视频人像分割sdk

文档版本：1.0

1）授权相关API

1.1）wx_vseg_lic_set_key_and_file

通过文件方式，为SDK设置license
调用其他api前，必须先调用wx_vseg_lic_set_key_and_file或者wx_vseg_lic_set_key_and_data两者之一, license文件可以使用auth_tools下载或者向顺势兄弟科技有限公司索要， 一个进程只需要调用一次，不需要反复调用。

- 函数
  > int wx_vseg_lic_set_key_and_file(const char *appkey, const char *license_path)
- 参数：
  - [in] appkey: appkey
  - [in] license_path: license文件路径
- 返回值：
  - 成功 - 0
  - 失败 - 错误码

1.2）wx_vseg_lic_set_key_and_data

通过内存的方式，为SDK设置license
调用其他api前，必须先调用wx_vseg_lic_set_key_and_file或者wx_vseg_lic_set_key_and_data两者之一, license文件可以使用auth_tools下载或者向顺势兄弟科技有限公司索要， 一个进程只需要调用一次，不需要反复调用

- 函数：
  > int wx_vseg_lic_set_key_and_data(const char *appkey, void *buffer, size_t size);
- 参数：
  - [in] appkey: appkey，
  - [in] buffer: license数据，
  - [in] size: buffer参数长度，
- 返回值：
  - 成功 - 0
  - 失败 - 错误码

1.3）wx_vseg_lic_valid_from

获取当前license文件的有效期起始时间戳，单位为秒
需要在wx_vseg_lic_set_file或者wx_vseg_lic_set_data函数调用后使用，

- 函数：
  > int wx_vseg_lic_valid_from(time_t *valid_from);
- 参数：
  - [in] valid_from： license有效期起始时间戳
- 返回值：
  - 成功 - 0
  - 失败 - 错误码

1.4）wx_vseg_lic_expired_at

获取当前license文件的有效期过期时间戳，单位为秒
需要在wx_vseg_lic_set_file或者wx_vseg_lic_set_data函数调用后使用.

- 函数：
  > int wx_vseg_lic_expired_at(time_t *expired_at);
- 参数：
  - [in] expired_at license有效期过期时间戳
- 返回值：
  - 成功 - 0
  - 失败 - 错误码

1.5）wx_vseg_lic_license_id

获取当前license文件的licenseId
需要在wx_vseg_lic_set_file或者wx_vseg_lic_set_data函数调用后使用，

- 函数：
  > const char * wx_vseg_lic_license_id();
- 返回值：
  - 成功 - 0
  - 失败 - 错误码

1.6）wx_vseg_lic_device_id

获取当前设备的device_id
获取当前设备的device_id

- 函数：
  > const char * wx_vseg_lic_device_id();
- 返回值：
  - 成功 - device_id
  - 失败 - NULL

2）抠图相关API

2.1）wx_vseg_handle_t

抠图操作句柄类型

2.2）wx_img_format_e

图像格式枚举
多通道图像，数据均为HWC方式存储

```
typedef wx_img_format_e{
    WX_IMG_UNK = 0, // 未知格式
    WX_IMG_RGB24,   // 3字节RGB格式
    WX_IMG_BGR24,   // 3字节BGR格式
    WX_IMG_RGBA,    // 4直接RGBA格式
    WX_IMG_BGRA,    // 4直接BGRA格式
    WX_IMG_F32      // 单通道浮点格式
} wx_img_format_e;
```

2.3）wx_image_t

图像结构
抠图类API，图像的输入和输出，均使用本结构

- 变量：
  - int width;
    宽
  - int height;
    高
  - wx_img_format_e format;
    格式，
  - int line_size;
    行宽（单位为：bytes），设置0的情况下，为width * element_size，
    比如，100x200的RGB24的图片，line_size为300,
    再比如，100x200的RGBA的图片，line_size为400。
  - void *data;
    数据

2.4）wx_vseg_handle_create

创建抠图操作句柄
该句柄在后续函数调用中作为输入，需要调用 wx_vseg_handle_release() 函数来释放， 如果一开始不确定图像尺寸，可以输入width和height可以输入0， 当输入0时，处理图像的宽高由第一次调用wx_vseg_push的输入frame的尺寸决定， 一旦图像的尺寸确定，后续的wx_vseg_push输入的图像，尺寸必须保持一致，不能变化

- 函数：

```
int wx_vseg_handle_create(
    int width,
    int height,
    bool async_mode,
    const char* config,
    wx_vseg_handle_t *handle);
```

- 参数：
  - [in] width： 目标图像的宽度，输入0时，由第一次wx_vseg_push调用指定宽高
  - [in] height： 目标图像的高度，输入0时，由第一次wx_vseg_push调用指定宽高
  - [in] async_mode： 是否是异步模式，false - 同步模式， true - 异步模式
  - [in] config： 保留参数，传入NULL
  - [out] handle： 创建的操作句柄
- 返回值：
  - 0：成功，
  - 其他：错误码

2.5）wx_vseg_sync

同步模式下，对某一帧进行抠图处理
本函数的handle，必须为以**同步模式**创建（函数wx_vseg_handle_create的async_mode参数为false），抠图结果是随着函数直接返回的

- 函数：

```
int wx_vseg_sync(
    wx_vseg_handle_t handle,
    const wx_image_t *frame,
    wx_image_t *alpha);
```

- 参数：
  - [in] handle： 句柄
  - [in] frame： 帧的原图，连续的图像输入，图像尺寸必须保持一致
  - [in/out] alpha： 抠图结果，WX_IMG_F32格式，[0-1.0]之间的二维矩阵，宽高和输入的frame一致，
    alpha.data的存储空间需要调用者分配，并且保证最小空间为frame.width * frame.height * 4
- 返回值：
  - 0：成功，
  - 其他：错误码

2.6）wx_vseg_async_push

异步模式下，传入一帧待抠图的图片
本函数的handle，必须为以异步模式创建（函数wx_vseg_handle_create的async_mode参

数为true） 抠图结果通过函数wx_vseg_async_pull()获取， 如果系统负载过高，可能会导致丢帧的情况，是否丢帧可以pts来对应

- 函数：

```
int wx_vseg_async_push(
    wx_vseg_handle_t handle,
    const wx_image_t *frame,
    uint64_t pts);
```

- 参数：
  - [in] handle： 异步句柄
  - [in] frame： 帧的原图，连续的图像输入，图像尺寸必须保持一致
  - [in] pts：用来标识某一帧，wx_vseg_async_pull会返回
- 返回值：
  - 0：成功
  - 其他：错误码

2.7）wx_vseg_async_pull

异步模式下，获取之前某一帧的抠图结果
本函数的handle，必须为以异步模式创建（函数wx_vseg_handle_create的async_mode参数为true），由于采用异步模式，所以本函数返回的结果，并不是最后一次wx_vseg_push的帧的结果，有可能是数帧之前的某帧的结果；另外，如果系统负载过高，可能会导致丢帧的情况， 是否丢帧可以pts来对应。

- 函数：

```
int wx_vseg_async_pull(
    wx_vseg_handle_t handle,
    int64_t wait_in_microsec,
    wx_image_t *alpha,
    uint64_t *pts);
```

- 参数：
  - [in] handle： 异步句柄
  - [in] wait_in_microsec： 等待时间（单位微秒：1/1000000秒）， 本参数可取以下值：
    - == 0 : 为不等待，不管有没有取到结果帧，都立即返回；
    - < 0 : 一直等待，直到获取到某一帧的抠图结果，或者出错；
    - > 0 : 获取到某一帧的抠图后返回，或者等待指定的时间后超时返回
  - [in/out] alpha 返回的抠图结果，WX_IMG_F32格式，0-1.0之间的浮点数二维矩阵，
    alpha.data的存储空间需要调用者分配，并且保证最小空间为frame.width * frame.height * 4
  - [out] pts：对应的结果帧的pts（wx_vseg_async_push函数的输入）
- 返回值：

- 0: 成功获取一帧；
- 1: 暂时没有结果帧可以返回，等待后再次获取；
- 2: 抠图结束，wx_vseg_handle_release被调用；
- 其他: 出错码

2.8）wx_vseg_handle_release

释放抠图句柄

- 函数：

```
void wx_vseg_handle_release(wx_vseg_handle_t handle);
```

- 参数：
  - [in] handle 输入背景图像
- 返回值：
  无

2.9）wx_vseg_blend

图像融合
根据传入的前景图、背景图、alpha，做图像融合，必须保证所有输入图具备相同的宽和高，

- 函数：

```
int wx_vseg_blend(const wx_image_t *foreground,
    const wx_image_t *background,
    const wx_image_t *alpha,
    wx_image_t *blend);
```

- 参数：
  - [in] foreground： 前景图，格式必须为RGB24/BGR24
  - [in] background： 背景图像，格式必须为RGB24/BGR24
  - [in] alpha： 输入alpha图，WX_IMG_F32格式，值为[0 - 1.0]之间
  - [in/out] blend： 融合后的图像,
    blend.data的存储空间需要调用者分配，并且保证最小空间为
    foreground.width * foreground.height * 3
- 返回值：
  - 0：成功
  - 其他：错误码

2.10）wx_vseg_change_background_sync

抠图并换背景
同步模式下对某一帧进行抠图并换背景操作，并输出新背景的合成图。
本函数的handle，必须为以同步模式创建（函数wx_vseg_handle_create的async_mode参

数为false）；
必须保证frame，background和blend图尺寸相同；
免费版和付费版均可调用本函数，但是对于免费版，只能输出低于25万像素的合成图。

- 函数：

```
int wx_vseg_change_background_sync(wx_vseg_handle_t handle,
                                   const wx_image_t *frame,
                                   const wx_image_t *background,
                                   wx_image_t *blend);
```

- 参数：
  - [in] handle： 句柄
  - [in] frame： 帧的原图，格式必须为RGB24/BGR24，连续的图像输入，图像尺寸必须保持一致
  - [in] background： 背景图像，格式必须为RGB24/BGR24
  - [in/out] blend： 融合后的图像,
    blend.data的存储空间需要调用者分配，并且保证最小空间为
    foreground.width * foreground.height * 3
- 返回值：
  - 0：成功
  - 其他：错误码

3）示例代码

3.1）同步方式抠图

```
// 以下代码需要opencv的支持
#include <opencv2/opencv.hpp>
#include "wx_vseg.h"

int main(int argc, char *argv[])
{
    wx_vseg_handle_t handle = nullptr;
    int res;

    // 设置appkey和license文件，两者必须匹配
    if ((res = wx_vseg_lic_set_key_and_file("your_app_key", "./my.lic")) != 0) {
        const char* errmsg = wx_vseg_get_last_error_msg();
        printf("set license fail: %d, %s\n", res, errmsg);
        return -1;
    }

    // 创建抠图handle
    res = wx_vseg_handle_create(0, 0, 0, nullptr, &handle);
    if (res != 0) {
        printf("wx_vseg_handle_create fail: %d\n", res);
        return -1;
    }
```

```
    // 使用opencv打开摄像头
    cv::VideoCapture cap(0);
    cv::Mat frame, alpha, bg, blend;
    int key = 0;
    while(key != 'q') {
        // 从摄像头读取一帧
        if (cap.read(frame)) {
            wx_image_t img_frame = {frame.cols,
                                    frame.rows,
                                    WX_IMG_BGR24,
                                    (int)frame.step[0],
                                    frame.data};
            if (alpha.empty()) {
                alpha.create(frame.rows, frame.cols, CV_32FC1);
            }
            wx_image_t img_alpha;
            // img_alpha作为输出参数，只需要分配并设置data变量即可
            img_alpha.data = alpha.data;

            // 对本帧进行抠图，返回alpha
            res = wx_vseg_sync(handle, &img_frame, &img_alpha);
            if (res != 0) {
                printf("wx_vseg_sync fail: %d\n", res);
            }
            else {
                if (blend.empty()) {
                    blend.create(frame.rows, frame.cols, CV_8UC3);
                }
                if (bg.empty()) {
                    bg.create(frame.rows, frame.cols, CV_8UC3);
                    bg = cv::Scalar(30, 150, 30); // 设置为绿色背景
                }
                wx_image_t img_blend;
                // img_blend作为输出参数，只需要分配并设置data变量即可
                img_blend.data = blend.data;
                wx_image_t img_bg = {bg.cols,
                                     bg.rows,
                                     WX_IMG_BGR24,
                                     (int)bg.step[0],
                                     bg.data};
                // 图像融合
                res = wx_vseg_blend(&img_frame, &img_bg, &img_alpha,
&img_blend);
                if (res != 0) {
                    printf("failed in wx_vseg_blend, res: %d\n", res);
                }
                else {
                    cv::imshow("blend", blend);
                }
            }
        }
        key = cv::waitKey(30);
    }
    return 0;
}
```