TDL SDK 软件开发指南

Version: 1.1.0

Release date: 2022-6-15

目录

修订记录

Revision	Date	Description
1.0.0	2021/6/30	初稿
1.0.1	2022/2/11	新增 API 范例说明
1.1.0	2022/6/15	新增 API
1.2.0	2022/7/27	新增 API

1 声明

法律声明

本数据手册包含北京晶视智能科技有限公司（下称“晶视智能”）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按“原样”提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

联系我们

地址 北京市海淀区丰豪东路 9 号院中关村集成电路设计园（ICPARK）1 号楼

深圳市宝安区福海街道展城社区会展湾云岸广场 T10 栋

电话 +86-10-57590723 +86-10-57590724

邮编 100094（北京）518100（深圳）

官方网站 https://www.sophgo.com/

技术论坛 https://developer.sophgo.com/forum/index.html

2 功能概述

2.1 目的

Cvitek 所提供的 TDL（Turnkey Deep Learning）集成算法，用以缩短应用程序开发所需的时间。

此架构实现了 TDL 所需算法包含其前后处理，提供统一且便捷的编程接口。

目前 TDL SDK 包含移动侦测，人脸检测，人脸识别，人脸追踪，行人检测，语义分割，车牌辨识，车牌检测，活体识别，IR 活体识别，婴儿检测，哭声检测，姿态检测，手势侦测，手势识别等算法。

3 设计概述

3.1 系统架构

下图为 TDL SDK 系统架构图；TDL SDK 架构在 Cvitek 的 Middleware 及 TPU SDK 上。

主要分为三大模块：Core，Service，Application。

Core 主要提供算法相关接口，封装复杂的底层操作及算法细节。令使用者可以直接使用 VI 或 VPSS 取得的 Video Frame Buffer 进行模型推理。

TDL SDK 内部会对模型进行相应的前后处理，并完成推理。

Service 提供算法相关辅助 API，例如：绘图, 特征比对, 区域入侵判定等功能。

Application 封装应用逻辑，目前包含人脸抓拍的应用逻辑。

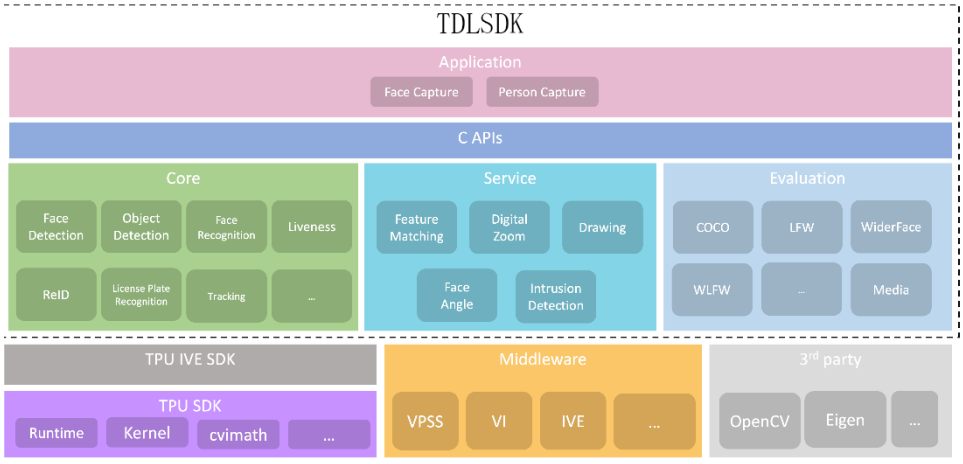

图 3.1: Figure 1.

这三个模块分别放在两个 Library 中:

模块	动态库	静态库
Core、Service	libcvi_tdl.so	libcvi_tdl.a
Application	libcv_tdl_app.so	libcv_tdl_app.a

3.2 档案结构

TDL SDK 档案结构如下:

目录名称	说明
include	TDL SDK 头文件
sample	范例源代码
doc	Markdown 格式文檔
lib	TDL SDK 静态和动态库
bin	TDL SDK 二进制文件

4 API 参考

4.1 句柄

【语法】

```
typedef void *cvitdl_handle_t;
typedef void *cvitdl_service_handle_t;
```

【描述】

TDL SDK 的句柄，不同模块之间有各自的句柄，但是创建 cvitdl_service_handle_t 模块时会需要使用到 cvitdl_handle_t 作为输入。

4.2 CVI_TDL_Core

4.2.1 通用

4.2.1.1 CVI_TDL_CreateHandle

【语法】

```
CVI_S32 CVI_TDL_CreateHandle(cvitdl_handle_t *handle);
```

【描述】

创建使用 TDL SDK 句柄。TDL SDK 会自动创建 VPSS Group。

【参数】

	数据型态	参数名称	说明
输入/输出	cvitdl_handle_t*	handle	输入句柄指针

4.2.1.2 CVI_TDL_CreateHandle2

【语法】

```
CVI_S32 CVI_TDL_CreateHandle2(cvitdl_handle_t *handle, const VPSS_GRP vpssGroupId, const
↪CVI_U8 vpssDev);
```

【描述】

创建使用 TDL SDK 句柄，并使用指定的 VPSS Group ID 及 Dev ID 创建 VPSS。

【参数】

	数据型态	参数名称	说明
输出	cvitdl_handle_t*	handle	输入句柄指针
输入	VPSS_GRP	vpssGroupId	VPSS 使用的 group id
输入	CVI_U8	vpssDev	VPSS Device id

4.2.1.3 CVI_TDL_DestroyHandle

【语法】

```
CVI_S32 CVI_TDL_DestroyHandle(cvitdl_handle_t handle);
```

【描述】

销毁创造的句柄 cvitdl_handle_t。同时销毁所有开启的模型

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	输入句柄

4.2.1.4 CVI_TDL_GetModelPath

```
const char *CVI_TDL_GetModelPath(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_
↪E model);
```

【描述】

取得内部已经设置支持的模型的模型路径。使用完毕需要自行释放 filepath 之变量。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID

【输出】

	数据型态	说明
输出	char*	模型路径指针

4.2.1.5 CVI_TDL_OpenModel

【语法】

```
CVI_S32 CVI_TDL_OpenModel(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_E
↪model, const char *filepath);
```

【描述】

开启并初始化模型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 index
输入	const char*	filepath	cvimodel 模型路径

4.2.1.6 CVI_TDL_SetSkipVpssPreprocess

【语法】

```
CVI_S32 CVI_TDL_SetSkipVpssPreprocess(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, bool skip);
```

【描述】

指定 model 不进行预处理。

TDL SDK 在默认情况下会使用内部创建的 VPSS 进行模型的预处理 (skip = false)。

当 skip 为 true 时，TDL SDK 将不会对该模型进行预处理。

模型输入必须由外部进行预处理后，再输入模型。

通常用于 VI 直接 Binding VPSS 且只使用单一模型的状况。

可以使用CVI_TDL_GetVpssChnConfig 来取得模型的 VPSS 预处理参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID
输入	bool	skip	是否跳过前处理

4.2.1.7 CVI_TDL_GetSkipVpssPreprocess

【语法】

```
CVI_S32 CVI_TDL_GetSkipVpssPreprocess(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, bool *skip);
```

【描述】

询问模型是否会在 TDL SDK 内进行预处理。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID
输出	bool*	skip	是否跳过前处理

4.2.1.8 CVI_TDL_SetVpssThread

【语法】

```
CVI_S32 CVI_TDL_SetVpssThread(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_
↪E model, const uint32_t thread);
```

【描述】

设置特定模型使用的线程 id。在 TDL SDK 内，一个 Vpss Thread 代表一组 Vpss Group 设置。

默认使用 Thread 0 为模型使用的 Vpss Group。

当在多线程上各自使用同一个 TDL SDK Handle 来进行模型推理时，必须使用此 API 指定不同的 Vpss Thread 来避免 Race Condition。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID
输入	uint32_t	thread	线程 id

4.2.1.9 CVI_TDL_SetVpssThread2

【语法】

```
CVI_S32 CVI_TDL_SetVpssThread2(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_
↪E model, const uint32_t thread, const VPSS_GRP vpssGroupId, const CVI_U8 dev);
```

【描述】

同 CVI_TDL_SetVpssThread。可以指定 Vpss Group ID。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID
输入	const uint32_t	thread	线程 id
输入	const VPSS_GRP	vpssGroupId	VPSS Group id
输入	const CVI_U8	dev	VPSS Device id

4.2.1.10 CVI_TDL_GetVpssThread

【语法】

```
CVI_S32 CVI_TDL_GetVpssThread(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_
↪E model, uint32_t *thread);
```

【描述】

取得模型使用的 thread id。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 ID
输出	uint32_t*	thread	VPSS 线程 id

4.2.1.11 CVI_S32 CVI_TDL_GetVpssGrpIds

【语法】

```
CVI_S32 CVI_TDL_GetVpssGrpIds(cvitdl_handle_t handle, VPSS_GRP **groups, uint32_t *num);
```

【描述】

取得句柄内全部使用到的 Vpss group id，使用完毕后 groups 要自行释放。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输出	VPSS_GRP **	groups	空指针的参考
输出	uint32_t*	num	groups 的长度

4.2.1.12 CVI_TDL_SetVpssTimeout

【语法】

```
CVI_S32 CVI_TDL_SetVpssTimeout(cvitdl_handle_t handle, uint32_t timeout);
```

【描述】

设置 TDL SDK 等待 VPSS 硬件超时的时间，预设为 100ms。

此设置适用于所有 TDL SDK 内的 VPSS Thread。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	uint32_t	timeout	超时时间

4.2.1.13 CVI_TDL_SetVBPool

【语法】

```
CVI_S32 CVI_TDL_SetVBPool(cvitdl_handle_t handle, uint32_t thread, VB_POOL pool_id);
```

【描述】

指定 VBPool 给 TDL SDK 内部 VPSS。指定后，TDL SDK 内部 VPSS 会直接从此 Pool 拿取内存。

若不用此 API 指定 Pool，默认由系统自动分配。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	uint32_t	thread	VPSS 线程 id
输入	VB_POOL	pool_id	VB Pool Id。若设置为 INVALID_POOLID，表示不指定 Pool，由系统自动分配。

4.2.1.14 CVI_TDL_GetVBPool

【语法】

```
CVI_S32 CVI_TDL_GetVBPool(cvitdl_handle_t handle, uint32_t thread, VB_POOL *pool_id);
```

【描述】

取得指定 VPSS 使用的 VBPool Id。若未使用CVI_TDL_SetVBPool 指定 Pool，则会得到 INVALID_POOLID。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	uint32_t	thread	VPSS 线程 id
输出	VB_POOL*	pool_id	目前使用的 VB Pool Id。

4.2.1.15 CVI_TDL_CloseAllModel

【语法】

```
CVI_S32 CVI_TDL_CloseAllModel(cvitdl_handle_t handle);
```

【描述】

卸除所有在句柄中已经加载的模型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄

4.2.1.16 CVI_TDL_CloseModel

【语法】

```
CVI_S32 CVI_TDL_CloseModel(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_E␣
↪model);
```

【描述】

卸除特定在句柄中已经加载的模型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 index

4.2.1.17 CVI_TDL_Dequantize

【语法】

```
CVI_S32 CVI_TDL_Dequantize(const int8_t *quantizedData, float *data, const uint32_t bufferSize,␣
↪const float dequantizeThreshold);
```

【描述】

Dequantize int8 数值到 Float。

【参数】

	数据型态	参数名称	说明
输入	const int8_t*	quantizedData	Int8 数据
输出	float*	data	Float 输出数据
输入	const uint32_t	bufferSize	Int8 数据数量
输入	const float	dequantizeThreshold	量化阀值

4.2.1.18 CVI_TDL_ObjectNMS

【语法】

```
CVI_S32 CVI_TDL_ObjectNMS(const cvtdl_object_t *obj, cvtdl_object_t *objNMS, const float␣
↪threshold, const char method);
```

【描述】

对 cvtdl_object_t 内的 bbox 做 Non-Maximum Suppression 算法。

【参数】

	数据型态	参数名称	说明
输入	const cvtdl_object_t*	obj	想进行 NMS 的 Object Meta
输出	cvtdl_object_t*	objNMS	NMS 后的结果
输入	const float	threshold	IOU threshold
输入	const char	method	‘u’ : Intersection over Union ‘m’ : Intersection over min area

4.2.1.19 CVI_TDL_FaceNMS

【语法】

```
CVI_S32 CVI_TDL_ObjectNMS(const cvtdl_face_t *face, cvtdl_face_t *faceNMS, const float␣
↪threshold, const char method);
```

【描述】

对 cvtdl_face_t 内的 bbox 做 Non-Maximum Suppression 算法。

【参数】

	数据型态	参数名称	说明
输入	const cvtdl_face_t*	face	想进行 NMS 的 face meta
输出	cvtdl_face_t*	faceNMS	NMS 后的结果
输入	const float	threshold	IOU threshold
输入	const char	method	‘u’ : Intersection over Union ‘m’ : Intersection over min area

4.2.1.20 CVI_TDL_FaceAlignment

【语法】

```
CVI_S32 CVI_TDL_FaceAlignment(VIDEO_FRAME_INFO_S *inFrame, const uint32_t␣
↪metaWidth, const uint32_t metaHeight, const cvtdl_face_info_t *info, VIDEO_FRAME_INFO_S␣
↪*outFrame, const bool enableGDC);
```

【描述】

对 inFrame 图像 face 进行 Face Alignment，采用 InsightFace Alignment 参数。

【参数】

	数据型态	参数名称	说明
输入	VIDEO_FRAME_INFO_S*	inFrame	输入图像
输入	const uint32_t metaWidth	metaWidth	Info 中 frame 的宽度
输入	const uint32_t metaHeight	metaHeight	Info 中 frame 的高度
输入	const cvtdl_face_info_t*	info	Face info
输出	VIDEO_FRAME_INFO_S*	outFrame	Face Alignment 后的人脸图像
输入	const bool	enableGDC	是否使用 GDC 硬件

4.2.1.21 CVI_TDL_CropImage

【语法】

```
CVI_S32 CVI_TDL_CropImage(VIDEO_FRAME_INFO_S *srcFrame, cvtdl_image_t *dst, cvtdl_
↪bbox_t *bbox, bool cvtRGB888);
```

【描述】

从 srcFrame 图像中截取 bbox 指定区域图像。

【参数】

	数据型态	参数名称	说明
输入	VIDEO_FRAME_INFO_S*	srcFrame	输入图像，目前仅支持 RGB Packed 格式
输出	cvtdl_image_t*	dst	输出图像
输入	cvtdl_bbox_t*	bbox	Bounding box
输入	bool	cvtRGB888	是否转换成 RGB888 格式输出

4.2.1.22 CVI_TDL_CropImage_Face

【语法】

```
CVI_S32 CVI_TDL_CropImage_Face(VIDEO_FRAME_INFO_S *srcFrame, cvtdl_image_t *dst,␣
↪cvtdl_face_info_t *face_info, bool align, bool cvtRGB888);
```

【描述】

从 srcFrame 图像中截取 face bbox 指定范围图像。

【参数】

	数据型态	参数名称	说明
输入	VIDEO_FRAME_INFO_S*	srcFrame	输入图像，目前仅支持 RGB Packed 格式
输出	cvtdl_image_t*	dst	输出图像
输入	cvtdl_face_info_t*	face_info	指定的 face info
输入	bool	align	是否进行 face alignmen。采用 InsightFace Alignment 参数。
输入	bool	cvtRGB888	是否转换成 RGB888 格式输出

4.2.1.23 CVI_TDL_SoftMax

【语法】

```
CVI_S32 CVI_TDL_SoftMax(const float *inputBuffer, float *outputBuffer, const uint32_t bufferSize);
```

【描述】

对 inputBuffer 计算 Softmax。

【参数】

	数据型态	参数名称	说明
输入	const float*	inputBuffer	想进行 softmax 的缓冲
输出	const float*	outputBuffer	Softmax 后的结果
输入	const uint32_t	bufferSize	缓冲大小

4.2.1.24 CVI_TDL_GetVpssChnConfig

【语法】

```
CVI_S32 CVI_TDL_GetVpssChnConfig(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, const CVI_U32 frameWidth, const CVI_U32 frameHeight, const CVI_U32 idx,␣
↪cvtdl_vpssconfig_t *chnConfig);
```

【描述】

取得在模型预处理使用的 VPSS 参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 id
输入	CVI_U32	frameWidth	输入图像宽
输入	CVI_U32	fra meHeight	输入图像高
输入	CVI_U32	idx	模型的输入 index
输出	cvtdl_vpssconfig_t*	chnConfig	回传的参数设定值

4.2.1.25 CVI_TDL_Free

```
CVI_TDL_Free(X)
```

【描述】

释放模型结果产生的数据结构。某些数据结构中包含 malloc 出来的子项，因此需要做释放。

【参数】

以下为支持的输入变量

- cvtdl_feature_t
- cvtdl_pts_t
- cvtdl_tracker_t
- cvtdl_face_info_t
- cvtdl_face_t
- cvtdl_object_info_t
- cvtdl_object_t

4.2.1.26 CVI_TDL_CopyInfo

```
CVI_TDL_CopyInfo(IN, OUT)
```

【描述】

泛型拷贝 cvitdl 结构 API。malloc 内部的指针空间并做完整复制。

【参数】

	数据型态	参数名称	说明
输入	支持型态: cvtdl_face_info_t cvtdl_object_info_t cvtdl_image_t	IN	复制来源
输出	支持型态: cvtdl_face_info_t cvtdl_object_info_t cvtdl_image_t	OUT	复制目的

4.2.1.27 CVI_TDL_RescaleMetaCenter

【描述】

将结构内的坐标还原到与输入图像相同之大小，适用于 padding 图像为上下左右，

【参数】

以下为支持的输入变量

- cvtdl_face_t
- cvtdl_object_t

4.2.1.28 CVI_TDL_RescaleMetaRB

【描述】

将结构内的坐标还原到与输入图像相同之大小，适用于 padding 图像为右下，

【参数】

以下为支持的输入变量

- cvtdl_face_t
- cvtdl_object_t

4.2.1.29 getFeatureTypeSize

```
int getFeatureTypeSize(feature_type_e type);
```

【描述】

取得特征值的单位大小。

【参数】

	数据型态	参数名称	说明
输入	feature_type_e	type	单位

【输出】

	数据型态	参数名称	说明
输出	int	X	单位为 byte 之单位大小

4.2.1.30 CVI_TDL_SetModelThreshold

【语法】

```
CVI_S32 CVI_TDL_SetModelThreshold(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, float threshold);
```

【描述】

设置模型阀值，目前仅支持针对 Detection 类型的模型进行设置。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 index
输入	float	threshold	阀值 (0.0~1.0)

4.2.1.31 CVI_TDL_GetModelThreshold

【语法】

```
CVI_S32 CVI_TDL_GetModelThreshold(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, float *threshold);
```

【描述】

取出模型阀值，目前仅支持 Detection 类型模型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 index
输出	float*	threshold	阀值

4.2.2 对象侦测

4.2.2.1 CVI_TDL_MobileDetV2_Vehicle

【语法】

```
CVI_S32 CVI_TDL_MobileDetV2_Vehicle(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *obj);
```

【描述】

使用 MobilDetV2-Vehicle 模型进行推理，此模型可侦测 Car, Motorcycle, Truck 三个类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.2 CVI_TDL_MobileDetV2_Pedestrian

【语法】

```
CVI_S32 CVI_TDL_MobileDetV2_Pedestrian(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *obj);
```

【描述】

使用 MobilDetV2-Pedestrian 系列模型进行推理，此模型可侦测 person 类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.3 CVI_TDL_MobileDetV2_Person_Vehicle

【语法】

```
CVI_S32 CVI_TDL_MobileDetV2_Person_Vehicle(cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_object_t *obj);
```

【描述】

使用 MobilDetV2-Person-Vehicle 模型进行推理，此模型可侦测人车非类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.4 CVI_TDL_MobileDetV2_Person_Pets

【语法】

```
CVI_S32 CVI_TDL_MobileDetV2_Person_Pets(cvitdl_handle_t handle, VIDEO_FRAME_INFO_
↪S *frame, cvtdl_object_t *obj);
```

【描述】

使用 MobilDetV2-Lite-Person-Pets 模型进行推理，此模型可侦测 person, cat, dog 等类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.5 CVI_TDL_MobileDetV2_COCO80

【语法】

```
CVI_S32 CVI_TDL_MobileDetV2_COCO80(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *obj);
```

【描述】

使用 MobilDetV2 COCO80 系列模型进行推理，此模型可侦测标准 COCO dataset 的 80 个类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.6 CVI_TDL_Yolov3

【语法】

```
CVI_S32 CVI_TDL_Yolov3 (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj);
```

【描述】

使用 YoloV3 模型进行推理，此模型可侦测 COCO 80 个类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.7 CVI_TDL_Yolov5

【语法】

```
CVI_S32 CVI_TDL_Yolov5 (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj);
```

【描述】

使用 YoloV5 模型进行推理，此模型可侦测 COCO 80 个类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.8 CVI_TDL_YoloX

【语法】

```
CVI_S32 CVI_TDL_YoloX(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj);
```

【描述】

使用 YoloX 模型进行推理，此模型可侦测 COCO 80 个类别。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	obj	侦测到的对象

4.2.2.9 CVI_TDL_SelectDetectClass

【语法】

```
CVI_S32 CVI_TDL_SelectDetectClass(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model, uint32_t num_classes, ...)
```

【描述】

过滤 Object Detection 模型输出结果，保留列举的类别或群组。

类别为不定参数，数量根据 num_classes 而定。

详细类别及群组 Index 可参考cvtdl_obj_class_id_e 及cvtdl_obj_det_group_type_e。

目前仅支持 MobileDetV2, YoloX 系列模型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	CVI_TDL_SUPPORTED_MODEL_E	model	模型 Index
输入	uint32_t	num_classes	保留的类别个数
输入	cvtdl_obj_class_id_e 或 cvtdl_obj_det_group_type_e	说明	留的 Class ID 或 Group ID

4.2.2.10 CVI_TDL_ThermalPerson

【语法】

```
CVI_S32 CVI_TDL_ThermalPerson(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_object_t *obj);
```

【描述】

热显图像人型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	faces	侦测到的人形

4.2.3 人脸侦测

4.2.3.1 CVI_TDL_RetinaFace

【语法】

```
CVI_S32 CVI_TDL_RetinaFace(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces);
```

【描述】

使用 RetinaFace 模型侦测人脸。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.2 CVI_TDL_RetinaFace_IR

【语法】

```
CVI_S32 CVI_TDL_RetinaFace_IR(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces);
```

【描述】

使用 RetinaFace 模型在 IR 图像中侦测人脸。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入 IR 图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.3 CVI_TDL_RetinaFace_Hardhat

【语法】

```
CVI_S32 CVI_TDL_RetinaFace_Hardhat(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces);
```

【描述】

使用 RetinaFace 模型侦测戴安全帽人脸。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入 IR 图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.4 CVI_TDL_ScrFDFace

【语法】

```
CVI_S32 CVI_TDL_ScrFDFace(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪face_t *faces);
```

【描述】

使用 ScrFD Face 模型侦测人脸。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.5 CVI_TDL_ThermalFace

【语法】

```
CVI_S32 CVI_TDL_ThermalFace(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_face_t *faces);
```

【描述】

热显图像人脸侦测。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.6 CVI_TDL_FLDet3

【语法】

```
CVI_S32 CVI_TDL_FLDet3(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces);
```

【描述】

判断传入的 faces 结构中的人脸座标点。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	侦测到的人脸
输出	cvtdl_face_t*	faces	人脸座标点

4.2.3.7 CVI_TDL_FaceQuality

【语法】

```
CVI_S32 CVI_TDL_FaceQuality(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces, bool *skip);
```

【描述】

判断传入的 faces 结构中的人脸质量评估并同时侦测人脸角度。质量受人脸清晰程度与是否遮挡影响。

人脸质量分数为 faces->info[i].face_quality，人脸角度放在 faces->info[i].head_pose 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	face	侦测到的人脸
输入	bool*	skip	Bool array： 指定哪个人脸需要做 face quality。 NULL 表示全部人脸都做。

4.2.3.8 CVI_TDL_FaceMaskDetection

【语法】

```
CVI_S32 CVI_TDL_FaceMaskDetection(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_face_t *faces);
```

【描述】

侦测戴口罩人脸。人脸分数存放在 faces->info[i].bbox.score，戴口罩人脸分数存放在 faces->info[i].mask_score。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.3.9 CVI_TDL_MaskClassification

【语法】

```
CVI_S32 CVI_TDL_MaskClassification(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪cvtdl_face_t *face);
```

【描述】

判断传入的 faces 中的所有人脸是否为戴口罩人脸。呼叫此接口前，必须先执行一次人脸侦测。戴口罩人脸分数存放在 faces->info[i].mask_score。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	faces	侦测到的人脸

4.2.4 人脸识别

4.2.4.1 CVI_TDL_FaceRecognition

【语法】

```
CVI_S32 CVI_TDL_FaceRecognition(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪cvtdl_face_t *faces);
```

【描述】

抽取人脸特征。此接口会针对 face 中所有人脸进行特征抽取。并放在 faces->info[i].feature 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.4.2 CVI_TDL_FaceRecognitionOne

【语法】

```
CVI_S32 CVI_TDL_FaceRecognitionOne(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_face_t *faces, int face_idx);
```

【描述】

抽取人脸特征。此接口仅会针对指定的 face index 进行特征抽取。并放在 faces->info[index].feature 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_face_t*	faces	侦测到的人脸
输入	int	face_idx	想进行特征抽取的 face index。-1 表示全部抽取。

4.2.4.3 CVI_TDL_FaceFeatureExtract

【语法】

```
CVI_S32 CVI_TDL_FaceFeatureExtract(cvitdl_handle_t handle, const uint8_t *rgb_pack, int␣
↪width, int height, int stride, cvtdl_face_info_t *face_info);
```

【描述】

抽取人脸特征。此接口仅会针对指定的 rgb_pack 位置进行特征抽取。并放在 face_info->feature.ptr 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	const uint8_t*	rgb_pack	输入图像 pixel 起始位置
输入	int	width	输入图像寬
输入	int	height	输入图像高
输入	int	stride	输入图像 Stride
输入/输出	cvtdl_face_info_t*	face_info	侦测到的人脸特征

4.2.4.4 CVI_TDL_FaceAttribute

【语法】

```
CVI_S32 CVI_TDL_FaceAttribute(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪cvtdl_face_t *faces);
```

【描述】

抽取人脸特征及人脸属性。此接口会针对 face 中所有人脸进行特征抽取及人脸属性。

人脸属性包含：性别, 表情, 年龄及种族，结果分别放在 faces->info[i].feature, faces->info[i].age, faces->info[i].emotion, faces->info[i].gender, faces->info[i].race。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.4.5 CVI_TDL_FaceAttributeOne

【语法】

```
CVI_S32 CVI_TDL_FaceAttributeOne(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪cvtdl_face_t *faces, int face_idx);
```

【描述】

抽取人脸特征。此接口仅会针对指定的 face index 进行特征抽取。

人脸属性包含：性别, 表情, 年龄及种族，结果分别放在 faces->info[i].feature, faces->info[i].age, faces->info[i].emotion, faces->info[i].gender, faces->info[i].race。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_face_t*	faces	侦测到的人脸
输入	int	face_idx	想进行特征抽取的 face index。-1 表示全部抽取。

4.2.4.6 CVI_TDL_MaskFaceRecognition

【语法】

```
CVI_S32 CVI_TDL_MaskFaceRecognition(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_face_t *faces);
```

【描述】

抽取戴口罩人脸特征。此接口会针对 face 中所有人脸进行特征抽取。并放在 faces->info[i].feature 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_face_t*	faces	侦测到的人脸

4.2.5 行人识别

4.2.5.1 CVI_TDL_OSNet

【语法】

```
CVI_S32 CVI_TDL_OSNet(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj);
```

【描述】

使用 person-reid 模型抽取行人特征。此接口会针对 obj 中所有的 Person 类别对象进行特征抽取。并放在 obj->info[i].feature 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_object_t*	obj	侦测到的对象

4.2.5.2 CVI_TDL_OSNetOne

【语法】

```
CVI_S32 CVI_TDL_OSNetOne(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj, int obj_idx);
```

【描述】

使用 person-reid 模型抽取行人特征。此接口仅会针对指定的 obj 对象进行特征抽取。并放在 obj->info[i].feature 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_object_t*	obj	侦测到的对象/ 输出行人特征
输入	int	obj_idx	想进行特征抽取的对象 index。-1 表示全部抽取。

4.2.6 手势识别

4.2.6.1 CVI_TDL_Hand_Detection

【语法】

```
CVI_S32 CVI_TDL_Hand_Detection(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *meta);
```

【描述】

手部框侦测。并将结果放在 meta->info[i] 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	cvtdl_object_t*	meta	侦测到的手框

4.2.6.2 CVI_TDL_HandClassification

【语法】

```
CVI_S32 CVI_TDL_HandClassification(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *meta);
```

【描述】

手势分类算法，此接口仅会针对指定的 frame 进行手势识别。并将结果放在 meta->info[i].name 与 meta->info[i].bbox.score 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_object_t*	meta	侦测到的手框/ 手势分类

4.2.6.3 CVI_TDL_HandKeypoint

【语法】

```
CVI_S32 CVI_TDL_HandKeypoint(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_handpose21_meta_ts *meta);
```

【描述】

手的关键点输出。并放在 meta->info[i] 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入/输出	cvtdl_handpose21_meta_ts*	meta	侦测到的手框/21个手部关节点

4.2.6.4 CVI_TDL_HandKeypointClassification

【语法】

```
CVI_S32 CVI_TDL_HandKeypointClassification(const cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_handpose21_meta_t *meta);
```

【描述】

手的关键点输出。并放在 meta->info[i] 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入 21 对手部特征点，x, y 依序放入 frame->stVFrame.pu8VirAddr[0]、frame->stVFrame.u32Height=1、frame->stVFrame.u32Width=42*sizeof(float)
输出	cvtdl_handpose21_meta_t*	meta	手势 meta->label、手势分数 meta->score

4.2.7 对象追踪

4.2.7.1 CVI_TDL_DeepSORT_Init

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_Init(const cvitdl_handle_t handle, bool use_specific_counter);
```

【描述】

初始化 DeepSORT 算法。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	bool	use_specific_counter	是否每一个对象类别各自分配 id

4.2.7.2 CVI_TDL_DeepSORT_GetDefaultConfig

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_GetDefaultConfig(cvtdl_deepsort_config_t *ds_conf);
```

【描述】

取得 DeepSORT 默认参数。

【参数】

	数据型态	参数名称	说明
输入	cvtdl_deepsort_config_t*	ds_conf	DeepSORT 参数

4.2.7.3 CVI_TDL_DeepSORT_SetConfig

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_SetConfig(const cvitdl_handle_t handle , cvtdl_deepsort_config_
↪t *ds_conf, int cvi_tdl_obj_type, bool show_config);
```

【描述】

设置 DeepSORT 参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	cvtdl_deepsort_config_t*	ds_conf	DeepSORT 参数
输入	int	cvi_tdl_obj_type	-1 表示此为默认设置。非-1 值表示针对 cvitdl_ob j_type 的类别设置参数。
输入	bool	show_config	显示设置

4.2.7.4 CVI_TDL_DeepSORT_GetConfig

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_GetConfig(const cvitdl_handle_t handle , cvtdl_deepsort_config_
↪t *ds_conf, int cvi_tdl_obj_type);
```

【描述】

询问 DeepSORT 设置的参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	TDL SDK 句柄
输出	cvtdl_deepsort_config_t*	ds_conf	DeepSORT 参数
输入	int	cvi_tdl_obj_type	-1 表示取得默认参数。非-1 值表示针对 cvitdl_ob j_type 的类别设置的参数

4.2.7.5 CVI_TDL_DeepSORT_CleanCounter

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_CleanCounter(const cvitdl_handle_t handle);
```

【描述】

清除 DeepSORT 纪录的 ID counter。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄

4.2.7.6 CVI_TDL_DeepSORT_Obj

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_Obj(const cvitdl_handle_t handle, cvtdl_object_t *obj, cvtdl_
↪tracker_t *tracker, bool use_reid);
```

【描述】

追踪对象，更新 Tracker 状态。

此接口会赋予每个 Object 一个 Unique ID。

可从 obj->info[i].unique_id 取得。tracker_t 会纪录 DeepSORT 对每个 object 的追踪状态及目前的预测 Bounding Box。

若想使用对象外观特征进行追踪，需将 use_reid 设置 true，并在追踪之前使用 CVI_TDL_OSNet 进行特征抽取。

目前特征抽取只支持人型。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	cvtdl_object_t*	obj	想进行追踪的对象
输出	cvtdl_tracker_t*	tracker	对象的追踪状态
输入	bool	use_reid	是否使用对象外观特征进行追踪

4.2.7.7 CVI_TDL_DeepSORT_Face

【语法】

```
CVI_S32 CVI_TDL_DeepSORT_Face(const cvitdl_handle_t handle, cvtdl_face_t *face, cvtdl_
↪tracker_t *tracker, bool use_reid);
```

【描述】

追踪人脸，更新 Tracker 状态。

此接口会赋予每个人脸一个 Unique ID。可从 face->info[i].unique_id 取得。

tracker_t 会纪录 DeepSORT 对每个人脸的追踪状态及目前的预测 Bounding Box。

若想使用人脸特征进行追踪，use_reid 须设置为 true。

并在追踪之前调用CVI_TDL_FaceRecognition 计算人脸特征。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	cvtdl_face_t*	face	想进行追踪的人脸
输出	cvtdl_tracker_t*	tracker	人脸的追踪状态
输入	bool	use_reid	是否使用外观特征进行追踪。目前仅能设置 false

4.2.8 运动侦测

4.2.8.1 CVI_TDL_Set_MotionDetection_Background

【语法】

```
CVI_S32 CVI_TDL_Set_MotionDetection_Background(const cvitdl_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame);
```

【描述】

设置 Motion Detection 背景，第一次运行此接口时会对 Motion Detection 进行初始化，后续再调用次接口仅会更新背景。

TDL SDK 中 Motion Detection 使用帧差法。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	背景

4.2.8.2 CVI_TDL_MotionDetection

【语法】

```
CVI_S32 CVI_TDL_MotionDetection(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *objects, uint8_t threshold, double min_area);
```

【描述】

使用帧差法侦测对象。侦测结果会存放在 objects 内。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	图像
输出	cvtdl_object_t*	object	运动侦测结果
输入	uint8_t	threshold	帧差法阀值，须为 0-255
输入	double	min_area	最小对象面积 (Pixels)，过滤掉小于此数值面积的物件。

4.2.8.3 CVI_TDL_Set_MotionDetection_ROI

【语法】

```
CVI_S32 CVI_TDL_Set_MotionDetection_ROI(const cvitdl_handle_t handle, MDROI_t *roi_s);
```

【描述】

使用帧差法侦测对象。侦测结果会存放在 objects 内。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	MDROI_t*	roi_s	设定移动侦测区域

4.2.9 车牌识别

4.2.9.1 CVI_TDL_LicensePlateDetection

【语法】

```
CVI_S32 CVI_TDL_LicensePlateDetection(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_object_t *vehicle_meta);
```

【描述】

车牌侦测。呼叫此 API 之前，必须先执行一次车辆侦测。

此算法会在已侦测到的对象上进行车牌侦测。

车牌位置会放在 obj->info[i].vehicle_properity->license_pts 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	图像
输入	cvtdl_object_t*	obj	对象 (车辆) 侦测结果

4.2.9.2 CVI_TDL_LicensePlateRecognition_TW

```
CVI_S32 CVI_TDL_LicensePlateRecognition_TW(const cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_object_t *obj);
```

【描述】

对传入的 obj 中所有车辆进行车牌识别 (台湾)。

呼叫此 API 之前，必须先调用 CVI_TDL_LicensePlateDetection 执行一次车牌侦测。

车牌号码储存在 obj->info[i].vehicle_properity->license_char 。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	图像
输入	cvtdl_object_t*	obj	车牌侦测结果

4.2.9.3 CVI_TDL_LicensePlateRecognition_CN

```
CVI_S32 CVI_TDL_LicensePlateRecognition_CN(const cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_object_t *obj);
```

【描述】

对传入的 obj 中所有车辆进行车牌识别 (大陆)。

呼叫此 API 之前，必须先调用 CVI_TDL_LicensePlateDetection 执行一次车牌侦测。

车牌号码储存在 obj->info[i].vehicle_properity->license_char 。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	图像
输入/输出	cvtdl_object_t*	bj	牌侦测结果

4.2.10 篡改侦测

4.2.10.1 CVI_TDL_TamperDetection

【语法】

```
CVI_S32 CVI_TDL_TamperDetection(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, float *moving_score);
```

【描述】

摄影机篡改侦测。此算法基于高斯模型建立背景模型，并用去背法算出差值当作篡改分数(moving_score)。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	图像
输出	float*	moving_score	篡改分数

4.2.11 活体识别

4.2.11.1 CVI_TDL_Liveness

【语法】

```
CVI_S32 CVI_TDL_Liveness(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *rgbFrame,␣
↪VIDEO_FRAME_INFO_S *irFrame, , cvtdl_face_t *rgb_faces, cvtdl_face_t *ir_faces);
```

【描述】

RGB, IR 双目活体识别。

判断 rgb_faces 和 ir_faces 中的人脸是否为活体。

活体分数置于 rgb_face->info[i].liveness_score 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	rgbFrame	RGB 图像
输入	VIDEO_FRAME_INFO_S*	irFrame	IR 图像
输入/输出	cvtdl_face_t*	rgb_meta	侦测到的 RGB 人脸/ 活体分数
输入	cvtdl_face_t*	ir_meta	侦测到的 IR 人脸

4.2.11.2 CVI_TDL_IrLiveness

【语法】

```
CVI_S32 CVI_TDL_IrLiveness(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *irFrame,
↪cvtdl_face_t *ir_faces);
```

【描述】

IR 單目活体识别。

判断 ir_faces 中的人脸是否为活体。

活体分数置于 ir_faces->info[i].liveness_score 中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	irFrame	IR 图像
输入/输出	cvtdl_face_t*	ir_faces	侦测到的 IR 人脸/活体分数

4.2.12 姿态检测

4.2.12.1 CVI_TDL_AlphaPose

【语法】

```
CVI_S32 CVI_TDL_AlphaPose(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪object_t *obj);
```

【描述】

使用 alphapose 模型进行推理，预测 17 个骨骼关键点。

检测结果置于 obj->info[i].pedestrian_properity->pose_17。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_object_t*	obj	侦测到的人 / 17 个骨骼关键点坐标

4.2.13 语义分割

4.2.13.1 CVI_TDL_DeeplabV3

【语法】

```
CVI_S32 CVI_TDL_DeeplabV3(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪VIDEO_FRAME_INFO_S *out_frame, cvtdl_class_filter_t *filter);
```

【描述】

使用 DeepLab V3 模型进行语义分割。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输出	VIDEO_FRAME_INFO_S*	out_frame	输出图像
输入	cvtdl_class_filter_t*	filter	保留的类别

4.2.14 跌倒检测

4.2.14.1 CVI_TDL_Fall

【语法】

```
CVI_S32 CVI_TDL_Fall(cvitdl_handle_t handle, cvtdl_object_t *obj);
```

【描述】

使用对象侦测与姿态检测之结果，判断跌倒状态。

在运行此 API 前需要先调用CVI_TDL_AlphaPose 取得人体关键点。

跌倒检测结果置于 obj->info[i].pedestrian_properity->fall 。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	cvtdl_object_t*	obj	跌倒状态结果

4.2.15 驾驶疲劳检测

4.2.15.1 CVI_TDL_FaceLandmarker

【语法】

```
CVI_S32 CVI_TDL_FaceLandmarker(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_face_t *faces);
```

【描述】

需先使用人脸检测，产生出 106 个人脸特征点检测的结果，将结果放入 face->dms[i].landmarks_106 并且更新 5 个人脸特征点 face->dms[i].landmarks_5。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	face	人脸

4.2.15.2 CVI_TDL_EyeClassification

【语法】

```
CVI_S32 CVI_TDL_EyeClassification (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_face_t *faces);
```

【描述】

需先使用人脸检测以及人脸特征点检测的结果，判断眼睛闭合状态，将结果放入 face->dms[i].reye_score/ face->dms[i].leye_score。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	face	人脸

4.2.15.3 CVI_TDL_YawnClassification

【语法】

```
CVI_S32 CVI_TDL_YawnClassification (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_face_t *faces);
```

【描述】

根据人脸检测和人脸关键点结果进行打哈欠检测。必须先调用 CVI_FaceRecognition 取得人脸检测和人脸关键点结果。打哈欠结果会放入 face->dms[i].yawn_score 。分数为 0.0~1.0 间。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	face	人脸

4.2.15.4 CVI_TDL_IncarObjectDetection

【语法】

```
CVI_S32 CVI_TDL_IncarObjectDetection(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S␣
↪*frame, cvtdl_face_t *faces);
```

【描述】

使用对象侦测检测对象（水杯 / 马克杯 / 电话）是否出现在驾驶周边，将判断结果输出成 object 格式，放入到 face->dms[i].dms.dms_od。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	cvtdl_face_t*	face	人脸

4.2.16 声音分类

4.2.16.1 CVI_TDL_SoundClassification

【语法】

```
CVI_S32 CVI_TDL_SoundClassification(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,
↪ int *index);
```

【描述】

判断 frame 中音讯属于哪个类别。并将各类别分数排序后输出。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	frame	输入图像
输入	int*	index	每个类别的分数

4.2.16.2 CVI_TDL_Get_SoundClassification_ClassesNum

【语法】

```
CVI_S32 CVI_TDL_Get_SoundClassification_ClassesNum(cvitdl_handle_t handle);
```

【描述】

取得音讯类别数量。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄

【输出】

	数据型态	说明
输出	int	类别数量

4.2.16.3 CVI_TDL_Set_SoundClassification_Threshold

【语法】

```
CVI_S32 CVI_TDL_Set_SoundClassification_Threshold(cvitdl_handle_t handle, const float th);
```

【描述】

设定音讯类别阀值。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_handle_t	handle	句柄
输入	const float	th	相似度阀值，高于此阀值之相似度才会取出

4.3 CVI_TDL_Service

4.3.1 通用

4.3.1.1 CVI_TDL_Service_CreateHandle

【语法】

```
CVI_S32 CVI_TDL_Service_CreateHandle(cvitdl_service_handle_t *handle, cvtdl_handle tdl_
↪handle);
```

【描述】

创建 Service 句柄

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t*	handle	句柄
输入	cvitdl_handle_t	tdl_handle	cvi_tdl_core 句柄

4.3.1.2 CVI_TDL_Service_DestroyHandle

【语法】

```
CVI_S32 CVI_TDL_Service_DestroyHandle(cvitdl_service_handle_t *handle);
```

【描述】

销毁 Service 句柄

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t*	handle	句柄

4.3.1.3 CVI_TDL_Service_Polygon_SetTarget

【语法】

```
CVI_S32 CVI_TDL_Service_Polygon_SetTarget(cvitdl_service_handle_t handle, const cvtdl_pts_
↪t *pts);
```

【描述】

设定区域入侵范围。pts 为凸多边形点坐标，顺序需为顺直针或逆时针。

调用CVI_TDL_Service_Polygon_Intersect 判断一个 bounding box 是否侵入已划定范围。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	cvtdl_pts_t*	pts	凸多边形点

4.3.1.4 CVI_TDL_Service_Polygon_Intersect

【语法】

```
CVI_S32 CVI_TDL_Service_Polygon_Intersect(cvitdl_service_handle_t handle, const cvtdl_bbox_
↪t *bbox, bool *has_intersect);
```

【描述】

根据 CVI_TDL_Service_Polygon_SetTarget 所设定区域入侵范围。判断给定之 gounding box 侵入范围。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	cvtdl_bbox_t*	bbox	Bounding box
输出	bool*	ha s_intersect	是否入侵

4.3.1.5 CVI_TDL_Service_RegisterFeatureArray

【语法】

```
CVI_S32 CVI_TDL_Service_RegisterFeatureArray(cvitdl_service_handle_t handle, const cvtdl_
↪service_feature_array_t featureArray, const cvtdl_service_feature_matching_e method);
```

【描述】

注册特征库，将 featureArray 中所含特征进行预计算并搬入内存中。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const cvtdl_service_feature_array_t	feature-Array	特征数组结构
输入	const cvtdl_service_feature_matching_e	method	比对方法，目前仅支持COS_SIMILARITY

4.3.1.6 CVI_TDL_Service_CalculateSimilarity

【语法】

```
CVI_S32 CVI_TDL_Service_CalculateSimilarity(cvitdl_service_handle_t handle, const cvtdl_
↪feature_t *feature_rhs, const cvtdl_feature_t *feature_lhs, float *score);
```

【描述】

使用处理器计算两个特征之 Cosine Similarity。其计算公式如下：

$$sim(\theta) = \frac{A \bullet B}{\|A\| \bullet \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

其中 n 为特征长度。目前仅支持 INT8 特征

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const cvtdl_feature_t*	feature_rhs	第一个特征
输入	const cvtdl_feature_t*	feature_lhs	第二个特征
输出	float*	score	相似度

4.3.1.7 CVI_TDL_Service_ObjectInfoMatching

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectInfoMatching(cvitdl_service_handle_t handle, const cvtdl_
↪object_info_t *object_info, const uint32_t topk, float threshold, uint32_t *indices, float *sims,␣
↪uint32_t *size);
```

【描述】

计算 object_info 中的对象特征和已注册之对象特征库之 Cosine Similarity。并取出大于 threshold 的 Top-K 个相似度。其计算公式如下：

$$sim(\theta) = \frac{A \bullet B}{\|A\| \bullet \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

其中 n 为特征长度。若特征库数量少于 1000 笔会以处理器进行计算，否则会以启动 TPU 进行计算。注册特征需要调用 CVI_TDL_Service_RegisterFeatureArray。目前仅支持 INT8 特征

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const cvtdl_object_info_t*	object_info	物件 Info
输入	const uint32_t	topk	取 topk 个相似度
输出	float	threshold	相似度阀值，高于此阀值之相似度才会取出
输出	uint32_t*	indices	符合条件之相似度在库内的 Index
输出	float*	sims	符合条件之相似度
输出	uint32_t*	size	最终取出的相似度个数

4.3.1.8 CVI_TDL_Service_FaceInfoMatching

【语法】

```
CVI_S32 CVI_TDL_Service_FaceInfoMatching(cvitdl_service_handle_t handle, const cvtdl_face_
↪info_t *face_info, const uint32_t topk, float threshold, uint32_t *indices, float *sims, uint32_t␣
↪*size);
```

【描述】

计算 face_info 中的人脸特征和已注册之人脸特征库之 Cosine Similarity。并取出大于 threshold 的 Top-K 个相似度。其计算公式如下：

$$sim(\theta) = \frac{A \bullet B}{\|A\| \bullet \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

其中 n 为特征长度。若特征库数量少于 1000 笔会以处理器进行计算，否则会以启动 TPU 进行计算。注册特征需要调用 CVI_TDL_Service_RegisterFeatureArray。目前仅支持 INT8 特征

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const cvtdl_face_info_t*	face_info	Face info
输入	const uint32_t	topk	取 topk 个相似度
输出	float	threshold	相似度阀值，高于此阀值之相似度才会取出
输出	uint32_t*	indices	符合条件之相似度在库内的 Index
输出	float*	sims	符合条件之相似度
输出	uint32_t*	size	最终取出的相似度个数

4.3.1.9 CVI_TDL_Service_RawMatching

【语法】

```
CVI_S32 CVI_TDL_Service_RawMatching(cvitdl_service_handle_t handle, const void *feature,␣
↪const feature_type_e type, const uint32_t topk, float threshold, uint32_t *indices, float *scores,␣
↪uint32_t *size);
```

【描述】

计算特征和已注册之特征库之 Cosine Similarity。并取出大于 threshold 的 Top-K 个相似度。其计算公式如下：

$$sim(\theta) = \frac{A \bullet B}{\|A\| \bullet \|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

其中 n 为特征长度。若特征库数量少于 1000 笔会以处理器进行计算，否则会以启动 TPU 进行计算。注册特征需要调用 CVI_TDL_Service_RegisterFeatureArray。和 CVI_TDL_Service_FaceInfoMatching 及 CVI_TDL_Service_ObjectInfoM atching 不同的是，

此 API 直接使用特征数组进行比对，不需传入 cvtdl_face_info_t 或 cvtdl_object_info_t。此 API 限制特征类型需要和特征库之特征类型相同。目前仅支持 INT8 特征

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const void*	feature	特征数组
输入	const feature_type_e	type	特征类型，目前仅支持 TYPE_INT8
输入	const uint32_t	topk	取 topk 个相似度
输出	float	threshold	相似度阀值，高于此阀值之相似度才会取出
输出	uint32_t*	indices	符合条件之相似度在库内的 Index
输出	float*	scores	符合条件之相似度
输出	uint32_t*	size	最终取出的相似度个数

4.3.1.10 CVI_TDL_Service_FaceAngle

【语法】

```
CVI_S32 CVI_TDL_Service_FaceAngle(const cvtdl_pts_t *pts, cvtdl_head_pose_t *hp);
```

【描述】

计算单个人脸姿态

【参数】

	数据型态	参数名称	说明
输入	cvtdl_pts_t*	pts	人脸 landmark
输出	cvtdl_head_pose_t*	hp	人脸姿态

4.3.1.11 CVI_TDL_Service_FaceAngleForAll

【语法】

```
CVI_S32 CVI_TDL_Service_FaceAngleForAll(const cvtdl_face_t *meta);
```

【描述】

计算多个人脸姿态

【参数】

	数据型态	参数名称	说明
输入	cvtdl_face_t*	meta	人脸资料

4.3.2 图像缩放

4.3.2.1 CVI_TDL_Service_FaceDigitalZoom

【语法】

```
CVI_S32 CVI_TDL_Service_FaceDigitalZoom(

cvitdl_service_handle_t handle,

const VIDEO_FRAME_INFO_S *inFrame,

const cvtdl_face_t *meta,

const float face_skip_ratio,

const float trans_ratio,

const float padding_ratio,

VIDEO_FRAME_INFO_S *outFrame);
```

【描述】

将人脸侦测结果之人脸进行放大 (zoom in)

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	VIDEO_FRAME_INFO_S*	inFrame	输入图像
输入	cvtdl_face_t*	meta	人脸资料
输入	float	face_skip_ratio	忽略比率
输入	float	trans_ratio	放大比率
输入	float	padding_ratio	扩展 bounding box 比例
输出	VIDEO_FRAME_INFO_S*	outFrame	输出图像

4.3.2.2 CVI_TDL_Service_ObjectDigitalZoom

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectDigitalZoom(cvitdl_service_handle_t handle,

const VIDEO_FRAME_INFO_S *inFrame, const cvtdl_object_t *meta, const float obj_skip_ratio,␣
↪const float trans_ratio, const float padding_ratio,

VIDEO_FRAME_INFO_S *outFrame);
```

【描述】

将对象侦测结果之对象进行放大 (zoom in)

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const VIDEO_FRAME_INFO_S*	inFrame	输入图像
输入	const cvtdl_object_t*	meta	对象数据
输入	const float	obj_skip_ratio	忽略比率
输入	const float	trans_ratio	放大比率
输入	const float	padding_ratio	扩展 bounding box 比例
输出	VIDEO_FRAME_INFO_S*	outFrame	输出图像

4.3.2.3 CVI_TDL_Service_ObjectDitgitalZoomExt

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectDigitalZoomExt(cvitdl_service_handle_t handle, const VIDEO_
↪FRAME_INFO_S *inFrame, const cvtdl_object_t *meta,

const float obj_skip_ratio, const float trans_ratio, const float pad_ratio_left, const float pad_ratio_
↪right, const float pad_ratio_top,

const float pad_ratio_bottom, VIDEO_FRAME_INFO_S *outFrame);
```

【描述】

将对象侦测结果之对象进行放大 (zoom in)

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const VIDEO_FRAME_INFO_S*	inFrame	输入图像
输入	const cvtdl_object_t*	meta	对象数据
输入	const float	obj_skip_ratio	忽略比率
输入	const float	trans_ratio	放大比率
输入	const float	pad_ratio_left	扩张率 (左)
输入	const float	pad_ratio_right	扩张率 (右)
输入	const float	pad_ratio_top	扩张率 (上)
输入	const float	pad_ratio_bottom	扩张率 (下)
输出	VIDEO_FRAME_INFO_S*	outFrame	输出图像

4.3.3 图像绘制

4.3.3.1 CVI_TDL_Service_FaceDrawPts

【语法】

```
CVI_S32 CVI_TDL_Service_FaceDrawPts(cvtdl_pts_t *pts, VIDEO_FRAME_INFO_S *frame);
```

【描述】

绘制人脸 landmark

【参数】

	数据型态	参数名称	说明
输入	cvtdl_pts_t*	pts	人脸 landmark
输入	VIDEO_FRAME_INFO_S*	hp	输入/输出图像

4.3.3.2 CVI_TDL_Service_FaceDrawRect

【语法】

```
CVI_S32 CVI_TDL_Service_FaceDrawRect(cvitdl_service_handle_t handle, const cvtdl_face_t␣
↪*meta, VIDEO_FRAME_INFO_S *frame, const bool drawText, cvtdl_service_brush_t brush);
```

【描述】

绘制人脸方框

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	cvtdl_face_t*	meta	人脸资料
输入	VIDEO_FRAME_INFO_S*	frame	输入/输出图像
输入	bool	drawText	是否绘制人脸名字
输入	cvtdl_service_brush_t	brush	颜色

4.3.3.3 CVI_TDL_Service_ObjectDrawPose

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectDrawPose(const cvtdl_object_t *meta, VIDEO_FRAME_
↪INFO_S *frame);
```

【描述】

绘制姿态侦测之 17 个骨骼点

【参数】

	数据型态	参数名称	说明
输入	const cvtdl_object_t*	meta	骨骼点侦测结果
输入	VIDEO_FRAME_INFO_S*	frame	输入图像

4.3.3.4 CVI_TDL_Service_ObjectDrawRect

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectDrawRect(cvitdl_service_handle_t handle, const cvtdl_object_
↪t *meta, VIDEO_FRAME_INFO_S *frame, const bool drawText);
```

【描述】

绘制对象侦测框

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	句柄
输入	const cvtdl_object_t*	meta	对象侦测结果
输入	VIDEO_FRAME_INFO_S*	frame	输入/输出图像
输入	const bool	drawText	是否绘制类别文字

4.3.3.5 CVI_TDL_Service_ObjectWriteText

【语法】

```
CVI_S32 CVI_TDL_Service_ObjectWriteText(char *name, int x, int y, VIDEO_FRAME_INFO_S␣
↪*frame, float r, float g, float b);
```

【描述】

绘制指定文字

【参数】

	数据型态	参数名称	说明
输入	char*	name	绘制的文字
输入	int	x	绘制的 x 坐标
输入	int	y	绘制的 y 坐标
输入/输出	IDEO_FRAME_INFO_S*	rame	输入/输出图像
输入	float	r	绘制颜色 r channel 值
输入	float	g	绘制颜色 g channel 值
输入	float	b	绘制颜色 b channel 值

4.3.3.6 CVI_TDL_Service_Incar_ObjectDrawRect

【语法】

```
CVI_S32 CVI_TDL_Service_Incar_ObjectDrawRect(cvitdl_service_handle_t handle, const cvtdl_
↪dms_od_t *meta, VIDEO_FRAME_INFO_S *frame, const bool drawText, cvtdl_service_brush_
↪t brush);
```

【描述】

Draw specified text

【参数】

	数据型态	参数名称	说明
输入	cvitdl_service_handle_t	handle	handle
输入	const cvtdl_dms_od_t*	meta	侦测结果输入
输入/输出	IDEO_FRAME_INFO_S*	rame	绘制的图片
输入	const bool	drawText	是否绘制类文字
输入	cvtdl_service_brush_t	brush	颜色

5 应用 (APP)

5.1 目的

CviTek TDL application，APP 是基于 TDL SDK，并针对不同客户端应用，所设计的 solution。

APP 整合 TDL SDK，提供客户更方便的操作 API。

APP code 为 open source，可以作为客户端开发的参考。

【编译】

1. 下载 TDL SDK 与其依赖之 SDK：MW、TPU、IVE。
2. 移动至 TDL SDK 的 module/app 目录
3. 执行以下指令：

```
make MW_PATH=<MW_SDK> TPU_PATH=<TPU_SDK> IVE_PATH=<IVE_SDK>

make install

make clean
```

编译完成的 lib 会放在 TDL SDK 的 tmp_install 目录下

5.2 API

5.2.1 句柄

【语法】

```
typedef struct {

cvitdl_handle_t tdl_handle;

IVE_HANDLE ive_handle;

face_capture_t *face_cpt_info;

```

(下页继续)

(续上页)

```
} cvitdl_app_context_t;

typedef cvitdl_app_context *cvitdl_app_handle_t;
```

【描述】

cvitdl_app_handle_t 为 cvitdl_app_context 的指针型态，其中包含 tdl handle、ive handle 与其他应用之数据结构。

5.2.1.1 CVI_TDL_APP_CreateHandle

【语法】

```
CVI_S32 CVI_TDL_APP_CreateHandle(cvitdl_app_handle_t *handle, cvitdl_handle_t tdl_
↪handle);
```

【描述】

创建使用 APP 所需的指标。需输入 tdl handle。

【参数】

	数据型态	参数名称	说明
输出	cvitdl_app_handle_t*	handle	输入句柄指标
输入	cvitdl_handle_t	a i_handle	TDL 句柄

5.2.1.2 CVI_TDL_APP_DestroyHandle

【语法】

```
CVI_S32 CVI_TDL_APP_DestroyHandle(cvitdl_app_handle_t handle);
```

【描述】

销毁创造的句柄 cvitdl_app_handle_t。

只会销毁个别应用程序所使用之数据，不影响 tdl handle。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标

5.2.2 人脸抓拍

人脸抓拍 (Face Capture) 结合人脸侦测、多对象追踪、人脸质量检测，功能为抓拍 (或截取) 影像中不同人的脸部照片。

抓拍条件可利用配置文件来调整，例如：抓拍时间点、人脸质量检测算法、人脸角度阀值…。

【配置文件】

参数名称	默认值	说明
Miss_Time_Limit	40	人脸遗失时间限制。当 APP 连续无法追踪到某个 face，会判定此 face 已离开。 [单位：frame]
Threshold_Size_Min	32	最小/最大可接受人脸大小，如果 face bbox 的任一边小于/大于此阀值，quality 会强制设为 0。
Threshold_Size_Max	512	
Quality_Assessment_Method	0	若人脸评估不使用 FQNet 时，启用内建质量检测算法 0: 基于人脸大小与角度 1: 基于眼睛距离
Threshold_Quality	0.1	人脸质量阀值，若新的 face 的 quality 大于此阀值，且比当前截取之 face 的 quality 还高，则会截取并更新暂存区 face 数据。
Threshold_Quality_High	0.95	人脸质量阀值(高),若暂存区某 face 的 quality 高于此阀值，则判定此 face 为高质量，后续不会再进行更新。 （仅适用于 level 2,3）
Threshold_Yaw	0.25	人脸角度阀值，若角度大于此阀值，quality 会强制设为 0。 （一单位为 90 度）
Threshold_Pitch	0.25	
Threshold_Roll	0.25	
FAST_Mode_Interval	10	FAST 模式抓拍间隔。 [单位：frame]
FAST_Mode_Capture_Num	3	FAST 模式抓拍次数。
CYCLE_Mode_Interval	20	CYCLE 模式抓拍间隔。 [单位：frame]
AUTO_Mode_Time_Limit	0	AUTO 模式最后输出的时限。 [单位：frame]
AUTO_Mode_Fast_Cap	1	AUTO 模式是否输出进行快速抓拍 1 次。
Capture_Aligned_Face	0	抓拍/截取人脸是否进行校正。

【人脸品质检测算法】

#	算法	计算方式
0	基于人脸大小与角度	1. Face Area Score 1. 定义标准人脸大小 A_base = 112 * 112 2. 计算侦测到的人脸面积 A_face = 长 * 宽 3. 计算 MIN(1.0, A_face/A_base) 作为分数 2. Face Pose Score 1. 分别计算人脸角度 yaw, pitch, roll 并取其绝对值 2. 计算 1 - (yaw + pitch + roll) / 3 作为分数 3. Face Quality = Face Area Score * Face Pose Score
1	基于眼睛距离	1. 定义标准瞳距 D = 80 2. 计算双眼距离 d 3. 计算 MIN(1.0, d/D) 当作分数

5.2.2.1 CVI_TDL_APP_FaceCapture_Init

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_Init(const cvitdl_app_handle_t handle, uint32_t buffer_
↪size);
```

【描述】

初始化人脸抓拍数据结构。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	uint32_t	buffer_size	人脸暂存区大小

5.2.2.2 CVI_TDL_APP_FaceCapture_QuickSetUp

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_QuickSetUp(const cvitdl_app_handle_t handle, int fd_
↪model_id, int fr_model_id, const char *fd_model_path, const char *fr_model_path, const char␣
↪*fq_model_path, const char *fl_model_path);
```

【描述】

快速设定人脸抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	int	fd_model_id	人脸侦测模型 ID
输入	int	fr_model_id	人脸识别检测模型 ID
输入	const char*	fd_model_path	人脸侦测模型路径
输入	const char*	fr_model_path	人脸识别检测模型路径
输入	const char*	fq_model_path	人脸质量检测模型路径
输入	const char*	fl_model_path	人脸座标检测模型路径

5.2.2.3 CVI_TDL_APP_FaceCapture_FusePedSetup

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_FusePedSetup(const cvitdl_app_handle_t handle, int ped_
↪model_id, const char *ped_model_path);
```

【描述】

快速设定人脸抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	int	fd_model_id	行人侦测模型 ID
输入	const char*	fl_model_path	行人侦测模型路径

5.2.2.4 CVI_TDL_APP_FaceCapture_GetDefaultConfig

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_GetDefaultConfig(face_capture_config_t *cfg);
```

【描述】

取得人脸抓拍预设参数。

【参数】

	数据型态	参数名称	说明
输出	face_capture_config_t*	cfg	人脸抓拍参数

5.2.2.5 CVI_TDL_APP_FaceCapture_SetConfig

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_SetConfig(const cvitdl_app_handle_t handle, face_
↪capture_config_t *cfg);
```

【描述】

设定人脸抓拍参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	face_capture_config_t*	cfg	人脸抓拍参数

5.2.2.6 CVI_TDL_APP_FaceCapture_FDFR

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_FDFR(const cvitdl_app_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame, cvtdl_face_t *p_face);
```

【描述】

设定人脸抓拍参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	VIDEO_FRAME_INFO_S*	frame	图像
输入	cvtdl_face_t*	p_face	人脸抓拍输出结果

5.2.2.7 CVI_TDL_APP_FaceCapture_SetMode

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_SetMode(const cvitdl_app_handle_t handle, capture_
↪mode_e mode);
```

【描述】

设定人脸抓拍模式。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	capture_mode_e	mode	人脸抓拍模式

5.2.2.8 CVI_TDL_APP_FaceCapture_Run

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_Run(const cvitdl_app_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame);
```

【描述】

执行人脸抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	VIDEO_FRAME_INFO_S*	frame	输入影像

5.2.2.9 CVI_TDL_APP_FaceCapture_CleanAll

【语法】

```
CVI_S32 CVI_TDL_APP_FaceCapture_CleanAll(const cvitdl_app_handle_t handle);
```

【描述】

清除所有人脸抓拍暂存区之数据数据。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标

5.2.3 人型抓拍

人型抓拍 (Face Capture) 结合人型侦测、多对象追踪、人脸质量检测，功能为抓拍 (或截取) 影像中不同人的脸部照片。

抓拍条件可利用配置文件来调整，例如：抓拍时间点、人脸质量检测算法、人脸角度阀值…。

【配置文件】

参数名称	说明	说明
Miss_Time_Limit	40	人脸遗失时间限制。当 APP 连续无法追踪到某个 face，会判定此 face 已离开。[单位：frame]
Threshold_Size_Min	32	最小/最大可接受人脸大小，如果 face bbox 的任一边小于 /大于此阀值，quality 会强制设为 0。
Threshold_Size_Max	512	
Quality_Assessment_Method	0	若人脸评估不使用 FQNet 时，启用内建质量检测算法 0: 基于人脸大小与角度 1: 基于眼睛距离
Threshold_Quality	0.1	人脸质量阀值，若新的 face 的 quality 大于此阀值，且比当前截取之 face 的 quality 还高，则会截取并更新暂存区 face 数据。
Threshold_Quality_High	0.95	人脸质量阀值(高),若暂存区某 face 的 quality 高于此阀值，则判定此 face 为高质量，后续不会再进行更新。（仅适用于 level 2,3）
Threshold_Yaw	0.25	人脸角度阀值，若角度大于此阀值，qua lity 会强制设为 0。（一单位为 90 度）
Threshold_Pitch	0.25	
Threshold_Roll	0.25	
FAST_Mode_Interval	10	FAST 模式抓拍间隔。[单位：frame]
FAST_Mode_Capture_Num	3	FAST 模式抓拍次数。
CYCLE_Mode_Interval	20	CYCLE 模式抓拍间隔。[单位：frame]
AUTO_Mode_Time_Limit	0	AUTO 模式最后输出的时限。[单位：frame]
AUTO_Mode_Fast_Cap	1	AUTO 模式是否输出进行快速抓拍 1 次。
Capture_Aligned_Face	0	抓拍/截取人脸是否进行校正。

【人脸品质检测算法】

#	算法	计算方式
0	基于人脸大小与角度	1. Face Area Score 1. 定义标准人脸大小 A_base = 112 * 112 2. 计算侦测到的人脸面积 A_face = 长 * 宽 3. 计算 MIN(1.0, A_face/A_base) 作为分数 2. Face Pose Score 1. 分别计算人脸角度 yaw, pitch, roll 并取其绝对值 2. 计算 1 - (yaw + pitch + roll) / 3 作为分数 3. Face Quality = Face Area Score * Face Pose Score
1	基于眼睛距离	1. 定义标准瞳距 D = 80 2. 计算双眼距离 d 3. 计算 MIN(1.0, d/D) 当作分数

5.2.3.1 CVI_TDL_APP_PersonCapture_Init

【语法】

```
CVI_TDL_APP_PersonCapture_Init(const cvitdl_app_handle_t handle, uint32_t buffer_size);
```

【描述】

初始化人形抓拍数据结构。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	uint32_t	buffer_size	人脸暂存区大小

5.2.3.2 CVI_TDL_APP_PersonCapture_QuickSetUp

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_QuickSetUp(const cvitdl_app_handle_t handle,
const char *od_model_name,
const char *od_model_path,
const char *reid_model_path);
```

【描述】

快速设定人型抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	const char*	od_model_name	人型侦测模型名称
输入	const char*	od_model_path	人型侦测模型路径
输入	const char*	reid_model_path	ReID 模型路径

5.2.3.3 CVI_TDL_APP_FaceCapture_GetDefaultConfig

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_GetDefaultConfig(person_capture_config_t *cfg);
```

【描述】

取得人型抓拍预设参数。

【参数】

	数据型态	参数名称	说明
输出	person_capture_config_t*	cfg	人型抓拍参数

5.2.3.4 CVI_TDL_APP_PersonCapture_SetConfig

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_SetConfig(const cvitdl_app_handle_t handle, person_
↪capture_config_t *cfg);
```

【描述】

设定人型抓拍参数。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	person_capture_config_t*	cfg	人型抓拍参数

5.2.3.5 CVI_TDL_APP_PersonCapture_SetMode

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_SetMode(const cvitdl_app_handle_t handle, capture_
↪mode_e mode);
```

【描述】

设定人型抓拍模式。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	capture_mode_e	mode	人型抓拍模式

5.2.3.6 CVI_TDL_APP_PersonCapture_Run

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_Run(const cvitdl_app_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame);
```

【描述】

执行人型抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	VIDEO_FRAME_INFO_S*	frame	输入影像

5.2.3.7 CVI_TDL_APP_ConsumerCounting_Run

【语法】

```
CVI_S32 CVI_TDL_APP_ConsumerCounting_Run(const cvitdl_app_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame);
```

【描述】

执行人型抓拍。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标
输入	VIDEO_FRAME_INFO_S*	frame	输入影像

5.2.3.8 CVI_TDL_APP_PersonCapture_CleanAll

【语法】

```
CVI_S32 CVI_TDL_APP_PersonCapture_ClanAll(const cvitdl_app_handle_t handle);
```

【描述】

清除所有人型抓拍暂存区之数据数据。

【参数】

	数据型态	参数名称	说明
输入	cvitdl_app_handle_t	handle	输入句柄指标

6 数据类型

6.1 CVI_TDL_Core

6.1.1 CVI_TDL_SUPPORTED_MODEL_E

【描述】

此 enum 定义 TDL SDK 中所有 Deep Learning Model。下表为每个模型 Id 和其模型功能说明。

模型 Index	模型 ID	说明
0	CVI_TDL_SUPPORTED_MODEL_RETINAFACE	人脸侦测 (RetinaFace)
1	CVI_TDL_SUPPORTED_MODEL_RETINAFACE_IR	红外线人脸侦测 (RetinaFace)
2	CVI_TDL_SUPPORTED_MODEL_RETINAFACE_HARDHAT	安全帽人脸检测 (RetinaFace)
3	CVI_TDL_SUPPORTED_MODEL_SCRFDFACE	人脸侦测 (ScrFD Face)
4	CVI_TDL_SUPPORTED_MODEL_THERMALFACE	热显人脸侦测
5	CVI_TDL_SUPPORTED_MODEL_THERMALPERSON	热显人体侦测
6	CVI_TDL_SUPPORTED_MODEL_FACEATTRIBUTE	人脸属性和人脸识别
7	CVI_TDL_SUPPORTED_MODEL_FACERECOGNITION	人脸识别
8	CVI_TDL_SUPPORTED_MODEL_MASKFACERECOGNITION	戴口罩人脸识别
9	CVI_TDL_SUPPORTED_MODEL_FACEQUALITY	人脸质量
10	CVI_TDL_SUPPORTED_MODEL_MASKCLASSIFICATION	人脸口罩识别
11	CVI_TDL_SUPPORTED_MODEL_HANDCLASSIFICATION	手势识别
12	CVI_TDL_SUPPORTED_MODEL_HAND_KEYPOINT	手势关键点侦测

下页继续

表 6.1 – 续上页

模型 Index	模型 ID	说明
13	CVI_TDL_SUPPORTED_MODEL_HAND_KEYPOINT_CLASSIFICATION	手势关键点识别
14	CVI_TDL_SUPPORTED_MODEL_LIVENESS	双目活体识别
15	CVI_TDL_SUPPORTED_MODEL_HAND_DETECTION	手部侦测
16	CVI_TDL_SUPPORTED_MODEL_MOBILEDETV2_PERSON_VEHICLE	人形及交通工具侦测
17	CVI_TDL_SUPPORTED_MODEL_MOBILEDETV2_VEHICLE	交通工具侦测
18	CVI_TDL_SUPPORTED_MODEL_MOBILEDETV2_PEDESTRIAN	行人侦测
19	CVI_TDL_SUPPORTED_MODEL_MOBILEDETV2_PERSON_PETS	猫狗及人型侦测
20	CVI_TDL_SUPPORTED_MODEL_MOBILEDETV2_COCO80	80 类对象侦测
21	CVI_TDL_SUPPORTED_MODEL_YOLOV3	80 类对象侦测
22	CVI_TDL_SUPPORTED_MODEL_YOLOV5	80 类对象侦测
23	CVI_TDL_SUPPORTED_MODEL_YOLOX	80 类对象侦测
24	CVI_TDL_SUPPORTED_MODEL_OSNET	行人重识 - 0 别
25	CVI_TDL_SUPPORTED_MODEL_SOUNDCLASSIFICATION	声音识别
26	CVI_TDL_SUPPORTED_MODEL_SOUNDCLASSIFICATION_V2	声音识别 V2
27	CVI_TDL_SUPPORTED_MODEL_WPODNET	车牌侦测
28	CVI_TDL_SUPPORTED_MODEL_LPRNET_TW	台湾地区车牌识别
29	CVI_TDL_SUPPORTED_MODEL_LPRNET_CN	大陆地区车牌识别
30	CVI_TDL_SUPPORTED_MODEL_DEEPLABV3	语意分割
31	CVI_TDL_SUPPORTED_MODEL_ALPHAPOSE	人体关键点侦测
32	CVI_TDL_SUPPORTED_MODEL_EYECLASSIFICATION	闭眼识别
33	CVI_TDL_SUPPORTED_MODEL_YAWNCLASSIFICATION	打哈欠识别
34	CVI_TDL_SUPPORTED_MODEL_FACELANDMARKER	人脸关键点侦测
35	CVI_TDL_SUPPORTED_MODEL_FACELANDMARKERDET2	人脸关键点侦测 2

下页继续

表 6.1 – 续上页

模型 Index	模型 ID	说明
36	CVI_TDL_SUPPORTED_MODEL_INCAROBJECTDETECTION	车内对象识别
37	CVI_TDL_SUPPORTED_MODEL_SMOKECLASSIFICATION	抽烟识别
38	CVI_TDL_SUPPORTED_MODEL_FACEMASKDETECTION	口罩人脸侦测
39	CVI_TDL_SUPPORTED_MODEL_IRLIVENESS	红外线活体侦测
40	CVI_TDL_SUPPORTED_MODEL_PERSON_PETS_DETECTION	人形及猫狗侦测
41	CVI_TDL_SUPPORTED_MODEL_PERSON_VEHICLE_DETECTION	人形及车辆侦测
42	CVI_TDL_SUPPORTED_MODEL_HAND_FACE_PERSON_DETECTION	手部、脸及人型侦测
43	CVI_TDL_SUPPORTED_MODEL_HEAD_PERSON_DETECTION	手部及人型侦测
44	CVI_TDL_SUPPORTED_MODEL_YOLOV8POSE	姿态侦测
45	CVI_TDL_SUPPORTED_MODEL_SIMCC_POSE	姿态侦测
46	CVI_TDL_SUPPORTED_MODEL_LANDMARK_DET3	人脸关键点侦测

下表为每个模型 Id 对应的模型档案及推理使用的 function：

模型 Index	Inference Function	模型档案
0	CVI_TDL_RetinaFace	retinaface_mnet0.25_342_608.cvimodel retinaface_mnet0.25_608_342.cvimodel retinaface_mnet0.25_608.cvimodel
1	CVI_TDL_RetinaFace_IR	retinafaceIR_mnet0.25_342_608.cvimodel retinafaceIR_mnet0.25_608_342.cvimodel retinafaceIR_mnet0.25_608_608.cvimodel
2	CVI_TDL_RetinaFace_Hardhat	hardhat_720_1280.cvimodel
3	CVI_TDL_ScrFDFace	scrfd_320_256_ir.cvimodel scrfd_480_270_int8.cvimodel scrfd_480_360_int8.cvimodel scrfd_500m_bnkps_432_768.cvimodel scrfd_768_432_int8_1x.cvimodel
4	CVI_TDL_ThermalFace	thermalfd-v1.cvimodel
5	CVI_TDL_ThermalPerson	thermal_person_detection.cvimodel
6	CVI_TDL_FaceAttribute CVI_TDL_FaceAttributeOne	cviface-v3-attribute.cvimodel

下页继续

表 6.2 – 续上页

模型 Index	Inference Function	模型档案
7	CVI_TDL_FaceRecognition CVI_TDL_FaceRecognitionOne	cviface-v4.cvimodel cviface-v5-m.cvimodel cviface-v5-s.cvimodel cviface-v6-s.cvimodel
8	CVI_TDL_MaskFaceRecognition	masked-fr-v1-m.cvimodel
9	CVI_TDL_FaceQuality	fqnet-v5_shufflenetv2-softmax.cvimodel
10	CVI_TDL_MaskClassification	mask_classifier.cvimodel
11	CVI_TDL_HandClassification	hand_cls_128x128.cvimodel
12	CVI_TDL_HandKeypoint	hand_kpt_128x128.cvimodel
13	CVI_TDL_HandKeypointClassification	hand_kpt_cls9.cvimodel
14	CVI_TDL_Liveness	liveness-rgb-ir.cvimodel
15	CVI_TDL_Hand_Detection	hand_det_qat_640x384.cvimodel
16	CVI_TDL_MobileDetV2_Vehicle	mobiledetv2-vehicle-d0-ls.cvimodel
17	CVI_TDL_MobileDetV2_Pedestrian	mobiledetv2-pedestrian-d0-ls-384.cvimodel mobiledetv2-pedestrian-d0-ls-640.cvimodel mobiledetv2-pedestrian-d0-ls-768.cvimodel mobileDetV2-pedestrian-d1-ls.cvimodel mobiledetv2-pedestrian-d1-ls-1024.cvimodel
18	CVI_TDL_MobileDetV2_Person_Vehicle	mobiledetv2-person-vehicle-ls-768.cvimodel mobiledetv2-person-vehicle-ls.cvimodel
19	CVI_TDL_MobileDetV2_Person_Pets	mobiledetv2-lite-person-pets-ls.cvimodel
20	CVI_TDL_MobileDetV2_COCO80	mobiledetv2-d0-ls.cvimodel mobiledetv2-d1-ls.cvimodel mobiledetv2-d2-ls.cvimodel
21	CVI_TDL_Yolov3	yolo_v3_416.cvimodel
22	CVI_TDL_Yolov5	yolov5s_3_branch_int8.cvimodel
23	CVI_TDL_YoloX	yolox_nano.cvimodel yolox_tiny.cvimodel
24	CVI_TDL_OSNet CVI_TDL_OSNetOne	person-reid-v1.cvimodel
25	CVI_TDL_SoundClassification	es_classification.cvimodel soundcmd_bf16.cvimodel
26	CVI_TDL_SoundClassification_V2	c10_lightv2_mse40_mix.cvimodel
27	CVI_TDL_LicensePlateDetection	wpodnet_v0_bf16.cvimodel
28	CVI_TDL_LicensePlateRecognition_TW	lprnet_v0_tw_bf16.cvimodel
29	CVI_TDL_LicensePlateRecognition_CN	lprnet_v1_cn_bf16.cvimodel
30	CVI_TDL_DeeplabV3	deeplabv3_mobilenetv2_640x360.cvimodel
31	CVI_TDL_AlphaPose	alphapose.cvimodel
32	CVI_TDL_EyeClassification	eye_v1_bf16.cvimodel

下页继续

表 6.2 – 续上页

模型 Index	Inference Function	模型档案
33	CVI_TDL_YawnClassification	yawn_v1_bf16.cvimodel
34	CVI_TDL_FaceLandmarker	face_landmark_bf16.cvimodel
35	CVI_TDL_FaceLandmarkerDet2	pipnet_blurness_v5_64_retinaface_50ep.cvimodel
36	CVI_TDL_IncarObjectDetection	incar_od_v0_bf16.cvimodel
37	CVI_TDL_SmokeClassification	N/A
38	CVI_TDL_FaceMaskDetection	retinaface_yolox_fdmask.cvimodel
39	CVI_TDL_IrLiveness	liveness-rgb-ir.cvimodel liveness-rgb-ir-3d.cvimodel
40	CVI_TDL_PersonPet_Detection	pet_det_640x384.cvimodel
41	CVI_TDL_PersonVehicle_Detection	yolov8n_384_640_person_vehicle.cvimodel
42	CVI_TDL_HandFacePerson_Detection	meeting_det_640x384.cvimodel
43	CVI_TDL_HeadPerson_Detection	yolov8n_headperson.cvimodel
44	CVI_TDL_Yolov8_Pose	yolov8n_pose_384_640.cvimodel
45	CVI_TDL_Simcc_Pose	simcc_mv2_pose.cvimodel
46	CVI_TDL_FLDet3	onet_int8.cvimodel

6.1.2 cvtdl_obj_class_id_e

【描述】

此 enum 定义对象侦测类别。每一类别归属于一个类别群组。

类别	类别群组
CVI_TDL_DET_TYPE_PERSON	CVI_TDL_DET_GROUP_PERSON
CVI_TDL_DET_TYPE_BICYCLE	CVI_TDL_DET_GROUP_VEHICLE
CVI_TDL_DET_TYPE_CAR	
CVI_TDL_DET_TYPE_MOTORBIKE	
CVI_TDL_DET_TYPE_AEROPLANE	
CVI_TDL_DET_TYPE_BUS	
CVI_TDL_DET_TYPE_TRAIN	
CVI_TDL_DET_TYPE_TRUCK	
CVI_TDL_DET_TYPE_BOAT	
CVI_TDL_DET_TYPE_TRAFFIC_LIGHT	CVI_TDL_DET_GROUP_OUTDOOR
CVI_TDL_DET_TYPE_FIRE_HYDRANT	
CVI_TDL_DET_TYPE_STREET_SIGN	
CVI_TDL_DET_TYPE_STOP_SIGN	
CVI_TDL_DET_TYPE_PARKING_METER	
CVI_TDL_DET_TYPE_BENCH	
CVI_TDL_DET_TYPE_BIRD	CVI_TDL_DET_GROUP_ANIMAL
CVI_TDL_DET_TYPE_CAT	

下页继续

表 6.3 – 续上页

类别	类别群组
CVI_TDL_DET_TYPE_DOG	
CVI_TDL_DET_TYPE_HORSE	
CVI_TDL_DET_TYPE_SHEEP	
CVI_TDL_DET_TYPE_COW	
CVI_TDL_DET_TYPE_ELEPHANT	
CVI_TDL_DET_TYPE_BEAR	
CVI_TDL_DET_TYPE_ZEBRA	
CVI_TDL_DET_TYPE_GIRAFFE	
CVI_TDL_DET_TYPE_HAT	CVI_TDL_DET_GROUP _ACCESSORY
CVI_TDL_DET_TYPE_BACKPACK	
CVI_TDL_DET_TYPE_UMBRELLA	
CVI_TDL_DET_TYPE_SHOE	
CVI_TDL_DET_TYPE_EYE_GLASSES	
CVI_TDL_DET_TYPE_HANDBAG	
CVI_TDL_DET_TYPE_TIE	
CVI_TDL_DET_TYPE_SUITCASE	
CVI_TDL_DET_TYPE_FRISBEE	CVI_TDL_DET_GROUP_SPORTS
CVI_TDL_DET_TYPE_SKIS	
CVI_TDL_DET_TYPE_SNOWBOARD	
CVI_TDL_DET_TYPE_SPORTS_BALL	
CVI_TDL_DET_TYPE_KITE	
CVI_TDL_DET_TYPE_BASEBALL_BAT	
CVI_TDL_DET_TYPE_BASEBALL_GLOVE	
CVI_TDL_DET_TYPE_SKATEBOARD	
CVI_TDL_DET_TYPE_SURFBOARD	
CVI_TDL_DET_TYPE_TENNIS_RACKET	
CVI_TDL_DET_TYPE_BOTTLE	CVI_TDL_DET_GROUP_KITCHEN
CVI_TDL_DET_TYPE_PLATE	
CVI_TDL_DET_TYPE_WINE_GLASS	
CVI_TDL_DET_TYPE_CUP	
CVI_TDL_DET_TYPE_FORK	
CVI_TDL_DET_TYPE_KNIFE	
CVI_TDL_DET_TYPE_SPOON	
CVI_TDL_DET_TYPE_BOWL	
CVI_TDL_DET_TYPE_BANANA	CVI_TDL_DET_GROUP_FOOD
CVI_TDL_DET_TYPE_APPLE	
CVI_TDL_DET_TYPE_SANDWICH	
CVI_TDL_DET_TYPE_ORANGE	
CVI_TDL_DET_TYPE_BROCCOLI	
CVI_TDL_DET_TYPE_CARROT	
CVI_TDL_DET_TYPE_HOT_DOG	
CVI_TDL_DET_TYPE_PIZZA	
CVI_TDL_DET_TYPE_DONUT	
CVI_TDL_DET_TYPE_CAKE	

下页继续

表 6.3 – 续上页

类别	类别群组
CVI_TDL_DET_TYPE_CHAIR	CVI_TDL_DET_GROUP_FURNITURE
CVI_TDL_DET_TYPE_SOFA	
CVI_TDL_DET_TYPE_POTTED_PLANT	
CVI_TDL_DET_TYPE_BED	
CVI_TDL_DET_TYPE_MIRROR	
CVI_TDL_DET_TYPE_DINING_TABLE	
CVI_TDL_DET_TYPE_WINDOW	
CVI_TDL_DET_TYPE_DESK	
CVI_TDL_DET_TYPE_TOILET	
CVI_TDL_DET_TYPE_DOOR	
CVI_TDL_DET_TYPE_TV_MONITOR	CVI_TDL_DET_GROUP_ELECTRONIC
CVI_TDL_DET_TYPE_LAPTOP	
CVI_TDL_DET_TYPE_MOUSE	
CVI_TDL_DET_TYPE_REMOTE	
CVI_TDL_DET_TYPE_KEYBOARD	
CVI_TDL_DET_TYPE_CELL_PHONE	
CVI_TDL_DET_TYPE_MICROWAVE	CVI_TDL_DET_GROUP_APPLIANCE
CVI_TDL_DET_TYPE_OVEN	
CVI_TDL_DET_TYPE_TOASTER	
CVI_TDL_DET_TYPE_SINK	
CVI_TDL_DET_TYPE_REFRIGERATOR	
CVI_TDL_DET_TYPE_BLENDER	
CVI_TDL_DET_TYPE_BOOK	CVI_TDL_DET_GROUP_INDOOR
CVI_TDL_DET_TYPE_CLOCK	
CVI_TDL_DET_TYPE_VASE	
CVI_TDL_DET_TYPE_SCISSORS	
CVI_TDL_DET_TYPE_TEDDY_BEAR	
CVI_TDL_DET_TYPE_HAIR_DRIER	
CVI_TDL_DET_TYPE_TOOTHBRUSH	
CVI_TDL_DET_TYPE_HAIR_BRUSH	

6.1.3 cvtdl_obj_det_group_type_e

【描述】

此 enum 定义对象类别群组。

类别群组	描述
CVI_TDL_DET_GROUP_ALL	全部类别
CVI_TDL_DET_GROUP_PERSON	人形
CVI_TDL_DET_GROUP_VEHICLE	交通工具
CVI_TDL_DET_GROUP_OUTDOOR	户外
CVI_TDL_DET_GROUP_ANIMAL	动物
CVI_TDL_DET_GROUP_ACCESSORY	配件
CVI_TDL_DET_GROUP_SPORTS	运动
CVI_TDL_DET_GROUP_KITCHEN	厨房
CVI_TDL_DET_GROUP_FOOD	食物
CVI_TDL_DET_GROUP_FURNITURE	家具
CVI_TDL_DET_GROUP_ELECTRONIC	电子设备
CVI_TDL_DET_GROUP_APPLIANCE	器具
CVI_TDL_DET_GROUP_INDOOR	室内用品
CVI_TDL_DET_GROUP_MASK_HEAD	自订类别
CVI_TDL_DET_GROUP_MASK_START	自订类别开始
CVI_TDL_DET_GROUP_MASK_END	自订类别结束

6.1.4 feature_type_e

【enum】

数值	参数名称	描述
0	TYPE_INT8	int8_t 特征类型
1	TYPE_UINT8	uint8_t 特征类型
2	TYPE_INT16	int16_t 特征类型
3	TYPE_UINT16	uint16_t 特征类型
4	TYPE_INT32	int32_t 特征类型
5	TYPE_UINT32	uint32_t 特征类型
6	TYPE_BF16	bf16 特征类型
7	TYPE_FLOAT	float 特征类型

6.1.5 meta_rescale_type_e

【enum】

数值	参数名称	描述
0	RESCALE_UNKNOWN	未知
1	RESCALE_NOASPECT	不依比例直接调整
2	RESCALE_CENTER	在四周进行 padding
3	RESCALE_RB	在右下进行 padding

6.1.6 cvtdl_bbox_t

数据类型	参数名称	描述
float	x1	侦测框左上点坐标之 x 值
float	y1	侦测框左上点坐标之 y 值
float	x2	侦测框右下点坐标之 x 值
float	y2	侦测框右下点坐标之 y 值
float	score	侦测框之信心程度

6.1.7 cvtdl_feature_t

数据类型	参数名称	描述
int8_t*	ptr	地址
uint32_t	size	特征维度
feature_type_e	type	特征型态

6.1.8 cvtdl_pts_t

数据类型	参数名称	描述
float*	x	坐标 x
float*	y	坐标 y
uint32_t	size	坐标点个数

6.1.9 cvtdl_4_pts_t

数据类型	参数名称	描述
float	x[4]	4 个坐标点之 x 坐标值
float	y[4]	4 个坐标点之 y 坐标值

6.1.10 cvtdl_vpssconfig_t

数据类型	参数名称	描述
VPSS_SCALE_COEF_E	chn_coeff	Rescale 方式
VPSS_CHN_ATTR_S	chn_attr	VPSS 属性数据

6.1.11 cvtdl_tracker_t

数据类型	参数名称	描述
uint32_t	size	追踪讯息数量
cvtdl_tracker_info_t*	info	追踪讯息结构

6.1.12 cvtdl_tracker_info_t

数据类型	参数名称	描述
cvtdl_trk_state_type_t	state	追踪状态
cvtdl_bbox_t	bbox	追踪预测之边界框

6.1.13 cvtdl_trk_state_type_t

【enum】

数值	参数名称	描述
0	CVI_TRACKER_NEW	追踪状态为新增
1	CVI_TRACKER_UNSTABLE	追踪状态为不稳定
2	CVI_TRACKER_STABLE	追踪状态为稳定

6.1.14 cvtdl_deepsort_config_t

数据类型	参数名称	描述
float	max_distance_iou	进行 BBox 匹配时最大 IOU 距离
float	ma x_distance_consine	进行 Feature 匹配时最大 consine 距离
int	max_unmatched_times_for_bbox_matching	参与 BBox 匹配的目标最大未匹配次数之数量
bool	enable_internal_FQ	启用内部特征品质
cvtdl_kalman_filter_config_t	kfilter_conf	Kalman Filter 设定
cvtdl_kalman_tracker_config_t	ktracker_conf	Kalman Tracker 设定

6.1.15 cvtdl_kalman_filter_config_t

数据类型	参数名称	描述
bool	enable_X_constraint_0	启用第 0 个 X 约束
bool	enable_X_constraint_1	启用第 1 个 X 约束
float	X_constraint_min[8]	X 约束下限
float	X_constraint_max[8]	X 约束上限
bool	enable_bounding_stay	保留边界
mahalanobis_confidence_e	confidence_level	马氏距离信心度
float	chi2_threshold	卡方阈值
float	Q_std_alpha[8]	Process Noise 参数
float	Q_std_beta[8]	Process Noise 参数
int	Q_std_x_idx[8]	Process Noise 参数
float	R_std_alpha[4]	Measurement Noise 参数
float	R_std_beta[4]	Measurement Noise 参数
int	R_std_x_idx[4]	Measurement Noise 参数

【描述】

对于追踪目标运动状态 X

Process Nose (运动偏差), Q, 其中

$Q[i] = \left(Alpha_Q[i] \bullet X\left[Idx_Q[i]\right] + Beta_Q[i]\right)^2$

Measurement Nose (量测偏差), R, 同理运动偏差公式

6.1.16 cvtdl_kalman_tracker_config_t

数据类型	参数名称	描述
int	max_unmatched_num	追踪目标最大遗失数
int	accreditation_threshold	追踪状态转为稳定之阀值
int	feature_budget_size	保存追踪目标 feature 之最大数量
int	feature_update_interval	更新 feature 之时间间距
bool	enable_QA_feature_init	启用 QA 特征初始化
bool	enable_QA_feature_update	启用 QA 特征更新
float	feature_init_quality_threshold	特征初始化品质阈值
float	feature_update_quality_threshold	特征更新品质阈值
float	P_std_alpha[8]	Initial Covariance 参数
float	P_std_beta[8]	Initial Covariance 参数
int	P_std_x_idx[8]	Initial Covariance 参数

【描述】

Initial Covariance (初始运动状态偏差), P, 同理运动偏差公式

6.1.17 cvtdl_liveness_ir_position_e

【enum】

数值	参数名称	描述
0	LIVENESS_IR_LEFT	IR 镜头在 RGB 镜头左侧
1	LIVENESS_IR_RIGHT	IR 镜头在 RGB 镜头右侧

6.1.18 cvtdl_head_pose_t

数据类型	参数名称	描述
float	yaw	偏摆角
float	pitch	俯仰角
float	roll	翻滚角
float	facialUnitNormalVector[3]	脸部之面向方位

6.1.19 cvtdl_face_info_t

数据类型	参数名称	描述
char	name[128]	人脸名
uint64_t	unique_id	人脸 ID
cvtdl_bbox_t	bbox	人脸侦测框
cvtdl_pts_t	pts	人脸特征点
cvtdl_feature_t	feature	人脸特征
cvtdl_face_emotion_e	emotion	表情
cvtdl_face_gender_e	gender	性别
cvtdl_face_race_e	race	种族
float	score	分数
float	age	年龄
float	liveness_score	活体机率值
float	hardhat_score	安全帽机率值
float	mask_score	人脸戴口罩机率值
float	recog_score	识别分数
float	face_quality	人脸品质
float	pose_score	姿势分数
float	pose_score1	姿势分数
float	sharpness_score	清晰度分数
float	blurness	模糊性
cvtdl_head_pose_t	head_pose	人脸角度信息
int	track_state	追踪状态

6.1.20 cvtdl_face_t

数据类型	参数名称	描述
uint32_t	size	人脸个数
uint32_t	width	原始图片之宽
uint32_t	height	原始图片之高
meta_rescale_type_e*	rescale_type	rescale 的形态
cvtdl_face_info_t*	info	人脸综合信息
cvtdl_dms_t*	dms	驾驶综合信息

6.1.21 cvtdl_pose17_meta_t

数据类型	参数名称	描述
float	x[17]	17 个骨骼关键点的 x 坐标
float	y[17]	17 个骨骼关键点的 y 坐标
float	score[17]	17 个骨骼关键点的预测信心值

6.1.22 cvtdl_vehicle_meta

数据类型	参数名称	描述
cvtdl_4_pts_t	license_pts	车牌 4 个角坐标
cvtdl_bbox_t	license_bbox	车牌边界框
char[125]	license_char	车牌号码

【描述】

车牌 4 个角坐标依序为左上、右上、右下至左下。

6.1.23 cvtdl_class_filter_t

数据类型	参数名称	描述
uint32_t*	preserved_class_ids	要保留的类别 id
uint32_t	num_preserved_classes	要保留的类别 id 个数

6.1.24 cvtdl_dms_t

数据类型	参数名称	描述
float	reye_score	右眼开合分数
float	leye_score	左眼开合分数
float	yawn_score	嘴巴闭合分数
float	phone_score	讲电话分数
float	smoke_score	抽烟分数
cvtdl_pts_t	landmarks_106	106 个特征点
cvtdl_pts_t	landmarks_5	5 个特征点
cvtdl_head_pose_t	head_pose	透过 106 个特征点算出来的人脸角度
cvtdl_dms_od_t	dms_od	车内的物件侦测结果

6.1.25 cvtdl_dms_od_t

数据类型	参数名称	描述
uint32_t	size	有几个物件
uint32_t	width	宽度
uint32_t	height	长度
meta_rescale_type_e	rescale_type	rescale 的形态
cvtdl_dms_od_info_t*	info	物件的资讯

6.1.26 cvtdl_dms_od_info_t

数据类型	参数名称	描述
char[128]	name	物体名称
int	classes	物体类别
cvtdl_bbox_t	bbox	物体边界框

6.1.27 cvtdl_face_emotion_e

【描述】

人脸表情

表情	描述
EMOTION_UNKNOWN	未知
EMOTION_HAPPY	高兴
EMOTION_SURPRISE	惊讶
EMOTION_FEAR	恐惧
EMOTION_DISGUST	厌恶
EMOTION_SAD	伤心
EMOTION_ANGER	生气
EMOTION_NEUTRAL	自然

6.1.28 cvtdl_face_race_e

种族	描述
RACE_UNKNOWN	未知
RACE_CAUCASIAN	高加索人
RACE_BLACK	黑人
RACE_ASIAN	亚洲人

6.1.29 cvtdl_pedestrian_meta

数据类型	参数名称	描述
cvtdl_pose17_meta_t	pose17	人体 17 关键点
bool	fall	受否跌倒

6.1.30 cvtdl_object_info_t

数据类型	参数名称	描述
char	name	对象类别名
uint64_t	unique_id	唯一 id
cvtdl_box_t	bbox	框的边界讯息
cvtdl_feature_t	feature	对象特征
int	classes	类别 ID
cvtdl_vehicle_meta	vehicle_property	车辆属性
cvtdl_pedestrian_meta	pedestrian_property	行人属性
int	track_state	追踪状态

6.1.31 cvtdl_object_t

数据类型	参数名称	描述
uint32_t	size	info 所含物件个数
uint32_t	width	原始图片之宽
uint32_t	height	原始图片之高
uint32_t	entry_num	entry 数量
uint32_t	miss_num	miss 数量
meta_rescale_type_e	rescale_type	模型前处理采用的 resize 方式
cvtdl_object_info_t*	info	物件信息

6.1.32 cvtdl_handpose21_meta_t

数据类型	参数名称	描述
float	xn[21]	归一化 x 点
float	x[21]	x 点
float	yn[21]	归一化 y 点
float	y[21]	y 点
float	bbox_x	框的 x 座标
float	bbox_y	框的 y 座标
float	bbox_w	框的宽
float	bbox_h	框的高
int	label	手势类别
float	score	手势分数

6.1.33 cvtdl_handpose21_meta_ts

数据类型	参数名称	描述
uint32_t	size	侦测到手的数量
uint32_t	width	图片宽
uint32_t	height	图片高
cvtdl_handpose21_meta_t*	info	手部关键点

6.1.34 Yolov5PreParam

数据类型	参数名称	描述
float	factor[3]	缩放因子
float	mean[3]	图像均值
meta_rescale_type_e	rescale_type	缩放模式
bool*	pad_reverse	反向填充
bool*	keep_aspect_ratio	保持宽高比例缩放
bool*	use_quantize_scale	量化缩放
bool*	use_crop	裁剪调整图像大小
VPSS_SCALE_COEF_E*	resize_method	缩放方法
PIXEL_FORMAT_E*	format	图像格式

6.1.35 YOLOV5AlgParam

数据类型	参数名称	描述
uint32_t	anchors[3][3][2]	模型錨點
float	conf_thresh	信心度阀值
float	nms_thresh	均方根阀值

6.2 CVI_TDL_Service

6.2.1 cvtdl_service_feature_matching_e

【描述】

特征比对计算方法，目前仅支持 Cosine Similarity。

【定义】

参数名称	描述
COS_SIMILARITY	Cosine similarity

6.2.2 cvtdl_service_feature_array_t

【描述】

特征数组，此结构包含了特征数组指针, 长度, 特征个数, 及特征类型等信息。在注册特征库时需要传入此结构。

【定义】

数据类型	参数名称	描述
int8_t*	ptr	特征数组指针
uint32_t	feature_length	单一特征长度
uint32_t	data_num	特征个数
feature_type_e	type	特征类型

6.2.3 cvtdl_service_brush_t

【描述】

绘图笔刷结构，可指定欲使用之 RGB 及笔刷大小。

【定义】

数据类型	参数名称	描述
Inner structure	color	欲使用的 RGB 值
uint32_t	size	笔刷大小

6.2.4 cvtdl_area_detect_e

【enum】

数值	参数名称	描述
0	UNKNOWN	int8_t 特征类型
1	NO_INTERSECT	不相交
2	ON_LINE	在线上
3	CROSS_LINE_POS	正向交叉
4	CROSS_LINE_NEG	负向交叉
5	INSIDE_POLYGON	在多边形内部
6	OUTSIDE_POLYGON	在多边形外部

7 错误码

错误代码	宏定义	描述
0xFFFFFFFF	CVI_TDL_FAILURE	API 调用失败
0xC0010101	CVI_TDL_ERR_INVALID_MODEL_PATH	不正确的模型路径
0xC0010102	CVI_TDL_ERR_OPEN_MODEL	开启模型失败
0xC0010103	CVI_TDL_ERR_CLOSE_MODEL	关闭模型失败
0xC0010104	CVI_TDL_ERR_GET_VPSS_CHN_CONFIG	取得 VPSS CHN 设置失败
0xC0010105	CVI_TDL_ERR_INFERENCE	模型推理失败
0xC0010106	CVI_TDL_ERR_INVALID_ARGS	不正确的参数
0xC0010107	CVI_TDL_ERR_INIT_VPSS	初始化 VPSS 失败
0xC0010108	CVI_TDL_ERR_VPSS_SEND_FRAME	送 Frame 到 VPSS 时失败
0xC0010109	CVI_TDL_ERR_VPSS_GET_FRAME	从 VPSS 取得 Frame 失败
0xC001010A	CVI_TDL_ERR_MODEL_INITIALIZED	模型未开启
0xC001010B	CVI_TDL_ERR_NOT_YET_INITIALIZED	功能未初始化
0xC001010C	CVI_TDL_ERR_NOT_YET_IMPLEMENTED	功能尚未实现
0xC001010D	CVI_TDL_ERR_ALLOC_ION_FAIL	分配 ION 内存失败
0xC0010201	CVI_TDL_ERR_MD_OPERATION_FAILED	运行 Motion Detection 失败