ST_MOBILE

人脸关键点检测跟踪说明文档（106点）

SENSETIME

目录

简介

st_mobile 人脸关键点（106 点）检测跟踪简介

st_mobile SDK 是 SenseTime 针对移动端优化后的开发包系列。其中包括 106 关键点检测跟踪，美颜，滤镜，贴纸，换脸，属性等功能。本开发包系列针对移动端直播，视频处理等常见场景，对算法实时性，CPU 占用率等性能指标做了优化。106 关键点检测跟踪是 st_mobile 的一个功能模块，下面是对 106 关键点检测跟踪的介绍。

人脸关键点检测是指在检测到的人脸框中，进一步定位人脸的五官和轮廓位置。在图片给定的人脸框中对关键点的坐标进行迭代收敛，返回迭代的结果。若框中没有人脸，依然会返回结果。

人脸关键点跟踪是在监测的基础上，加入面部轨迹追踪与视频动态解析技术，极大地提高了人脸的检测速度，能够随着视频内容的变化迅速定位人脸所在的位置。SenseTime人脸关键点跟踪技术，采用了最新的基于级联回归的算法，结合了深度学习的五官标定初始化，综合多个不同标准的多点数据集知识，使得同一个模型可以应用于不同数量的关键点跟踪，以保障更低的误差和更好的适应性。我们的人脸关键点定位算法可对各种表情、姿态、角度丰富多变的人脸进行精准关键点定位。

人脸关键点（106 点）检测跟踪 SDK 包含的主要功能有：

1. 静态图片的人脸关键点检测功能
2. 人脸 106 点关键点的实时检测和跟踪
3. 支持最多 32 个人脸的多人脸实时检测和跟踪
4. 支持实时人脸的三维旋转角度输出

st_mobile V2.0 人脸关键点（106 点）检测跟踪版本说明

new feature

- 接口变动，所有 cv 变为了 st

- 加入静态图片检测功能，包含的接口增加如下：
  - st_mobile_face_detection_create
  - st_mobile_face_detection_detect
  - st_mobile_face_detection_release_result
  - st_mobile_face_detection_destroy
- 多增加了六种颜色转换格式
  - ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_BGR888 转换
  - ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_BGRA8888 转换
  - ST_PIX_FMT_NV12 到 ST_PIX_FMT_RGBA8888 转换
  - ST_PIX_FMT_NV12 到 ST_PIX_FMT_RGB888 转换
  - ST_PIX_FMT_RGBA8888 到 ST_PIX_FMT_NV12 转换
  - ST_PIX_FMT_RGB888 到 ST_PIX_FMT_NV12 转换
- Track 的 config 增加了四个。
  - #define ST_MOBILE_TRACKING_RESIZE_IMG_320W 0x00000001
  - #define ST_MOBILE_TRACKING_RESIZE_IMG_640W 0x00000002
  - #define ST_MOBILE_TRACKING_RESIZE_IMG_1280W 0x00000004
  - #define ST_MOBILE_TRACKING_DEFAULT_CONFIG ST_MOBILE_TRACKING_MULTI_THREAD|ST_MOBILE_TRACKING_RESIZE_IMG_320W

注：若选择单线程追踪像 V1.0 一样的性能，需将 config 设置成

ST_MOBILE_TRACKING_SINGLE_THREAD|ST_MOBILE_TRACKING_RESIZE_IMG_320W

若选择双线程追踪则将 config 设为

平台支持说明

SDK 的平台支持请下表格：

Android 支持

CPU	ARM V7 或以上，支持 NEON 指令集
系统版本	Android 4.0 or later

iOS 支持

机型	iPhone 4S or later
系统版本	iOS 7.0 or later

注：iOS 现已支持模拟器模式（需要用到摄像头的功能在模拟器下无法测试）

iOS 编译架构包括：armv7, arm64, i386, x86_64。

性能测试

我们对 st_mobile V2.0 做了性能测试，下面列举了部分机型的测试结果。对于人脸关键点检测，在有人脸时测试的图片尺寸是 596*800，无人脸时测试的图片尺寸为 309*220。测试结果如下。

人脸关键点（106 点）检测测试结果

iOS 部分机型测试结果

机型	有人脸算法执行时间	无人脸算法执行时间
iPhone 4S(7.1.2)	155ms	85ms

iPhone 5C(9.0.1)	63ms	35ms
iPhone 5S(8.3)	27ms	14ms
iPhone 6(9.3.2)	22ms	12ms
iPhone 6 plus(9.0)	22ms	11ms
iPhone 6S(9.1)	12ms	7ms

Android 部分机型测试结果

机型	有人脸算法执行时间	无人脸算法执行时间
小米 4(4.4.4)	105ms	60ms
S5(5.0)	82ms	50ms
note4(5.1.1)	55ms	26ms
nexus6(6.0.1)	65ms	33ms
S4(4.4.2)	101ms	44ms
nexus5X (N)	62ms	35ms
nexus6P(6.0.1)	63ms	36ms
小米 5(6.0)	51ms	33ms
nexus5(6.0.1)	52ms	24ms
小米 3W(4.4.2)	93ms	46ms
红米 2(4.4.4)	147ms	80ms
酷派 S6(4.3)	129ms	64ms
MX4(4.4.2)	122ms	63ms
vivo X5 Pro(5.0)	84ms	53ms

OPPO R9(5.1)	62ms	32ms
MX5(5.1)	77ms	44ms
华为畅玩 5(5.1)	80ms	45ms
魅蓝 note2(5.1)	83ms	45ms
华为 mate7(4.4.2)	157ms	78ms
华为 P8(5.0.1)	93ms	47ms
华为 mate8(6.0)	85ms	47ms
华为 mate1(4.1.2)	154ms	79ms
S6(5.1.1)	89ms	64ms

人脸关键点（106 点）跟踪测试结果

iOS 部分机型测试结果

iPhone 6 Plus(9.0)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	29-31	13.5-16.4	2-9
无人脸	18-20	12.1-14.7	0-1

iPhone 6S(9.1)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	18-20	16.4-18	1-7
无人脸	11-14	14-15.7	0-1

iPhone6(9.3.2)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	31-32	13.6-16.9	2-11
无人脸	17-22	11.7-14.7	0-1

iPhone5s(8.4)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	29-31	13.3-15.5	3-11
无人脸	14-16	10.7-13.2	0-1

iPhone5c(7.0)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	38-39	11.1-12.6	7-12
无人脸	18-19	8.6-11.3	0-2

iPhone 4S (7.1.2)

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	78-97	8.1-11.2	15-23
无人脸	42-43	6.6-8.7	0-2

Android 部分机型测试结果

VIVO X5 Pro

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）

有人脸	31-36	28-30	8-11
无人脸	31-32	28-31	0-1

三星 S5

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	34-43	32-49	7-13
无人脸	31-37	32-48	0-2

OPPO R9

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	23-25	25-28	8-15
无人脸	20-26	22-28	0-3

小米 4

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	27-34	25-28	7-23
无人脸	29-35	23-26	0-3

三星 Note4

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	23-27	22-27	6-14
无人脸	25-27	21-25	0-3

Nexus6

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	24-27	23-34	7-15

无人脸	25-28	19-27	0-2

三星 S4

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	28-51	23-24	14-40
无人脸	25-57	21-22	1-6

Nexus5X

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	32-34	21-26	13-16
无人脸	32-35	23-28	0-2

Nexus6P

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	15-19	27-36	12-15
无人脸	16-17	24-32	0-1

Nexus5

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	28-32	21-29	8-20
无人脸	27-32	21-26	0-4

小米 5

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	31-35	27-37	4-5
无人脸	30-32	26-35	0-1

小米 3W

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	27-32	23-27	8-18
无人脸	24-32	23-25	0-3

酷派 S6

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	30-32	25-27	27-50
无人脸	28-31	23-25	0-6

MX4

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	21-27	52-56	15-28
无人脸	22-27	52-54	0-3

MX5

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	29-34	22-26	8-14
无人脸	30-35	20-23	0-2

华为畅玩 5

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	26-30	24-27	10-22
无人脸	19-29	18-26	0-2

魅蓝 note2

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	23-32	32-38	17-26
无人脸	31-38	28-32	0-2

华为 mate7

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	17-18	18-21	8-24
无人脸	19-20	18-20	0-2

华为 P8

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	17-20	25-28	10-14
无人脸	18-21	24-27	0-2

华为 mate8

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	14-15	29-33	4-6
无人脸	14-16	24-29	0-1

华为 mate1

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）
有人脸	32-43	22-24	27-45
无人脸	31-43	18-19	2-9

三星 S6

测试项	CPU 占有率（%）	内存（MB）	算法处理时长（ms）

有人脸	17-19	27-39	6-10
无人脸	18-20	23-33	0-2

开发使用说明

使用说明

106 点每个序号的位置定义如图 1 所示。106 点可以获取细微的面部结构信息，用于复杂的交互场景，并进行美容修改或趣味贴图。

图 1： 106 关键点位置定义

以下为 106 关键点数集合与位置对应。

脸框	0-32	鼻梁	43-46
左眉毛	33-37, 64-67	右眉毛	38-42, 68-71
左眼眶	52-57	右眼眶	58-63
左眼瞳孔	72-74, 104	右眼瞳孔	75-77, 105
鼻子下沿	47-51	鼻子外侧	78-83
上嘴唇	84-90, 96-100	下嘴唇	91-95, 101-103

通用类型定义

名称：st_handle_t

功能：用于保存函数及数据句柄，需调用其对应的销毁函数进行内存释放和回收。

声明：

```
typedef void *st_handle_t;
```

名称：st_result_t

功能：函数返回的错误代码类型

声明：

```
typedef int      st_result_t;

#define ST_OK (0)

#define ST_E_INVALIDARG (-1)

#define ST_E_HANDLE (-2)

#define ST_E_OUTOFMEMORY (-3)
```

#define ST_E_FAIL (-4)

#define ST_E_DELNOTFOUND (-5)

#define ST_E_INVALID_PIXEL_FORMAT (-6)

#define ST_E_FILE_NOT_FOUND (-10)

#define ST_E_INVALID_FILE_FORMAT (-11)

#define ST_E_INVALID_APPID (-12)

#define ST_E_INVALID_AUTH (-13)

#define ST_E_AUTH_EXPIRE (-14)

#define ST_E_FILE_EXPIRE (-15)

#define ST_E_DONGLE_EXPIRE (-16)

#define ST_E_ONLINE_AUTH_FAIL (-17)

#define ST_E_ONLINE_AUTH_TIMEOUT (-18)

参数：

ST_OK (0)：正常运行

ST_E_INVALIDARG (-1)：无效参数

ST_E_HANDLE (-2)：句柄错误

ST_E_OUTOFMEMORY (-3)： 内存不足

ST_E_FAIL (-4) ：内部错误

ST_E_DELNOTFOUND (-5)：定义缺失

ST_E_INVALID_PIXEL_FORMAT (-6)：不支持的图像格式

ST_E_FILE_NOT_FOUND (-10)：模型文件不存在

ST_E_INVALID_FILE_FORMAT (-11)：模型格式不正确，导致加载失败

ST_E_INVALID_APPID (-12)：包名错误

ST_E_INVALID_AUTH (-13)：加密狗功能不支持

ST_E_AUTH_EXPIRE (-14)：SDK 过期

ST_E_FILE_EXPIRE (-15)：模型文件过期

ST_E_DONGLE_EXPIRE (-16)：加密狗过期

ST_E_ONLINE_AUTH_FAIL (-17)：在线验证失败

ST_E_ONLINE_AUTH_TIMEOUT (-18)：在线验证超时

名称：st_rect_t

功能：矩阵类型，用于表示人脸框坐标

声明：

```
typedef struct st_rect_t {
int left;
int top;
int right;
int bottom;
} st_rect_t;
```

参数：

Left：矩形最左边的坐标

top：矩形最上边的坐标

right：矩形最右边的坐标，矩形宽度为 left-right

bottom：矩形最下边的坐标

名称：st_pointf_t

功能：浮点坐标类型，用于表示一个二维平面上的坐标，坐标类型为浮点数，单位为像素。

声明：

```
typedef struct st_pointf_t {
float x;
float y;
} st_pointf_t;
```

参数：

x：点的水平方向坐标，为浮点数

y：点的竖直方向坐标，为浮点数

名称：st_pointi_t

功能：浮点坐标类型，用于表示一个二维平面上的坐标，坐标类型为浮点数，单位为像素。

声明：

```
typedef struct st_pointi_t {
int x;
int y;
} st_pointi_t;
```

参数：

x：点的水平方向坐标，为整数

y：点的竖直方向坐标，为整数

名称：st_pixel_format

功能：图片格式的枚举型

声明：

typedef enum {

ST_PIX_FMT_GRAY8

ST_PIX_FMT_YUV420P

ST_PIX_FMT_NV12

ST_PIX_FMT_NV21

ST_PIX_FMT_BGRA8888

ST_PIX_FMT_BGR888

}st_pixel_format;

参数：

ST_PIX_FMT_GRAY8：Y 1 8bpp (单通道 8bit 灰度像素)

ST_PIX_FMT_YUV420P：YUV 4:2:0 12bpp (3 通道，一个亮度通道，另两个为 U 分量和 V 分量通道，所有通道都是连续的)

ST_PIX_FMT_NV12：YUV 4:2:0 12bpp (2 通道，一个通道是连续的亮度通道，另一通道为 UV 分量交错)

ST_PIX_FMT_NV21：YUV 4:2:0 12bpp (2 通道，一个通道是连续的亮度通道，另一通道为 VU 分量交错)

ST_PIX_FMT_BGRA8888：BGRA 8:8:8:8 32bpp (4 通道 32bit BGRA 像素)

ST_PIX_FMT_BGR888：BGR 8:8:8 24bpp (3 通道 24bit BGR 像素)

名称：st_rotate_type

功能：图像旋转说明

声明：

ST_CLOCKWISE_ROTATE_0 = 0,

ST_CLOCKWISE_ROTATE_90 = 1,

ST_CLOCKWISE_ROTATE_180 = 2,

ST_CLOCKWISE_ROTATE_270 = 3

参数：

ST_CLOCKWISE_ROTATE_0 = 0：图像不需要转向

ST_CLOCKWISE_ROTATE_90 = 1：图像需要顺时针旋转 90 度

ST_CLOCKWISE_ROTATE_180 = 2：图像需要顺时针旋转 180 度

ST_CLOCKWISE_ROTATE_270 = 3：图像需要顺时针旋转 270 度

名称：st_mobile_106_t

功能：人脸信息结构体

声明：

```
typedef struct st_mobile_106_t {
    st_rect_t rect;
    float score;
    st_pointf_t points_array[106];
    int yaw;
```

```
    int pitch;

    int roll;

    int eye_dist;

    int ID;

} st_mobile_106_t;
```

参数：

Rect：代表面部的矩形区域

Score：置信度

points_array[106]：人脸 106 关键点的数组

yaw：人脸的 pose 信息，水平转角，真实度量的左负右正，范围[-90-90]。

pitch：人脸的 pose 信息，俯仰角，真实度量的上负下正，范围[-90-90]。

roll：人脸的 pose 信息，旋转角，真实度量的左负右正，范围[-180-180]。

eye_dist：人脸的 pose 信息，两眼间距

ID：faceID

人脸关键点（106 点）跟踪

类型定义

名称：配置选项宏定义

功能：tracking 配置选项，对应 st_mobile_tracker_106_create 中的 config 参数，具体配置如下。

声明：

```
#define ST_MOBILE_TRACKING_MULIT_THREAD 0x00000000

#define ST_MOBILE_TRACKING_SINGLE_THREAD 0x00010000
```

#define ST_MOBILE_TRACKING_RESIZE_IMG_320W 0x00000001

#define ST_MOBILE_TRACKING_RESIZE_IMG_640W 0x00000002

#define ST_MOBILE_TRACKING_RESIZE_IMG_1280W 0x00000004

#define ST_MOBILE_TRACKING_DEFAULT_CONFIG MOBILE_TRACKING_MULIT_THREAD

|ST_MOBILE_TRACKING_RESIZE_IMG_320W

参数：

ST_MOBILE_TRACKING_MULIT_THREAD：多线程，功耗较多，卡顿较少

ST_MOBILE_TRACKING_SINGLE_THREAD：tracking 使用单线程的配置方式，功耗较少，对于性能弱的手机，会偶尔有卡顿现象

ST_MOBILE_TRACKING_RESIZE_IMG_320W：选择是否将图像缩小后进行 track，最后再将结果处理为源图像对应的结果。如果都不选择，直接处理原图。缩小后可提高处理速度。此参数是将图像 resize 为长边 320 的图像之后再检测，结果处理为原图像对应结果

ST_MOBILE_TRACKING_RESIZE_IMG_640W：resize 图像为长边 640 的图像之后再检测，结果处理为原图像对应结果

ST_MOBILE_TRACKING_RESIZE_IMG_1280W：resize 图像为长边 1280 的图像之后再检测，结果处理为原图像对应结果

ST_MOBILE_TRACKING_DEFAULT_CONFIG MOBILE_TRACKING_MULIT_THREAD

|ST_MOBILE_TRACKING_RESIZE_IMG_320W：默认 tracking 配置，使用多线程 +320W,可最大限度的提高速度，并减少卡顿

名称：st_color_convert_type

功能：支持颜色转换格式

声明：

typedef enum {

ST_BGRA_YUV420P = 0,

ST_BGR_YUV420P = 1,

ST_BGRA_NV12 = 2,

ST_BGR_NV12 = 3,

ST_BGRA_NV21 = 4,

ST_BGR_NV21 = 5,

ST_YUV420P_BGRA = 6,

ST_YUV420P_BGR = 7,

ST_NV12_BGRA = 8,

ST_NV12_BGR = 9,

ST_NV21_BGRA = 10,

ST_NV21_BGR = 11,

ST_BGRA_GRAY = 12,

ST_BGR_BGRA = 13,

ST_BGRA_BGR = 14,

ST_YUV420P_GRAY = 15,

ST_NV12_GRAY = 16,

ST_NV21_GRAY = 17,

ST_BGR_GRAY = 18,

ST_GRAY_YUV420P = 19,

ST_GRAY_NV12 = 20,

ST_GRAY_NV21 = 21,

ST_NV21_RGBA = 22,

ST_BGR_RGBA = 23,

ST_BGRA_RGBA = 24,

```
    ST_RGBA_BGRA = 25,
    ST_BGRA_RGBA = 26,
    ST_RGBA_BGRA = 27,
    ST_GRAY_BGR = 28,
    ST_GRAY_BGRA = 29
    ST_NV12_RGBA = 30
    ST_NV12_RGB = 31
    ST_RGBA_NV12 = 32
    ST_RGB_NV12 = 33
} st_color_convert_type;
```

参数：

ST_BGRA_YUV420P = 0：ST_PIX_FMT_BGRA8888 到 ST_PIX_FMT_YUV420P 转换

ST_BGR_YUV420P = 1：ST_PIX_FMT_BGR888 到 ST_PIX_FMT_YUV420P 转换

ST_BGRA_NV12 = 2：ST_PIX_FMT_BGRA8888 到 ST_PIX_FMT_NV12 转换

ST_BGR_NV12 = 3：ST_PIX_FMT_BGR888 到 ST_PIX_FMT_NV12 转换

ST_BGRA_NV21 = 4：ST_PIX_FMT_BGRA8888 到 ST_PIX_FMT_NV21 转换

ST_BGR_NV21 = 5： ST_PIX_FMT_BGR888 到 ST_PIX_FMT_NV21 转换

ST_YUV420P_BGRA = 6：ST_PIX_FMT_YUV420P 到 ST_PIX_FMT_BGRA8888 转换

ST_YUV420P_BGR = 7：ST_PIX_FMT_YUV420P 到 ST_PIX_FMT_BGR888 转换

ST_NV12_BGRA = 8： ST_PIX_FMT_NV12 到 ST_PIX_FMT_BGRA8888 转换

ST_NV12_BGR = 9：V_PIX_FMT_NV12 到 ST_PIX_FMT_BGR888 转换

ST_NV21_BGRA = 10：ST_PIX_FMT_NV21 到 ST_PIX_FMT_BGRA8888 转换

ST_NV21_BGR = 11：ST_PIX_FMT_NV21 到 ST_PIX_FMT_BGR888 转换

ST_BGRA_GRAY = 12：ST_PIX_FMT_BGRA8888 到 ST_PIX_FMT_GRAY8 转换

ST_BGR_BGRA = 13：ST_PIX_FMT_BGR888 到 ST_PIX_FMT_BGRA8888 转换

ST_BGRA_BGR = 14：ST_PIX_FMT_BGRA8888 到 ST_PIX_FMT_BGR888 转换

ST_YUV420P_GRAY = 15：ST_PIX_FMT_YUV420P 到 ST_PIX_FMT_GRAY8 转换

ST_NV12_GRAY = 16：ST_PIX_FMT_NV12 到 ST_PIX_FMT_GRAY8 转换

ST_NV21_GRAY = 17：ST_PIX_FMT_NV21 到 ST_PIX_FMT_GRAY8 转换

ST_BGR_GRAY = 18：ST_PIX_FMT_BGR888 到 ST_PIX_FMT_GRAY8 转换

ST_GRAY_YUV420P = 19： ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_YUV420P 转换

ST_GRAY_NV12 = 20：ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_NV12 转换

ST_GRAY_NV21 = 21：ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_NV21 转换

ST_NV21_RGBA = 22：NV21 到 RGBA 转换

ST_BGR_RGBA = 23： BGR 到 RGBA 转换

ST_BGRA_RGBA = 24： BGRA 到 RGBA 转换

ST_RGBA_BGRA = 25：RGBA 到 BGRA 转换

ST_BGRA_RGBA = 26: ST_PIX_FMT_BGRA 到 ST_PIX_FMT_RGBA 转换

ST_RGBA_BGRA = 27:ST_PIX_FMT_RGBA 到 ST_PIX_FMT_BGRA 转换

ST_GRAY_BGR = 28:ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_BGR888 转换

ST_GRAY_BGRA = 29:ST_PIX_FMT_GRAY8 到 ST_PIX_FMT_BGRA8888 转换

ST_NV12_RGBA = 30：ST_PIX_FMT_NV12 到 ST_PIX_FMT_RGBA8888 转换

ST_NV12_RGB = 31：ST_PIX_FMT_NV12 到 ST_PIX_FMT_RGB888 转换

ST_RGBA_NV12 = 32：ST_PIX_FMT_RGBA8888 到 ST_PIX_FMT_NV12 转换

ST_RGB_NV12 = 33：ST_PIX_FMT_RGB888 到 ST_PIX_FMT_NV12 转换

函数及功能说明

名称：st_mobile_tracker_106_create

功能：创建实时人脸 106 关键点跟踪句柄

声明：

```
ST_SDK_API st_result_t
st_mobile_tracker_106_create(
    const char* model_path,
    unsigned int config,
    st_handle_t* handle
);
```

参数：

[in] model_path：模型文件的绝对路径或相对路径，若不指定模型可为 NULL；模型中包含 detect+align+pose 模型

[in] config 配置选项，例如 ST_MOBILE_TRACKING_DEFAULT_CONFIG，默认使用双线程跟踪+RESIZE320W，可选择使用单线程（ST_MOBILE_TRACKING_SINGLE_THREAD | ST_MOBILE_RESIZE_IMG_320W），实时视频预览建议使用双线程，图片或视频后处理建议使用单线程

[out] Handle：人脸跟踪句柄，失败返回 NULL

成功返回 ST_OK，失败返回其他错误信息，错误码定义在 st_common.h 中，如 ST_E_FAIL 等

名称：st_mobile_tracker_106_set_facelimit

功能： 设置检测到的最大人脸数目 max_facecount， 持续 track 已检测到的 max_facecount 个人脸直到人脸数小于 max_facecount 再继续做 detect。

声明：

```
ST_SDK_API

st_result_t st_mobile_tracker_106_set_facelimit(

st_handle_t handle,

int max_facecount

);
```

参数：

[in] handle:已初始化的关键点跟踪句柄

[in] max_facecount:设置为 1 即是单脸跟踪，有效范围为[1, 32]

成功返回 ST_OK，错误则返回错误码，错误码定义在 st_common.h 中，如 ST_E_FAIL 等

名称：st_mobile_tracker_106_set_detect_interval

功能： 设置 tracker 每多少帧进行一次 detect

声明：

```
ST_SDK_API

st_result_t st_mobile_tracker_106_set_detect_interval(

st_handle_t handle,

int val

        );
```

参数：

[in] handle:已初始化的关键点跟踪句柄

[in]Val：有效范围[1, -)

成功返回 ST_OK，错误则返回错误码，错误码定义在 st_common.h 中，如 ST_E_FAIL 等

名称：st_mobile_tracker_106_reset

功能：重置实时人脸 106 关键点跟踪

声明：

```
ST_SDK_API st_result_t
st_mobile_tracker_106_reset(
    st_handle_t handle
);
```

参数：

Handle：已初始化的实时目标人脸 106 关键点跟踪句柄

名称：st_mobile_tracker_106_track

功能：对连续视频帧进行实时快速人脸 106 关键点跟踪

声明：

```
ST_SDK_API st_result_t
st_mobile_tracker_106_track(
    st_handle_t handle,
```

```
    const unsigned char *image,
    st_pixel_format pixel_format,
    int image_width,
    int image_height,
    int image_stride,
    st_mobile_rotate orientation,
    st_mobile_106_t **p_faces_array,
    int *p_faces_count
);
```

参数：

Handle：已初始化的实时人脸跟踪句柄

image：用于检测的图像数据

pixel_format：支持所有用于检测的图像数据的像素格式，如果对速度有要求，不推荐 BGRA 和 BGR 格式。

image_width：用于检测的图像的宽度(以像素为单位)

image_height：用于检测的图像的高度(以像素为单位)

[in]image_stride：图像跨度，以像素为单位，目前仅支持字节对齐的 padding，不支持 roi

orientation：视频图像中人脸的方向，顺时针旋转的角度

p_faces_array：检测到的人脸信息数组，api 负责分配内存，需要调用 st_mobile_tracker_106_release_result 释放

p_faces_count：检测到的人脸数量

成功返回 ST_OK，否则返回错误码，错误码定义在 st_common.h 中，如 ST_E_FAIL 等。

名称：st_mobile_tracker_106_release_result

功能：释放实时人脸 106 关键点跟踪返回结果时分配的空间

声明：

```
ST_SDK_API void
st_mobile_tracker_106_release_result(
    st_mobile_106_t *faces_array,
    int faces_count
);
```

参数：

[in]faces_array：跟踪到的人脸信息数组

[in]faces_count；跟踪到的人脸数量

名称：st_mobile_tracker_106_destroy

功能：销毁已初始化的 track106 句柄

声明：

```
ST_SDK_API void
st_mobile_tracker_106_destroy(
    st_handle_t handle
);
```

参数：

[in]Handle：已初始化的句柄

名称：st_mobile_color_convert

功能：进行颜色格式转换，不建议使用关于 YUV420P 的转换，速度较慢

声明：

```
ST_SDK_API st_result_t
st_mobile_color_convert(
const unsigned char *image_src,
unsigned char *image_dst,
int image_width,
int image_height,
st_ color_convert_type type
);
```

参数：

image_src：用于待转换的图像数据

image_dst：转换后的图像数据

image_width：用于转换的图像的宽度(以像素为单位)

image_height：用于转换的图像的高度(以像素为单位)

color_convert_type；需要转换的颜色格式

正常返回 ST_OK，否则返回错误类型

人脸关键点（106 点）检测

类型定义

名称：配置选项宏定义

功能：detect 配置开关，对应 st_mobile_face_detection_create 中的 config 参数。

声明：

```
#define ST_MOBILE_DETECT_DEFAULT_CONFIG          0x00000000

#define ST_MOBILE_DETECT_RESIZE_IMG_320W         0x00000001

#define ST_MOBILE_DETECT_RESIZE_IMG_640W         0x00000002

#define ST_MOBILE_DETECT_RESIZE_IMG_1280W        0x00000004
```

参数：

ST_MOBILE_DETECT_DEFAULT_CONFIG：默认 detect 配置，直接处理原图,可以最大限度的检测到相应人脸

ST_MOBILE_DETECT_RESIZE_IMG_320W：选择将图像缩小为长边 320 的图像之后再检测，最后再将结果处理为源图像对应结果。如果不选择，直接处理原图。缩小后可提高处理速度。resize 图像为长边 320 的图像之后再检测，结果处理为原图像对应结果

ST_MOBILE_DETECT_RESIZE_IMG_640W：resize 图像为长边 640 的图像之后再检测，结果处理为原图像对应结果

ST_MOBILE_DETECT_RESIZE_IMG_1280W：resize 图像为长边 1280 的图像之后再检测，结果处理为原图像对应结果

函数及功能说明

名称：st_mobile_face_detection_create

功能：创建人脸检测句柄

声明：

```
ST_SDK_API st_result_t
st_mobile_face_detection_create(
const char* model_path,
unsigned int config,
st_handle_t* handle
);
```

参数：

[in] model_path 模型文件的绝对路径或相对路径，例如 models/track.tar，可以与 track106 模型使用相同模型。模型内文件支持 detect; detect+align;detect+align+pose 三种模型

[in] config 配置选项，例如 ST_MOBILE_DETECT_DEFAULT_CONFIG

[out] handle 人脸检测句柄，失败返回 NULL

成功返回 ST_OK，失败返回其他错误信息

名称：st_mobile_face_detection_detect

功能：对图片进行人脸检测

声明：

```
ST_SDK_API st_result_t
st_mobile_face_detection_detect(
 st_handle_t handle,
 const unsigned char *image,
 st_pixel_format pixel_format,
 int image_width,
```

```
    int image_height,
    int image_stride,
    st_rotate_type orientation,
    st_mobile_106_t **p_faces_array,
    int *p_faces_count
);
```

参数：

[in] handle：已初始化的人脸检测句柄

[in] image：用于检测的图像数据

[in] pixel_format：用于检测的图像数据的像素格式，支持所有像素格式，如果对速度有要求，不推荐 BGRA 和 BGR 格式。

[in] image_width：用于检测的图像的宽度（以像素为单位)

[in] image_height：用于检测的图像的高度(以像素为单位)

[in] image_stride：用于检测的图像的跨度(以像素为单位)，即每行的字节数；目前仅支持字节对齐的 padding，不支持 roi

[in] orientation：图像中人脸的方向

[out] p_faces_array：检测到的人脸信息数组，api 负责分配内存，需要调用 st_mobile_face_detection_release_result 函数释放

[out]p_faces_count：检测到的人脸数量

成功返回 ST_OK，否则返回错误类型

名称：st_mobile_face_detection_release_result

功能：释放人脸检测结果

声明：

```
ST_SDK_API void
st_mobile_face_detection_release_result(
    st_mobile_106_t *faces_array,
    int faces_count
);
```

参数：

[in]faces_array：跟踪到到的人脸信息数组

[in]faces_count：跟踪到的人脸数量

名称：st_mobile_face_detection_destroy

功能：销毁已初始化的人脸检测句柄

声明：

```
ST_SDK_API void
st_mobile_face_detection_destroy(
st_handle_t handle
);
```

参数：

[in] handle：已初始化的句柄

功能使用说明

具体的功能使用方式，可以参考根目录下 sample 文件夹中的示例工程。