，如果不需要美颜，请把textureId输入的texture id传回。【非常重要】**
int onTexture(String callId, **int** textureId, **int** width, **int** height, **int** stride, **int** rotate, **long** extraData);

///@brief 表示本地视频流纹理销毁
///@param[in] callId 订阅的用户，就是RegisterTexturePreObserver所填写的callId
void onTextureDestroy(String callId);

美颜接入流程

在接入前，我们需要**在SDK的实例extra字段中添加开关：**user_specified_video_preprocess：TRUE；

【iOS】

1. SDK的instance构造函数里面，扩展字段添加[extrasDic setValue:@"TRUE" forKey:@"user_specified_video_preprocess"];改extraDic经过序列化之后传入到AliRtcEngine sharedInstance的第二个字段extra中；
2. 在调用本地预览开启接口startPreview之后，调用subscribeVideoTexture订阅opengl纹理数据。通常是对本地用户进行美颜，uid填写为""字符串即可；
3. 【需要人脸识别和美型功能时候】：在调用本地预览开启接口startPreview之后，再调用subscribeVideoPreprocessData订阅采集前处理YUV数据。通常是对采集图像做人脸识别；
4. 【需要人脸识别和美型功能时候】：在onVideoDetectCallback回调中做第三方算法的人脸识别操作，返回的long为人脸识别之后的该图像的人脸结构体指针；
5. 在onVideoTextureCreated中做第三方算法的初始化工作，其中context为opengl的上下文；
6. 在onVideoTexture做第三方算法的每一帧美颜处理工作，**如果不需要美颜或者第三方算法处理不成功，请将输入的texture id return给该函数。**如果美颜处理成功，则返回第三方算法处理过的texture id；
7. 在onVideoTextureDestory做第三方算法的销毁工作。

【Android】

1. SDK的instance构造函数里面，扩展字段添加 jsonObject.addProperty("user_specified_video_preprocess", "TRUE");改jsonObject转化成string后传入到AliRtcEngine.*getInstance的第二个字段extra里面；*

2. 在调用本地预览开启接口startPreview之后，调用RegisterTexturePreObserver订阅opengl纹理数据。通常是对本地用户进行美颜，callid填写为""字符串即可；
3. 【需要人脸识别和美型功能时候】：在调用本地预览开启接口startPreview之后，再调用RegisterPreprocessVideoObserver订阅采集前处理YUV数据。通常是对采集图像做人脸识别；
4. 【需要人脸识别和美型功能时候】：在AliDetectObserver的onData回调中做第三方算法的人脸识别操作，返回的long为人脸识别之后的该图像的人脸结构体指针；
5. 在传入observer函数回调onTextureCreate中第三方算法的初始化工作，其中context为opengl的上下文；
6. 在传入observer函数回调onTexture做第三方算法的每一帧美颜处理工作，**如果不需要美颜或者第三方算法处理不成功，请将输入的texture id return给该函数。**如果美颜处理成功，则返回第三方算法处理过的texture id；
7. 在传入observer函数回调onTextureDestroy做第三方算法的销毁工作。