



## TDL SDK C 接口

©2025 北京晶视智能科技有限公司  
本文件所含信息归北京晶视智能科技有限公司所有。  
未经授权，严禁全部或部分复制或披露该等信息。

# 目录

|          |                     |           |
|----------|---------------------|-----------|
| <b>1</b> | <b>模型列表</b>         | <b>2</b>  |
| 1.1      | 检测类模型列表             | 2         |
| 1.2      | 人脸识别类模型列表           | 3         |
| 1.3      | 人脸属性和特征点类模型列表       | 3         |
| 1.4      | 图像分类模型列表            | 3         |
| 1.5      | 声音分类模型列表            | 4         |
| 1.6      | 关键点识别类模型列表          | 4         |
| 1.7      | 线识别类模型列表            | 4         |
| 1.8      | 车牌识别类模型列表           | 4         |
| 1.9      | 分割类模型列表             | 5         |
| 1.10     | 特征值提取类模型列表          | 5         |
| <b>2</b> | <b>结构体参考</b>        | <b>6</b>  |
| 2.1      | TDLDataTypeE        | 6         |
| 2.2      | TDLBox              | 7         |
| 2.3      | TDLFeature          | 7         |
| 2.4      | TDLPoints           | 8         |
| 2.5      | TDLLandmarkInfo     | 8         |
| 2.6      | TDLObjectInfo       | 9         |
| 2.7      | TDLObject           | 9         |
| 2.8      | TDLFaceInfo         | 10        |
| 2.9      | TDLFace             | 11        |
| 2.10     | TDLClassInfo        | 11        |
| 2.11     | TDLClass            | 12        |
| 2.12     | TDLKeypointInfo     | 12        |
| 2.13     | TDLKeypoint         | 13        |
| 2.14     | TDLSegmentation     | 13        |
| 2.15     | TDLInstanceSegInfo  | 14        |
| 2.16     | TDLInstanceSeg      | 14        |
| 2.17     | TDLLanePoint        | 15        |
| 2.18     | TDLLane             | 15        |
| 2.19     | TDLDepthLogits      | 16        |
| 2.20     | TDLTracker          | 16        |
| 2.21     | TDLocr              | 17        |
| <b>3</b> | <b>API 参考</b>       | <b>18</b> |
| 3.1      | 句柄                  | 18        |
| 3.2      | TDL_CreateHandle    | 18        |
| 3.3      | TDL_CreateHandleEx  | 19        |
| 3.4      | TDL_DestroyHandle   | 19        |
| 3.5      | TDL_DestroyHandleEx | 19        |

|      |   |    |
|------|---|----|
| 3.6  | TDL_WrapVPSSFrame . . . . .             | 20 |
| 3.7  | TDL_ReadImage . . . . .                 | 20 |
| 3.8  | TDL_ReadBin . . . . .                   | 20 |
| 3.9  | TDL_DestroyImage . . . . .              | 21 |
| 3.10 | TDL_OpenModel . . . . .                 | 21 |
| 3.11 | TDL_CloseModel . . . . .                | 21 |
| 3.12 | TDL_Detection . . . . .                 | 22 |
| 3.13 | TDL_FaceDetection . . . . .             | 22 |
| 3.14 | TDL_FaceAttribute . . . . .             | 23 |
| 3.15 | TDL_FaceLandmark . . . . .              | 23 |
| 3.16 | TDL_Classification . . . . .            | 24 |
| 3.17 | TDL_ObjectClassification . . . . .      | 24 |
| 3.18 | TDL_KeypointDetection . . . . .         | 25 |
| 3.19 | TDL_InstanceSegmentation . . . . .      | 25 |
| 3.20 | TDL_SemanticSegmentation . . . . .      | 26 |
| 3.21 | TDL_FeatureExtraction . . . . .         | 26 |
| 3.22 | TDL_LaneDetection . . . . .             | 27 |
| 3.23 | TDL_DepthStereo . . . . .               | 27 |
| 3.24 | TDL_Tracking . . . . .                  | 28 |
| 3.25 | TDL_CharacterRecognition . . . . .      | 28 |
| 3.26 | TDL_LoadModelConfig . . . . .           | 29 |
| 3.27 | TDL_SetModelDir . . . . .               | 29 |
| 3.28 | TDL_SetModelThreshold . . . . .         | 30 |
| 3.29 | TDL_IspClassification . . . . .         | 30 |
| 3.30 | TDL_Keypoint . . . . .                  | 31 |
| 3.31 | TDL_DetectionKeypoint . . . . .         | 31 |
| 3.32 | TDL_IntrusionDetection . . . . .        | 32 |
| 3.33 | TDL_MotionDetection . . . . .           | 32 |
| 3.34 | TDL_APP_Init . . . . .                  | 33 |
| 3.35 | TDL_APP_SetFrame . . . . .              | 33 |
| 3.36 | TDL_APP_Capture . . . . .               | 34 |
| 3.37 | TDL_APP_ObjectCounting . . . . .        | 35 |
| 3.38 | TDL_APP_ObjectCountingSetLine . . . . . | 35 |
| 3.39 | TDL_WrapImage . . . . .                 | 36 |
| 3.40 | TDL_LLMApiCall . . . . .                | 37 |

## 修订记录

| Date       | Description  | Owner |
|------------|--------------|-------|
| 2025/03/28 | 初稿           | 张思意   |
| 2025/03/31 | 添加 API 定义    | 刘俊斐   |
| 2025/04/02 | 添加模型列表和结构体定义 | 郑鑫烨   |
| 2025/08/29 | 更新模型列表和结构体定义 | 郑鑫烨   |

# 1 模型列表

## 1.1 检测类模型列表

| 模型名称                                   | 注释  |
|--|---|
| TDL_MODEL_MBv2_DET_PERSON              | 人体检测模型 (0:person)   |
| TDL_MODEL_YOLOV8N_DET_HAND             | 手部检测模型 (0:hand)   |
| TDL_MODEL_YOLOV8N_DET_PET_PERSON       | 宠物与人检测模型 (0: 猫, 1: 狗, 2: 人)                                   |
| TDL_MODEL_YOLOV8N_DET_PERSON_VEHICLE   | 人与车辆检测模型 (0: 车, 1: 公交, 2: 卡车, 3: 骑摩托车者, 4: 人, 5: 自行车, 6: 摩托车) |
| TDL_MODEL_YOLOV8N_DET_HAND_FACE_PERSON | 手、脸与人检测模型 (0: 手, 1: 脸, 2: 人)                                  |
| TDL_MODEL_YOLOV8N_DET_HEAD_PERSON      | 人头检测模型 (0: 人, 1: 头)   |
| TDL_MODEL_YOLOV8N_DET_HEAD_HARDHAT     | 头部与安全帽检测模型 (0: 头, 1: 安全帽)                                     |
| TDL_MODEL_YOLOV8N_DET_FIRE_SMOKE       | 火与烟检测模型 (0: 火, 1: 烟)  |
| TDL_MODEL_YOLOV8N_DET_FIRE             | 火检测模型 (0: 火)  |
| TDL_MODEL_YOLOV8N_DET_HEAD_SHOULDER    | 头肩检测模型 (0: 头肩)  |
| TDL_MODEL_YOLOV8N_DET_LICENSE_PLATE    | 车牌检测模型 (0: 车牌)  |
| TDL_MODEL_YOLOV8N_DET_TRAFFIC_LIGHT    | 交通信号灯检测模型 (0: 红, 1: 黄, 2: 绿, 3: 关闭, 4: 等待)                    |
| TDL_MODEL_YOLOV8N_DET_MONITOR_PERSON   | 人体检测模型 (0:person)   |
| TDL_MODEL_YOLOV5_DET_COCO80            | YOLOv5 COCO80 检测模型  |
| TDL_MODEL_YOLOV6_DET_COCO80            | YOLOv6 COCO80 检测模型  |
| TDL_MODEL_YOLOV7_DET_COCO80            | YOLOv7 COCO80 检测模型  |
| TDL_MODEL_YOLOV8_DET_COCO80            | YOLOv8 COCO80 检测模型  |
| TDL_MODEL_YOLOV10_DET_COCO80           | YOLOv10 COCO80 检测模型   |
| TDL_MODEL_PPYOLOE_DET_COCO80           | PPYOLOE COCO80 检测模型   |
| TDL_MODEL_YOLOX_DET_COCO80             | YOLOX COCO80 检测模型   |

## 1.2 人脸识别类模型列表

| 模型名称                         | 注释                         |
|------------------------------|----------------------------|
| TDL_MODEL_SCRFD_DET_FACE     | SCRFD 人脸检测模型 (0: 人脸 + 关键点) |
| TDL_MODEL_RETINA_DET_FACE    | RETINA 人脸检测模型              |
| TDL_MODEL_RETINA_DET_FACE_IR | RETINA 红外人脸检测模型            |

## 1.3 人脸属性和特征点类模型列表

| 模型名称   | 注释                        |
|--|---------------------------|
| TDL_MODEL_KEYPOINT_FACE_V2                           | 5 个关键点 + 模糊评分的人脸检测模型      |
| TDL_MODEL_CLS_ATTRIBUTE_GENDER<br>_AGE_GLASS         | 人脸属性分类模型 (年龄, 性别, 眼镜)     |
| TDL_MODEL_CLS_ATTRIBUTE_GENDER<br>_AGE_GLASS_MASK    | 人脸属性分类模型 (年龄, 性别, 眼镜, 口罩) |
| TDL_MODEL_CLS_ATTRIBUTE_GENDER<br>_AGE_GLASS_EMOTION | 人脸属性分类模型 (年龄, 性别, 眼镜, 情绪) |

## 1.4 图像分类模型列表

| 模型名称                                    | 注释  |
|---|---|
| TDL_MODEL_CLS_MASK                      | 口罩检测模型 (0: 戴口罩, 1: 不戴口罩)  |
| TDL_MODEL_CLS_RGBLIVENESS               | 活体检测模型 (0: 活体, 1: 伪造)   |
| TDL_MODEL_CLS_ISP_SCENE                 | ISP 场景分类模型 ( “0: 雪, 1: 雾, 2: 逆光, 3: 草, 4: 普通场景” )                     |
| TDL_MODEL_CLS_HAND_GESTURE              | 手势分类模型 (0: 拳头, 1: 五指, 2: 无, 3: 二)                                     |
| TDL_MODEL_CLS_KEYPOINT_HAND<br>_GESTURE | 手势关键点分类模型 (0: 拳头, 1: 五指, 2: 四指, 3: 无, 4: 好, 5: 一, 6: 三, 7: 三 2, 8: 二) |

## 1.5 声音分类模型列表

| 模型名称                                      | 注释                                  |
|---|-------------------------------------|
| TDL_MODEL_CLS_SOUND_BABAY_CRY             | 婴儿哭声分类模型 (0: 背景, 1: 哭声)             |
| TDL_MODEL_CLS_SOUND_COMMAND_NIHAOSHIYUN   | 命令声音分类模型 (0: 背景, 1: nihaoshiyun)    |
| TDL_MODEL_CLS_SOUND_COMMAND_NIHAOSUANNENG | 命令声音分类模型 (0: 背景, 1: nihaosuan-neng) |
| TDL_MODEL_CLS_SOUND_COMMAND_XIAOAIXIAOAI  | 命令声音分类模型 (0: 背景, 1: xiaoaixiaoai)   |
| TDL_MODEL_CLS_SOUND_COMMAND               | 命令声音分类模型 (0: 背景, 1: 命令 1, 2: 命令 2)  |

## 1.6 关键点识别类模型列表

| 模型名称                                   | 注释                |
|--|-------------------|
| TDL_MODEL_KEYPOINT_LICENSE_PLATE       | 车牌关键点检测模型         |
| TDL_MODEL_KEYPOINT_HAND                | 手部关键点检测模型         |
| TDL_MODEL_KEYPOINT_YOLOV8POSE_PERSON17 | 人体 17 个关键点检测模型    |
| TDL_MODEL_KEYPOINT_SIMCC_PERSON17      | SIMCC 17 个关键点检测模型 |

## 1.7 线识别类模型列表

| 模型名称                    | 注释     |
|-------------------------|--------|
| TDL_MODEL_LSTR_DET_LANE | 车道检测模型 |

## 1.8 车牌识别类模型列表

| 模型名称                                | 注释     |
|-------------------------------------|--------|
| TDL_MODEL_RECOGNITION_LICENSE_PLATE | 车牌识别模型 |

## 1.9 分割类模型列表

|                                   |  |
|-----------------------------------|--|
| TDL_MODEL_YOLOV8_SEG_COCO80       | YOLOv8 COCO80 分割模型                           |
| TDL_MODEL_SEG_PERSON_FACE_VEHICLE | 人、脸与车辆分割模型 (0: 背景, 1: 人, 2: 脸, 3: 车辆, 4: 车牌) |
| TDL_MODEL_SEG_MOTION              | 动作分割模型 (0: 静态, 2: 过渡, 3: 运动)                 |

## 1.10 特征值提取类模型列表

|                              |                      |
|------------------------------|----------------------|
| TDL_MODEL_FEATURE_IMG        | 图像特征提取模型             |
| TDL_MODEL_IMG_FEATURE_CLIP   | CLIP 图像嵌入模型          |
| TDL_MODEL_TEXT_FEATURE_CLIP  | CLIP 文本嵌入模型          |
| TDL_MODEL_FEATURE_CVIFACE    | Cviface 256 维特征提取模型  |
| TDL_MODEL_FEATURE_BMFACE_R34 | ResNet34 512 维特征提取模型 |
| TDL_MODEL_FEATURE_BMFACE_R50 | ResNet50 512 维特征提取模型 |



# 2 结构体参考

## 2.1 TDLDataTypeE

### 【说明】

数据类型枚举类

### 【定义】

```
typedef enum {
    TDL_TYPE_INT8 = 0, /**< Equals to int8_t. */
    TDL_TYPE_UINT8, /**< Equals to uint8_t. */
    TDL_TYPE_INT16, /**< Equals to int16_t. */
    TDL_TYPE_UINT16, /**< Equals to uint16_t. */
    TDL_TYPE_INT32, /**< Equals to int32_t. */
    TDL_TYPE_UINT32, /**< Equals to uint32_t. */
    TDL_TYPE_BF16, /**< Equals to bf17. */
    TDL_TYPE_FP16, /**< Equals to fp16. */
    TDL_TYPE_FP32, /**< Equals to fp32. */
    TDL_TYPE_UNKOWN /**< Equals to unkown. */
} TDLDataTypeE;
```

### 【成员】

| 数据类型枚举类         | 注释                             |
|-----------------|--------------------------------|
| TDL_TYPE_INT8   | 有符号 8 位整数                      |
| TDL_TYPE_UINT8  | 无符号 8 位整数                      |
| TDL_TYPE_INT16  | 有符号 16 位整数                     |
| TDL_TYPE_UINT16 | 无符号 16 位整数                     |
| TDL_TYPE_INT32  | 有符号 32 位整数                     |
| TDL_TYPE_UINT32 | 无符号 32 位整数                     |
| TDL_TYPE_BF16   | 16 位浮点数 (1 位符号, 8 位指数和 7 位尾数)  |
| TDL_TYPE_FP16   | 16 位浮点数 (1 位符号, 5 位指数和 10 位尾数) |
| FTDL_TYPE_FP32  | 32 位浮点数                        |

## 2.2 TDLBox

**【说明】**

box 的坐标数据

**【定义】**

```
typedef struct {
    float x1;
    float y1;
    float x2;
    float y2;
} TDLBox;
```

**【成员】**

| 数据类型枚举类 | 注释            |
|---------|---------------|
| x1      | box 左上角 x 的坐标 |
| y1      | box 左上角 y 的坐标 |
| x2      | box 右下角 x 的坐标 |
| y2      | box 右下角 y 的坐标 |

## 2.3 TDLFeature

**【说明】**

特征值数据

**【定义】**

```
typedef struct {
    int8_t *ptr;
    uint32_t size;
    TDLDataTypeE type;
} TDLFeature;
```

**【成员】**

| 数据类型枚举类 | 注释    |
|---------|-------|
| ptr     | 特征值数据 |
| size    | 数据大小  |
| type    | 数据类型  |

## 2.4 TDLPoints

**【说明】**

坐标队列数据

**【定义】**

```
typedef struct {
    float *x;
    float *y;
    uint32_t size;
    float score;
} TDLPoints;
```

**【成员】**

| 数据类型枚举类 | 注释         |
|---------|------------|
| x       | 坐标队列的 x 数据 |
| y       | 坐标队列的 y 数据 |
| size    | 坐标队列的大小    |
| score   | 分数         |

## 2.5 TDLLandmarkInfo

**【说明】**

特征点信息

**【定义】**

```
typedef struct {
    float x;
    float y;
    float score;
} TDLLandmarkInfo;
```

**【成员】**

| 数据类型枚举类 | 注释        |
|---------|-----------|
| x       | 特征点的 x 坐标 |
| y       | 特征点的 y 坐标 |
| score   | 分数        |

## 2.6 TDLObjectInfo

### 【说明】

目标检测信息

### 【定义】

```
typedef struct {
    TDLBox box;
    float score;
    int class_id;
    uint32_t landmark_size;
    TDLLandmarkInfo *landmark_properity;
    TDLObjectTypeE obj_type;
} TDLObjectInfo;
```

### 【成员】

| 数据类型枚举类         | 注释         |
|-----------------|------------|
| score           | 目标检测的分数    |
| class_id        | 目标检测的类别 id |
| landmark_size   | 目标检测的特征点大小 |
| TDLLandmarkInfo | 目标检测的特征点信息 |
| obj_type        | 目标检测的类型    |

## 2.7 TDLObject

### 【说明】

目标检测数据

### 【定义】

```
typedef struct {
    uint32_t size;
    uint32_t width;
    uint32_t height;

    TDLObjectInfo *info;
} TDLObject;
```

### 【成员】

| 数据类型枚举类 | 注释        |
|---------|-----------|
| size    | 目标检测的个数   |
| width   | 目标检测图像的宽度 |
| height  | 目标检测图像的高度 |
| info    | 目标检测信息    |

## 2.8 TDLFaceInfo

### 【说明】

人脸信息

### 【定义】

```
typedef struct {  
    char name[128];  
    float score;  
    uint64_t track_id;  
    TDLBox box;  
    TDLPoints landmarks;  
    TDLFeature feature;  
  
    float gender_score;  
    float glass_score;  
    float age;  
    float liveness_score;  
    float hardhat_score;  
    float mask_score;  
  
    float recog_score;  
    float face_quality;  
    float pose_score;  
    float blurness;  
} TDLFaceInfo;
```

### 【成员】

| 数据类型枚举类        | 注释          |
|----------------|-------------|
| name           | 人脸的姓名       |
| score          | 人脸的分数       |
| track_id       | 人脸的追踪 id    |
| box            | 人脸的 box 信息  |
| landmarks      | 人脸的特征点      |
| feature        | 人脸的特征值      |
| gender_score   | 人脸的性别分数     |
| glass_score    | 人脸是否带眼镜     |
| age            | 人脸的年龄       |
| liveness_score | 人脸的活体分数     |
| hardhat_score  | 人脸的是否带安全帽分数 |
| recog_score    | 人脸的识别率分数    |
| face_quality   | 人脸的质量分数     |
| pose_score     | 人脸的姿态分数     |
| blurness       | 人脸的模糊度      |

## 2.9 TDLFace

**【说明】**

人脸数据

**【定义】**

```
typedef struct {
    uint32_t size;
    uint32_t width;
    uint32_t height;
    TDLFaceInfo *info;
} TDLFace;
```

**【成员】**

| 数据类型枚举类 | 注释      |
|---------|---------|
| size    | 人脸的个数   |
| width   | 人脸图像的宽度 |
| height  | 人脸图像的高度 |
| info    | 人脸信息    |

## 2.10 TDLClassInfo

**【说明】**

分类信息

**【定义】**

```
typedef struct {
    int32_t class_id;
    float score;
} TDLClassInfo;
```

**【成员】**

| 数据类型枚举类  | 注释    |
|----------|-------|
| class_id | 分类的类别 |
| score    | 分类的分数 |

## 2.11 TDLClass

**【说明】**

分类数据

**【定义】**

```
typedef struct {
    uint32_t size;
    TDLClassInfo *info;
} TDLClass;
```

**【成员】**

| 数据类型枚举类 | 注释    |
|---------|-------|
| size    | 分类的个数 |
| info    | 分类信息  |

## 2.12 TDLKeypointInfo

**【说明】**

关键点信息

**【定义】**

```
typedef struct {
    float x;
    float y;
    float score;
} TDLKeypointInfo;
```

**【成员】**

| 数据类型枚举类 | 注释        |
|---------|-----------|
| x       | 关键点的 x 坐标 |
| y       | 关键点的 y 坐标 |
| score   | 关键点的分数    |

## 2.13 TDLKeypoint

**【说明】**

关键点数据

**【定义】**

```
typedef struct {
    uint32_t size;
    uint32_t width;
    uint32_t height;
    TDLKeypointInfo *info;
} TDLKeypoint;
```

**【成员】**

| 数据类型枚举类 | 注释     |
|---------|--------|
| size    | 关键点的个数 |
| width   | 图像的宽度  |
| height  | 图像的高度  |
| info    | 关键点信息  |

## 2.14 TDLSegmentation

**【说明】**

语义分割数据

**【定义】**

```
typedef struct {
    uint32_t width;
    uint32_t height;
    uint32_t output_width;
    uint32_t output_height;
    uint8_t *class_id;
    uint8_t *class_conf;
} TDLSegmentation;
```

**【成员】**

| 数据类型枚举类       | 注释      |
|---------------|---------|
| width         | 图像的宽度   |
| height        | 图像的高度   |
| output_width  | 输出图像的宽度 |
| output_height | 输出图像的高度 |
| class_id      | 分类的类别   |
| class_conf    | 分类的坐标信息 |



## 2.15 TDLInstanceSegInfo

### 【说明】

实例分割信息

### 【定义】

```
typedef struct {
    uint8_t *mask;
    float *mask_point;
    uint32_t mask_point_size;
    TDLObjectInfo *obj_info;
} TDLInstanceSegInfo;
```

### 【成员】

| 数据类型枚举类         | 注释                  |
|-----------------|---------------------|
| mask            | 实例分割的 mask 队列       |
| mask_point      | 实例分割的 mask_point 队列 |
| mask_point_size | 实例分割的 point 个数      |
| output_height   | 输出图像的高度             |
| obj_info        | 实例分割的目标检测信息         |

## 2.16 TDLInstanceSeg

### 【说明】

实例分割数据

### 【定义】

```
typedef struct {
    uint32_t size;
    uint32_t width;
    uint32_t height;
    uint32_t mask_width;
    uint32_t mask_height;
    TDLInstanceSegInfo *info;
} TDLInstanceSeg;
```

### 【成员】

| 数据类型枚举类     | 注释       |
|-------------|----------|
| size        | 实例分割的个数  |
| width       | 图像的宽度    |
| height      | 图像的高度    |
| mask_width  | mask 的宽度 |
| mask_height | mask 的高度 |
| info        | 实例分割信息   |

# 2.17 TDLLanePoint

【说明】

线检测的坐标点

【定义】

```
typedef struct {
    float x[2];
    float y[2];
    float score;
} TDLLanePoint;
```

【成员】

| 数据类型枚举类 | 注释     |
|---------|--------|
| x       | x 坐标队列 |
| y       | y 坐标队列 |
| score   | 线检测的分数 |

# 2.18 TDLLane

【说明】

线检测数据

【定义】

```
typedef struct {
    uint32_t size;
    uint32_t width;
    uint32_t height;
    TDLLanePoint *lane;
    int lane_state;
} TDLLane;
```

【成员】

| 数据类型枚举类    | 注释     |
|------------|--------|
| size       | 线检测的个数 |
| width      | 图像的宽度  |
| height     | 图像的高度  |
| lane       | 线检测坐标点 |
| lane_state | 线条状态   |

## 2.19 TDLDepthLogits

### 【说明】

深度估计数据

### 【定义】

```
typedef struct {
    int w;
    int h;
    int8_t *int_logits;
} TDLDepthLogits;
```

### 【成员】

| 数据类型枚举类    | 注释     |
|------------|--------|
| w          | 图像的宽度  |
| h          | 图像的高度  |
| int_logits | 深度估计信息 |

## 2.20 TDLTracker

### 【说明】

追踪数据

### 【定义】

```
typedef struct {
    uint32_t size;
    uint64_t id;
    TDLBox bbox;
    int out_num;
} TDLTracker;
```

### 【成员】

| 数据类型枚举类 | 注释        |
|---------|-----------|
| size    | 追踪目标的个数   |
| id      | 追踪目标的 id  |
| bbox    | 追踪目标的 box |
| out_num | 追踪目标的小时次数 |

# 2.21 TDLOcr

【说明】

文本识别数据

【定义】

```
typedef struct {
    uint32_t size;
    char* text_info;
} TDLOcr;
```

【成员】

| 数据类型枚举类   | 注释      |
|-----------|---------|
| size      | 文本识别的个数 |
| text_info | 文本识别的信息 |

# 3 API 参考

## 3.1 句柄

### 【语法】

```
typedef void *TDLHandle;
typedef void *TDLHandleEx;
typedef void *TDLImage;
```

### 【描述】

TDL SDK 句柄，TDLHandle 是核心操作句柄，TDLHandleEx 是扩展操作句柄，TDLImage 是图像数据抽象句柄。

## 3.2 TDL\_CreateHandle

### 【语法】

```
TDLHandle TDL_CreateHandle(const int32_t tpu_device_id);
```

### 【描述】

创建一个 TDLHandle 对象。

### 【参数】

|    | 数据类型          | 参数名称          | 说明            |
|----|---------------|---------------|---------------|
| 输入 | const int32_t | tpu_device_id | 指定 TPU 设备的 ID |

## 3.3 TDL\_CreateHandleEx

### 【语法】

```
TDLHandleEx TDL_CreateHandleEx(const int32_t tpu_device_id);
```

### 【描述】

创建一个 TDLHandleEx 对象（在使用 tdl\_ex.h 中的 api 时使用）。

### 【参数】

|    | 数据类型          | 参数名称          | 说明            |
|----|---------------|---------------|---------------|
| 输入 | const int32_t | tpu_device_id | 指定 TPU 设备的 ID |

## 3.4 TDL\_DestroyHandle

### 【语法】

```
int32_t TDL_DestroyHandle(TDLHandle handle);
```

### 【描述】

销毁一个 TDLHandle 对象。

### 【参数】

|    | 数据类型      | 参数名称   | 说明                 |
|----|-----------|--------|--------------------|
| 输入 | TDLHandle | handle | 需要销毁的 TDLHandle 对象 |

## 3.5 TDL\_DestroyHandleEx

### 【语法】

```
int32_t TDL_DestroyHandleEx(TDLHandleEx handle);
```

### 【描述】

销毁一个 TDLHandleEx 对象（在使用 tdl\_ex.h 中的 api 时使用）。

### 【参数】

|    | 数据类型        | 参数名称   | 说明                   |
|----|-------------|--------|----------------------|
| 输入 | TDLHandleEx | handle | 需要销毁的 TDLHandleEx 对象 |

## 3.6 TDL\_WrapVPSSFrame

### 【语法】

```
TDLImage TDL_WrapVPSSFrame(void *vpss_frame, bool own_memory);
```

### 【描述】

包装一个 VPSS 帧为 TDLImageHandle 对象。

### 【参数】

|    | 数据类型  | 参数名称       | 说明           |
|----|-------|------------|--------------|
| 输入 | void* | vpss_frame | 需要包装的 VPSS 帧 |
| 输入 | bool  | own_memory | 是否拥有内存所有权    |

## 3.7 TDL\_ReadImage

```
TDLImage TDL_ReadImage(const char *path);
```

### 【描述】

读取一张图片为 TDLImageHandle 对象。

### 【参数】

|    | 数据类型        | 参数名称 | 说明   |
|----|-------------|------|------|
| 输入 | const char* | path | 图片路径 |

## 3.8 TDL\_ReadBin

### 【语法】

```
TDLImage TDL_ReadBin(const char *path, int count, TDLDataTypeE data_type);
```

### 【描述】

读取文件内容为 TDLImageHandle 对象。

### 【参数】

|    | 数据类型         | 参数名称      | 说明       |
|----|--------------|-----------|----------|
| 输入 | const char*  | path      | bin 文件路径 |
| 输入 | int          | count     | 文件中数据量   |
| 输入 | TDLDataTypeE | data_type | 输入数据类型   |

## 3.9 TDL\_DestroyImage

### 【语法】

```
int32_t TDL_DestroyImage(TDLImage image_handle);
```

### 【描述】

销毁一个 TDLImageHandle 对象。

### 【参数】

|    | 数据类型     | 参数名称         | 说明                      |
|----|----------|--------------|-------------------------|
| 输入 | TDLImage | image_handle | 需要销毁的 TDLImageHandle 对象 |

## 3.10 TDL\_OpenModel

### 【语法】

```
int32_t TDL_OpenModel(TDLHandle handle,
                      const TDLModel model_id,
                      const char *model_path);
```

### 【描述】

加载指定类型的模型到 TDLHandle 对象中。

### 【参数】

|    | 数据类型           | 参数名称       | 说明           |
|----|----------------|------------|--------------|
| 输入 | TDLHandle      | handle     | TDLHandle 对象 |
| 输入 | const TDLModel | model_id   | 模型类型枚举       |
| 输入 | const char*    | model_path | 模型路径         |

## 3.11 TDL\_CloseModel

### 【语法】

```
int32_t TDL_CloseModel(TDLHandle handle,
                       const TDLModel model_id);
```

### 【描述】

卸载指定类型的模型并释放相关资源。



## 【参数】

|    | 数据类型           | 参数名称     | 说明           |
|----|----------------|----------|--------------|
| 输入 | TDLHandle      | handle   | TDLHandle 对象 |
| 输入 | const TDLModel | model_id | 模型类型枚举       |

## 3.12 TDL\_Detection

## 【语法】

```
int32_t TDL_Detection(TDLHandle handle,
                      const TDLModel model_id,
                      TDLImage image_handle,
                      TDLObject *object_meta);
```

## 【描述】

执行指定模型的推理检测，并返回检测结果元数据。

## 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLObject*     | object_meta  | 输出检测结果元数据         |

## 3.13 TDL\_FaceDetection

## 【语法】

```
int32_t TDL_FaceDetection(TDLHandle handle,
                          const TDLModel model_id,
                          TDLImage image_handle,
                          TDLFace *face_meta);
```

## 【描述】

执行人脸检测并返回人脸检测结果元数据。

## 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLFace*       | face_meta    | 输出人脸检测结果元数据       |

## 3.14 TDL\_FaceAttribute

### 【语法】

```
int32_t TDL_FaceAttribute(TDLHandle handle,
                          const TDLModel model_id,
                          TDLImage image_handle,
                          TDLFace *face_meta);
```

### 【描述】

执行人脸属性分析，需配合已检测到的人脸框进行特征分析。

### 【参数】

|       | 数据类型           | 参数名称         | 说明                |
|-------|----------------|--------------|-------------------|
| 输入    | TDLHandle      | handle       | TDLHandle 对象      |
| 输入    | const TDLModel | model_id     | 模型类型枚举            |
| 输入    | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输入/输出 | TDLFace*       | face_meta    | 输入人脸检测结果，输出补充属性信息 |

## 3.15 TDL\_FaceLandmark

### 【语法】

```
int32_t TDL_FaceLandmark(TDLHandle handle,
                          const TDLModel model_id,
                          TDLImage image_handle,
                          TDLImage *crop_image_handle,
                          TDLFace *face_meta);
```

### 【描述】

执行人脸关键点检测，在已有的人脸检测结果上补充关键点坐标。

### 【参数】

|       | 数据类型           | 参数名称         | 说明                                     |
|-------|----------------|--------------|--|
| 输入    | TDLHandle      | handle       | TDLHandle 对象                           |
| 输入    | const TDLModel | model_id     | 模型类型枚举                                 |
| 输入    | TDLImage       | image_handle | TDLImageHandle 对象                      |
| 输入    | TDLImage       | crop_image   | TDLImageHandle 对象, 裁剪后的图像, 为 NULL 时不生效 |
| 输入/输出 | TDLFace*       | face_meta    | 输入人脸检测结果, 输出补充关键点坐标                    |

## 3.16 TDL\_Classification

### 【语法】

```
int32_t TDL_Classification(TDLHandle handle,
                           const TDLModel model_id,
                           TDLImage image_handle,
                           TDLClassInfo *class_info);
```

### 【描述】

执行通用分类识别。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLClassInfo*  | class_info   | 输出分类结果            |

## 3.17 TDL\_ObjectClassification

### 【语法】

```
int32_t TDL_ObjectClassification(TDLHandle handle,
                                  const TDLModel model_id,
                                  TDLImage image_handle,
                                  TDLObject *object_meta,
                                  TDLClass *class_info);
```

### 【描述】

对检测到的目标进行细粒度分类。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输入 | TDLObject*     | object_meta  | 已检测到的目标信息         |
| 输出 | TDLClass*      | class_info   | 输出目标分类结果          |

## 3.18 TDL\_KeypointDetection

### 【语法】

```
int32_t TDL_KeypointDetection(TDLHandle handle,
                              const TDLModel model_id,
                              TDLImage image_handle,
                              TDLKeypoint *keypoint_meta);
```

### 【描述】

执行人体/物体关键点检测。

### 【参数】

|    | 数据类型           | 参数名称          | 说明                |
|----|----------------|---------------|-------------------|
| 输入 | TDLHandle      | handle        | TDLHandle 对象      |
| 输入 | const TDLModel | model_id      | 模型类型枚举            |
| 输入 | TDLImage       | image_handle  | TDLImageHandle 对象 |
| 输出 | TDLKeypoint*   | keypoint_meta | 输出关键点坐标及置信度       |

## 3.19 TDL\_InstanceSegmentation

### 【语法】

```
int32_t TDL_InstanceSegmentation(TDLHandle handle,
                                  const TDLModel model_id,
                                  TDLImage image_handle,
                                  TDLInstanceSeg *inst_seg_meta);
```

### 【描述】

执行实例分割 (Instance Segmentation)，检测图像中每个独立目标的像素级轮廓。

### 【参数】

|    | 数据类型            | 参数名称         | 说明                       |
|----|-----------------|--------------|--------------------------|
| 输入 | TDLHandle       | handle       | TDLHandle 对象             |
| 输入 | const TDLModel  | model_id     | 模型类型枚举                   |
| 输入 | TDLImage        | image_handle | TDLImageHandle 对象        |
| 输出 | TDLInstanceSeg* | inst_seg     | 输出实例分割结果（包含 mask 和 bbox） |

## 3.20 TDL\_SemanticSegmentation

### 【语法】

```
int32_t TDL_SemanticSegmentation(TDLHandle handle,
    const TDLModel model_id,
    TDLImage image_handle,
    TDLSegmentation *seg_meta);
```

### 【描述】

执行语义分割（Semantic Segmentation），对图像进行像素级分类。

### 【参数】

|    | 数据类型             | 参数名称         | 说明                |
|----|------------------|--------------|-------------------|
| 输入 | TDLHandle        | handle       | TDLHandle 对象      |
| 输入 | const TDLModel   | model_id     | 模型类型枚举            |
| 输入 | TDLImage         | image_handle | TDLImageHandle 对象 |
| 输出 | TDLSegmentation* | seg_meta     | 输出分割结果（类别标签图）     |

## 3.21 TDL\_FeatureExtraction

### 【语法】

```
int32_t TDL_FeatureExtraction(TDLHandle handle,
    const TDLModel model_id,
    TDLImage image_handle,
    TDLFeature *feature_meta);
```

### 【描述】

提取图像的深度特征向量。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLFeature*    | feature_meta | 输出特征向量            |

## 3.22 TDL\_LaneDetection

【语法】

```
int32_t TDL_LaneDetection(TDLHandle handle,
                          const TDLModel model_id,
                          TDLImage image_handle,
                          TDLLane *lane_meta);
```

【描述】

检测道路车道线及其属性。

【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLLane*       | lane_meta    | 输出车道线坐标及属性        |

## 3.23 TDL\_DepthStereo

【语法】

```
int32_t TDL_DepthStereo(TDLHandle handle,
                        const TDLModel model_id,
                        TDLImage image_handle,
                        TDLDepthLogits *depth_logist);
```

【描述】

基于双目立体视觉的深度估计，输出深度置信度图。

【参数】

|    | 数据类型            | 参数名称         | 说明                |
|----|-----------------|--------------|-------------------|
| 输入 | TDLHandle       | handle       | TDLHandle 对象      |
| 输入 | const TDLModel  | model_id     | 模型类型枚举            |
| 输入 | TDLImage        | image_handle | TDLImageHandle 对象 |
| 输出 | TDLDepthLogits* | depth_logist | 输出深度置信度数据         |

## 3.24 TDL\_Tracking

### 【语法】

```
int32_t TDL_Tracking(TDLHandle handle,
                    const TDLModel model_id,
                    TDLImage image_handle,
                    TDLObject *object_meta,
                    TDLTracker *tracker_meta);
```

### 【描述】

多目标跟踪，基于检测结果进行跨帧目标关联。

### 【参数】

|       | 数据类型           | 参数名称         | 说明                |
|-------|----------------|--------------|-------------------|
| 输入    | TDLHandle      | handle       | TDLHandle 对象      |
| 输入    | const TDLModel | model_id     | 模型类型枚举            |
| 输入    | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输入/输出 | TDLObject*     | object_meta  | 输入检测结果，输出补充跟踪 ID  |
| 输出    | TDLTracker*    | tracker_meta | 输出跟踪器状态信息         |

## 3.25 TDL\_CharacterRecognition

### 【语法】

```
int32_t TDL_CharacterRecognition(TDLHandle handle,
                                const TDLModel model_id,
                                TDLImage image_handle,
                                TDLOcr *char_meta);
```

### 【描述】

字符识别，支持文本检测与识别。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                |
|----|----------------|--------------|-------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象      |
| 输入 | const TDLModel | model_id     | 模型类型枚举            |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象 |
| 输出 | TDLOcr*        | char_meta    | 输出识别结果（文本内容和位置）   |

### 3.26 TDL\_LoadModelConfig

【语法】

```
int32_t TDL_LoadModelConfig(TDLHandle handle,
                             const char *model_config_json_path);
```

【描述】

加载模型配置信息，加载后可以仅通过模型 id 去打开模型。

【参数】

|    | 数据类型        | 参数名称                   | 说明   |
|----|-------------|------------------------|--|
| 输入 | TDLHandle   | handle                 | TDLHandle 对象   |
| 输入 | const char* | model_config_json_path | 模型配置文件路径，如果为 NULL，默认使用 configs/model/model_config.json |

### 3.27 TDL\_SetModelDir

【语法】

```
int32_t TDL_SetModelDir(TDLHandle handle,
                         const char *model_dir);
```

【描述】

设置模型文件夹路径。

【参数】

|    | 数据类型        | 参数名称      | 说明                            |
|----|-------------|-----------|-------------------------------|
| 输入 | TDLHandle   | handle    | TDLHandle 对象                  |
| 输入 | const char* | model_dir | 为 tdl_models 仓库路径（下面各平台的子文件夹） |



## 3.28 TDL\_SetModelThreshold

### 【语法】

```
int32_t TDL_SetModelThreshold(TDLHandle handle,
                             const TDLModel model_id,
                             float threshold);
```

### 【描述】

设置模型的阈值。

### 【参数】

|    | 数据类型           | 参数名称      | 说明           |
|----|----------------|-----------|--------------|
| 输入 | TDLHandle      | handle    | TDLHandle 对象 |
| 输入 | const TDLModel | model_id  | 要设置的模型类型枚举值  |
| 输入 | float          | threshold | 模型的阈值        |

## 3.29 TDL\_IspClassification

### 【语法】

```
int32_t TDL_IspClassification(TDLHandle handle,
                              const TDLModel model_id,
                              TDLImage image_handle,
                              TDLIspMeta *isp_meta,
                              TDLClass *class_info);
```

### 【描述】

执行 ISP 图像分类任务。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                 |
|----|----------------|--------------|--------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象       |
| 输入 | const TDLModel | model_id     | 指定目标分类模型类型枚举值      |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象  |
| 输入 | TDLIspMeta*    | isp_meta     | 输入参数, 包含 isp 相关的数据 |
| 输出 | TDLClass*      | class_info   | 输出参数, 存储目标分类结果     |

## 3.30 TDL\_Keypoint

### 【语法】

```
int32_t TDL_Keypoint(TDLHandle handle,
                    const TDLModel model_id,
                    TDLImage image_handle,
                    TDLKeypoint *keypoint_meta);
```

### 【描述】

执行关键点检测任务。

### 【参数】

|    | 数据类型           | 参数名称          | 说明                   |
|----|----------------|---------------|----------------------|
| 输入 | TDLHandle      | handle        | TDLHandle 对象         |
| 输入 | const TDLModel | model_id      | 指定关键点检测模型类型枚举值       |
| 输入 | TDLImage       | image_handle  | TDLImageHandle 对象    |
| 输出 | TDLKeypoint*   | keypoint_meta | 输出参数，存储检测到的关键点坐标及置信度 |

## 3.31 TDL\_DetectionKeypoint

### 【语法】

```
int32_t TDL_DetectionKeypoint(TDLHandle handle,
                              const TDLModel model_id,
                              TDLImage image_handle,
                              TDLObject *object_meta);
```

### 【描述】

执行关键点检测任务（根据目标的坐标进行裁剪后再执行关键点检测）。

### 【参数】

|    | 数据类型           | 参数名称         | 说明                   |
|----|----------------|--------------|----------------------|
| 输入 | TDLHandle      | handle       | TDLHandle 对象         |
| 输入 | const TDLModel | model_id     | 指定关键点检测模型类型枚举值       |
| 输入 | TDLImage       | image_handle | TDLImageHandle 对象    |
| 输出 | TDLObject*     | object_meta  | 输出参数，存储检测到的关键点坐标及置信度 |

## 3.32 TDL\_IntrusionDetection

### 【语法】

```
int32_t TDL_IntrusionDetection(TDLHandle handle,
                               TDLPoints *regions,
                               TDLBox *box,
                               bool *is_intrusion);
```

### 【描述】

执行入侵检测。

### 【参数】

|    | 数据类型       | 参数名称         | 说明            |
|----|------------|--------------|---------------|
| 输入 | TDLHandle  | handle       | TDLHandle 对象  |
| 输入 | TDLPoints* | regions      | 背景区域点集数组      |
| 输入 | TDLBox*    | box          | 检测区域 bbox     |
| 输出 | bool*      | is_intrusion | 输出参数，存储入侵检测结果 |

## 3.33 TDL\_MotionDetection

### 【语法】

```
int32_t TDL_MotionDetection(TDLHandle handle,
                             TDLImage background,
                             TDLImage detect_image,
                             TDLObject *roi,
                             uint8_t threshold,
                             double min_area,
                             TDLObject *obj_meta);
```

### 【描述】

执行移动侦测任务。

### 【参数】

|    | 数据类型       | 参数名称         | 说明           |
|----|------------|--------------|--------------|
| 输入 | TDLHandle  | handle       | TDLHandle 对象 |
| 输入 | TDLImage   | back-ground  | 背景图像         |
| 输入 | TDLImage   | detect_image | 检测图像         |
| 输入 | TDLObject* | roi          | 检测区域         |
| 输入 | uint8_t    | threshold    | 阈值           |
| 输入 | double     | min_area     | 最小面积         |
| 输出 | TDLObject* | obj_meta     | 输出参数, 存储检测结果 |

## 3.34 TDL\_APP\_Init

### 【语法】

```
int32_t TDL_APP_Init(TDLHandle handle,
                    const char *task,
                    const char *config_file,
                    char ***channel_names,
                    uint8_t *channel_size);
```

### 【描述】

初始化 APP 任务。

### 【参数】

|    | 数据类型        | 参数名称          | 说明                |
|----|-------------|---------------|-------------------|
| 输入 | TDLHandle   | handle        | TDLHandle 对象      |
| 输入 | const char* | task          | APP 任务名称          |
| 输入 | const char* | config_file   | APP 的 json 配置文件路径 |
| 输出 | char***     | channel_names | 每一路视频流的名称信息       |
| 输出 | uint8_t*    | channel_size  | 视频流的路数            |

## 3.35 TDL\_APP\_SetFrame

### 【语法】

```
int32_t TDL_APP_SetFrame(TDLHandle handle,
                        const char *channel_name,
                        TDLImage image_handle,
```

(下页继续)

(续上页)

```
uint64_t frame_id,
int buffer_size);
```

【描述】

往 APP 送帧。

【参数】

|    | 数据类型        | 参数名称         | 说明                             |
|----|-------------|--------------|--------------------------------|
| 输入 | TDLHandle   | handle       | TDLHandle 对象                   |
| 输入 | const char* | channel_name | 当前 channel 的名称                 |
| 输入 | TDLImage    | image_handle | TDLImageHandle 对象              |
| 输入 | uint64_t    | frame_id     | 当前 TDLImageHandle 对象的 frame id |
| 输入 | int         | buffer_size  | 推理线程缓存的帧数                      |

### 3.36 TDL\_APP\_Capture

【语法】

```
int32_t TDL_APP_Capture(TDLHandle handle,
                        const char *channel_name,
                        TDLCaptureInfo *capture_info);
```

【描述】

执行人脸抓拍任务。

【参数】

|    | 数据类型            | 参数名称         | 说明             |
|----|-----------------|--------------|----------------|
| 输入 | TDLHandle       | handle       | TDLHandle 对象   |
| 输入 | const char*     | channel_name | 当前 channel 的名称 |
| 输出 | TDLCaptureInfo* | capture_info | 抓拍结果           |

### 3.37 TDL\_APP\_ObjectCounting

【语法】

```
int32_t TDL_APP_ObjectCounting(TDLHandle handle,
                                const char *channel_name,
                                TDLObjectCountingInfo *object_counting_info);
```

【描述】

执行客流统计 (TDL\_APP\_Init task 为 consumer\_counting) 或越界检测任务 (TDL\_APP\_Init task 为 cross\_detection)

【参数】

|    | 数据类型                   | 参数名称                      | 描述           |
|----|------------------------|---------------------------|--------------|
| 输入 | TDLHandle              | handle                    | TDLHandle 对象 |
| 输入 | const char*            | chan-<br>nel_name         | 当前通道名称       |
| 输出 | TDLObjectCountingInfo* | ob-<br>ject_counting_info | 统计/检测结果      |

### 3.38 TDL\_APP\_ObjectCountingSetLine

【语法】

```
int32_t TDL_APP_ObjectCountingSetLine(TDLHandle handle,
                                         const char *channel_name, int x1,
                                         int y1, int x2, int y2, int mode);
```

【描述】

客流统计或越界检测运行过程中重新设置画线位置。

【参数】

|    | 数据类型        | 参数名称              | 描述   |
|----|-------------|-------------------|--|
| 输入 | TDLHandle   | handle            | TDLHandle 对象   |
| 输入 | const char* | chan-<br>nel_name | 当前通道名称   |
| 输入 | int         | x1                | 端点 1 横坐标   |
| 输入 | int         | y1                | 端点 1 纵坐标   |
| 输入 | int         | x2                | 端点 2 横坐标   |
| 输入 | int         | y2                | 端点 2 纵坐标   |
| 输入 | int         | mode              | 对于客流统计: mode 为 0 时, 对于竖直线, 从左到右为进入, 对于非竖直线, 从上到下为进入, mode 为 1 相反。对于越界检测: mode 为 0 时, 对于竖直线, 从左到右为越过, 对于非竖直线, 从上到下为越过, mode 为 1 相反, mode 为 2 双向检测 |

### 3.39 TDL\_WrapImage

【语法】

```
int32_t TDL_WrapImage(TDLImage image,  
                      VIDEO_FRAME_INFO_S *frame);
```

【描述】

将 TDLImage 包装为 VIDEO\_FRAME\_INFO\_S。

【参数】

|    | 数据类型                | 参数名称  | 描述                |
|----|---------------------|-------|-------------------|
| 输入 | TDLImage            | image | TDLImageHandle 对象 |
| 输出 | VIDEO_FRAME_INFO_S* | frame | 输出参数, 存储包装后的帧信息   |

# 3.40 TDL\_LLMApiCall

【语法】

```
int32_t TDL_LLMApiCall(TDLHandle handle, const char *client_type,
                      const char *method_name, const char *params_json,
                      char *result_buf, size_t buf_size)
```

【描述】

向指定的 LLM 客户端发起调用请求。

【参数】

|    | 数据类型        | 参数名称        | 描述                        |
|----|-------------|-------------|---------------------------|
| 输入 | TDLHandle   | handle      | TDL_CreateHandle 返回的上下文句柄 |
| 输入 | const char* | client_type | LLM 客户端类型                 |
| 输入 | const char* | method_name | 调用的接口方法名                  |
| 输入 | const char* | params_json | 请求体的 json 字符串             |
| 输入 | size_t      | buf_size    | result_buf 可用空间大小         |
| 输出 | char*       | result_buf  | 存放调用返回的 JSON 结果或错误信息      |