TDL SDK Software Development Guide

Version: 1.1.0

Release date: 2022-6-15

Contents

Revision History

Revision	Date	Description
1.0.0	2021/6/30	Draft
1.0.1	2022/2/11	Update API Example description
1.1.0	2022/6/15	Update API
1.2.0	2022/7/27	Update API

1 Disclaimer

Terms and Conditions

The document and all information contained herein remain the CVITEK Co., Ltd' s ("CVITEK") confidential information, and should not disclose to any third party or use it in any way without CVITEK' s prior written consent. User shall be liable for any damage and loss caused by unauthority use and disclosure.

CVITEK reserves the right to make changes to information contained in this document at any time and without notice.

All information contained herein is provided in "AS IS" basis, without warranties of any kind, expressed or implied, including without limitation mercantability, non-infringement and fitness for a particular purpose. In no event shall CVITEK be liable for any third party' s software provided herein, User shall only seek remedy against such third party. CVITEK especially claims that CVITEK shall have no liable for CVITEK' s work result based on Customer' s specification or published shandard.

Contact Us

Address

Building 1, Yard 9, FengHao East Road, Haidian District, Beijing, 100094, China

Building T10, UpperCoast Park, Huizhanwan, Zhancheng Community, Fuhai Street, Baoan District, Shenzhen, 518100, China

Phone

+86-10-57590723 +86-10-57590724

Website

https://www.sophgo.com/

Forum

https://developer.sophgo.com/forum/index.html

2 Function Overview

2.1 Objective

Cvitek provides TDL integration algorithms to reduce the time required for application development.

This architecture realizes the algorithm required by TDL, including its pre and post processing, and provides a unified and convenient programming interface.

At present, TDL SDK includes motion detection, face detection, face recognition, face tracking, pedestrian detection, semantic segmentation, license plate recognition, license plate detection, live recognition, IR live recognition, infant detection, cry detection, attitude detection, gesture detection, Gesture Recognition and other algorithms.

3 Design Overview

3.1 System Architecture

The following is the TDL SDK system architecture diagram; The TDL SDK architecture is on Cvitek’ s Middleware and TPU SDK.

It is mainly divided into three modules: Core, Service and Application.

Core mainly provides algorithm-related interfaces, encapsulating complex underlying operations and algorithm details.

Users can directly use the Video Frame Buffer obtained by VI or VPSS for model reasoning.

TDL SDK will conduct corresponding pre and post processing on the model and complete reasoning.

Service provides algorithmic auxiliary apis, such as mapping, feature comparison, and area intrusion determination.

Application Encapsulates the application logic, including the face capture application logic.

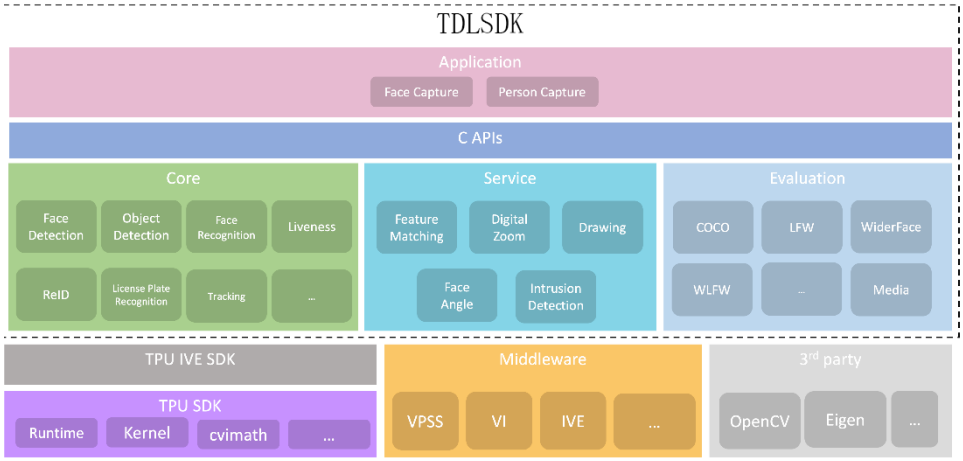

Fig. 3.1: Figure 1.

The three modules are placed in two libraries:

Module	Dynamic Library	Static Library
Core, Service	libcvi_tdl.so	libcvi_tdl.a
Application	libcvi_tdl_app.so	libcvi_tdl_app.a

3.2 Directory Structure

The directory structure of TDL SDK is as follows:

Directory Name	Description
include	TDL SDK headers
sample	Sample source code
doc	Markdown-formatted document
lib	TDL SDK static and dynamic libraries
bin	TDL SDK binary file

4 API Reference

4.1 Handle

【Syntax】

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

【Description】

In TDL SDK, each module has its own handle, but when creating the cvitdl_service_handle_t module, the cvitdl_handle_t is required as an input.

4.2 CVI_TDL_Core

4.2.1 Common

4.2.1.1 CVI_TDL_CreateHandle

【Syntax】

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

【Description】

Create handle to use TDL SDK. The TDL SDK automatically creates a VPSS Group.

【Parameter】

	Data Type	Parameter	Description
Input/Output	cvitdl_handle_t*	handle	Input handle pointer

4.2.1.2 CVI_TDL_CreateHandle2

【Syntax】

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

【Description】

Create a handle to use the TDL SDK and use the specified VPSS Group ID and Dev ID to create a VPSS.

【Parameter】

	Data Type	Parameter	Description
Output	cvitdl_handle_t*	handle	Input handle pointer
Input	VPSS_GRP	vpssGroupId	Group ID used by VPSS
Input	CVI_U8	vpssDev	VPSS Device id

4.2.1.3 CVI_TDL_DestroyHandle

【Syntax】

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

【Description】

Destroy the created handle cvitdl_handle_t. Destroy all open models at the same time

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Input handle pointer

4.2.1.4 CVI_TDL_GetModelPath

```
const char *CVI_TDL_GetModelPath(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_
↪MODEL_E model);
```

【Description】

Gets the model path of the model that has been set up internally to support. Release the filepath variable after use.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Input handle pointer
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID

【Output】

	Data Type	Description
Output	char*	Model path pointer

4.2.1.5 CVI_TDL_OpenModel

【Syntax】

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

【Description】

Enable and initialize the model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Input handle pointer
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model index
Input	const char*	filepath	cvimodel model path

4.2.1.6 CVI_TDL_SetSkipVpssPreprocess

【Syntax】

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

【Description】

Disable preprocessing for a specified model.

By default, TDL SDK uses the internally created VPSS to preprocess the model (skip = false).

When skip is set to true, TDL SDK will not preprocess the model.

The model input must be preprocessed externally before being input to the model.

This is usually used when directly binding VPSS to VI and using only a single model.

CVI_TDL_GetVpssChnConfig can be used to obtain the VPSS preprocessing parameters for the model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Input	bool	skip	Whether to skip preprocessing

4.2.1.7 CVI_TDL_GetSkipVpssPreprocess

【Syntax】

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

【Description】

Inquire whether the model will be preprocessed within TDL SDK.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Output	bool*	skip	Inquire whether the model will be preprocessed within TDL SDK.

4.2.1.8 CVI_TDL_SetVpssThread

【Syntax】

```
CVI_S32 CVI_TDL_SetVpssThread(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_E␣
↪model, const uint32_t thread);
```

【Description】

Set the thread ID used by a specific model. In TDL SDK, a VPSS thread represents a set of VPSS group settings. Thread 0 is used by default for the VPSS group used by the model. When multiple threads use the same TDL SDK handle for model inference, this API must be used to specify different VPSS threads to avoid race conditions.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Input	uint32_t	thread	Thread ID

4.2.1.9 CVI_TDL_SetVpssThread2

【Syntax】

```
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);
```

【Description】

Same as CVI_TDL_SetVpssThread. You can specify the Vpss Group ID.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Input	uint32_t	thread	Thread ID
Input	VPSS_GRP	vpssGroupId	VPSS Group id
Input	const CVI_U8	dev	VPSS Device id

4.2.1.10 CVI_TDL_GetVpssThread

【Syntax】

```
CVI_S32 CVI_TDL_GetVpssThread(cvitdl_handle_t handle, CVI_TDL_SUPPORTED_MODEL_E␣
↪model, uint32_t *thread);
```

【Description】

Gets the thread id used by the model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Output	uint32_t*	thread	VPSS thread ID

4.2.1.11 CVI_S32 CVI_TDL_GetVpssGrpIds

【Syntax】

```
CVI_S32 CVI_TDL_GetVpssGrpIds(cvitdl_handle_t handle, VPSS_GRP **groups, uint32_
↪t *num);
```

【Description】

Get all the VPSS group IDs used in the handle. After use, the groups must be released manually.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Output	VPSS_GRP **	groups	Reference to a null pointer
Output	uint32_t*	num	Length of groups

4.2.1.12 CVI_TDL_SetVpssTimeout

【Syntax】

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

【Description】

Set the TDL SDK to wait for VPSS hardware timeout, with a default setting of 100ms. This setting applies to all VPSS threads within TDL SDK.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handel
Input	uint32_t	timeout	timeout

4.2.1.13 CVI_TDL_SetVBPool

【Syntax】

```
CVI_S32 CVI_TDL_SetVBPool(cvitdl_handle_t handle, uint32_t thread, VB_POOL pool_
↪id);
```

【Description】

Specify a VBPool for the internal VPSS in TDL SDK. After being specified, the internal VPSS in TDL SDK will directly obtain memory from this pool. If this API is not used to specify a pool, the system will allocate one automatically.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	uint32_t	thread	VPSS thread ID
Input	VB_POOL	pool_id	VB Pool Id. If INVALID_POOLID is set, the Pool is not specified and is automatically allocated by the system.

4.2.1.14 CVI_TDL_GetVBPool

【Syntax】

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

【Description】

Get the VBPool ID used by the specified VPSS. If *CVI_TDL_SetVBPool* is not used to specify a pool, INVALID_POOLID will be returned.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	uint32_t	thread	VPSS thread ID
Output	VB_POOL*	pool_id	The current VB Pool Id is used.

4.2.1.15 CVI_TDL_CloseAllModel

【Syntax】

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

【Description】

Unload all models that have been loaded in the handle.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle

4.2.1.16 CVI_TDL_CloseModel

【Syntax】

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

【Description】

Dismount the specific model that has been loaded in the handle.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model index

4.2.1.17 CVI_TDL_Dequantize

【Syntax】

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

【Description】

Dequantize int8 values to float.

【Parameter】

	Data Type	Parameter	Description
Input	const int8_t*	quantizedData	Int8 data
Output	float*	data	Float output data
Input	const uint32_t	bufferSize	Int8 data quantity
Input	const float	dequantizeThreshold	Quantization threshold

4.2.1.18 CVI_TDL_ObjectNMS

【Syntax】

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

【Description】

Run the Non-Maximum Suppression algorithm for bboxes in cvtdl_object_t.

【Parameter】

	Data Type	Parameter	Description
Input	const cvtdl_object_t*	obj	Object Meta of the NMS
Output	cvtdl_object_t*	objNMS	Result after NMS
Input	const float	threshold	IOU threshold
Input	const char	method	'u' : Intersection over Union 'm' : Intersection over min area

4.2.1.19 CVI_TDL_FaceNMS

【Syntax】

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

【Description】

Run the Non-Maximum Suppression algorithm for bboxes in cvtdl_face_t.

【Parameter】

	Data Type	Parameter	Description
Input	const cvtdl_face_t*	face	face meta of the NMS
Output	cvtdl_face_t*	faceNMS	Result after NMS
Input	const float	threshold	IOU threshold
Input	const char	method	'u' : Intersection over Union 'm' : Intersection over min area

4.2.1.20 CVI_TDL_FaceAlignment

【Syntax】

```
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);
```

【Description】

InsightFace Alignment parameter was used to perform Face Alignment for inFrame image Face.

【Parameter】

	Data Type	Parameter	Description
Input	VIDEO_FRAME_INFO_S*	inFrame	Input image
Input	const uint32_t metaWidth	metaWidth	The width of the frame in Info
Input	const uint32_t metaHeight	metaHeight	The height of the frame in Info
Input	const cvtdl_face_info_t*	info	Face info
Output	VIDEO_FRAME_INFO_S*	outFrame	The face image after face alignment.
Input	const bool	enableGDC	Whether to use GDC hardware

4.2.1.21 CVI_TDL_CropImage

【Syntax】

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

【Description】

Retrieves the Bbox-specified region image from the srcFrame image.

【Parameter】

	Data Type	Parameter	Description
Input	VIDEO_FRAME_INFO_S*	srcFrame	Input image, currently only supports RGB packed format.
Output	cvtdl_image_t*	dst	Output image.
Input	cvtdl_bbox_t*	bbox	Bounding box
Input	bool	cvtRGB888	Whether to convert to RGB888 format for output.

4.2.1.22 CVI_TDL_CropImage_Face

【Syntax】

```
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);
```

【Description】

Extract the image within the specified face bbox from the srcFrame image.

【Parameter】

	Data Type	Parameter	Description
Input	VIDEO_FRAME_INFO_S*	srcFrame	Input image, currently only supports RGB packed format.
Output	cvtdl_image_t*	dst	Output image.
Input	cvtdl_face_info_t*	face_info	Specified face info.
Input	bool	align	Whether to perform face alignment. Using InsightFace alignment parameters.
Input	bool	cvtRGB888	Whether to convert the output format to RGB888.

4.2.1.23 CVI_TDL_SoftMax

【Syntax】

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

【Description】

Compute Softmax for the inputBuffer.

【Parameter】

	Data Type	Parameter	Description
Input	const float*	inputBuffer	Buffer to be softmaxed
Output	const float*	outputBuffer	resulting softmax buffer
Input	const uint32_t	bufferSize	buffer size

4.2.1.24 CVI_TDL_GetVpssChnConfig

【Syntax】

```
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);
```

【Description】

Get the VPSS parameters used in model pre-processing.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model ID
Input	CVI_U32	frameWidth	Input image width
Input	CVI_U32	fra meHeight	Input image height
Input	CVI_U32	idx	The input index of the model
Output	cvtdl_vpssconfig_t*	chnConfig	The specified parameter value to be returned.

4.2.1.25 CVI_TDL_Free

```
CVI_TDL_Free(X)
```

【Description】

Free the data structure generated by the model results. Some data structures may contain sub-items allocated by malloc, so they need to be released.

【Parameter】

The following are the supported input variables:

- *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)
```

【Description】

Generic API for copying CVI_TDL structures. This API will allocate the pointer memory for the copied structure and perform a complete copy.

【Parameter】

	Data Type	Parameter	Description
Input	Support type: cvtdl_face_info_t cvtdl_object_info_t cvtdl_image_t	IN	Replication source
Output	Support type: cvtdl_face_info_t cvtdl_object_info_t cvtdl_image_t	OUT	Replication purpose

4.2.1.27 CVI_TDL_RescaleMetaCenter

【Description】

Rescale the coordinates in the structure to the size of the input image with padding on top, bottom, left, and right.

【Parameter】

Here are the supported input variables:

- *cvtdl_face_t*
- *cvtdl_object_t*

4.2.1.28 CVI_TDL_RescaleMetaRB

【Description】

In the padding image, the coordinates in the structure are restored to the same size as the input image.

【Parameter】

Here are the supported input variables:

- *cvtdl_face_t*
- *cvtdl_object_t*

4.2.1.29 getFeatureTypeSize

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

【Description】

Gets the unit size of the eigenvalue.

【Parameter】

	Data Type	Parameter	Description
Input	feature_type_e	type	Unit

【Output】

	Data Type	Parameter	Description
Output	int	X	The unit size is byte

4.2.1.30 CVI_TDL_SetModelThreshold

【Syntax】

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

【Description】

Set the threshold value for the model, currently only supported for models of the detection type.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model index
Input	float	threshold	Threshold (0.0~1.0)

4.2.1.31 CVI_TDL_GetModelThreshold

【Syntax】

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

【Description】

Take out the model threshold, and only the Detection type model is supported at present.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model index
Output	float*	threshold	Threshold

4.2.2 Object Detection

4.2.2.1 CVI_TDL_MobileDetV2_Vehicle

【Syntax】

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

【Description】

Perform inference using the MobilDetV2-Vehicle model, which can detect three categories: Car, Motorcycle, and Truck.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_object_t*	obj	Objects detected

4.2.2.2 CVI_TDL_MobileDetV2_Pedestrian

【Syntax】

```
CVI_S32 CVI_TDL_MobileDetV2_Pedestrian(cvitdl_handle_t handle, VIDEO_FRAME_INFO_
↪S *frame, cvtdl_object_t *obj);
```

【Description】

Perform inference using MobilDetV2-Pedestrian series model which can detect objects of the class “person” .

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_object_t*	obj	Objects detected

4.2.2.3 CVI_TDL_MobileDetV2_Person_Vehicle

【Syntax】

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

【Description】

Perform inference using the MobilDetV2-Person-Vehicle model, which can detect the person, car, and non-motorized vehicle classes.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_object_t*	obj	Objects detected

4.2.2.4 CVI_TDL_MobileDetV2_Person_Pets

【Syntax】

```
CVI_S32 CVI_TDL_MobileDetV2_Person_Pets(cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_object_t *obj);
```

【Description】

Perform inference using the MobilDetV2-Lite-Person-Pets model, which can detect person, cat, dog, and other classes.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input Image
Output	cvtdl_object_t*	obj	Detected Objects

4.2.2.5 CVI_TDL_MobileDetV2_COCO80

【Syntax】

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

【Description】

Perform inference using MobilDetV2 COCO80 series model, which can detect 80 classes in the standard COCO dataset.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input Image
Output	cvtdl_object_t*	obj	Detected Objects.

4.2.2.6 CVI_TDL_Yolov3

【Syntax】

```
CVI_S32 CVI_TDL_Yolov3 (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_object_t *obj);
```

【Description】

Perform inference using the YOLOv3 model, which can detect 80 classes in the COCO dataset.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input Image
Output	cvtdl_object_t*	obj	Detected Objects

4.2.2.7 CVI_TDL_Yolov5

【Syntax】

```
CVI_S32 CVI_TDL_Yolov5 (cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame,␣
↪cvtdl_object_t *obj);
```

【Description】

Perform inference using the YOLOv5 model, which can detect 80 classes in the COCO dataset.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input Image
Output	cvtdl_object_t*	obj	Detected Objects

4.2.2.8 CVI_TDL_YoloX

【Syntax】

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

【Description】

Perform inference using the YoloX model, which can detect 80 classes in the COCO dataset.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input Image
Output	cvtdl_object_t*	obj	Detected Objects

4.2.2.9 CVI_TDL_SelectDetectClass

【Syntax】

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

【Description】

Filter the output results of the Object Detection model and keep the listed classes or groups.

The number of classes is determined by num_classes, and the detailed class and group Index can be referred to *cvtdl_obj_class_id_e* and *cvtdl_obj_det_group_type_e* .

This function is currently only supported for MobileDetV2 and YoloX models.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	CVI_TDL_SUPPORTED_MODEL_E	model	Model Index
Input	uint32_t	num_classes	Number of reserved categories
Input	cvtdl_obj_class_id_e 或 cvtdl_obj_det_group_type_e	Description	Eserved Class ID or Group ID

4.2.2.10 CVI_TDL_ThermalPerson

【Syntax】

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

【Description】

Thermal imaging of a human body.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_object_t*	faces	The human form that was detected

4.2.3 Face detection

4.2.3.1 CVI_TDL_RetinaFace

【Syntax】

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

【Description】

Detect faces using the RetinaFace model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_face_t*	faces	Detected faces

4.2.3.2 CVI_TDL_RetinaFace_IR

【Syntax】

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

【Description】

The RetinaFace model was used for face detection in IR images.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input IR image
Output	cvtdl_face_t*	faces	Detected faces

4.2.3.3 CVI_TDL_RetinaFace_Hardhat

【Syntax】

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

【Description】

Detect faces wearing safety helmets using RetinaFace model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input IR image
Output	cvtdl_face_t*	faces	The faces wedetected

4.2.3.4 CVI_TDL_ScrFDFace

【Syntax】

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

【Description】

Detect faces using RetinaFace model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_face_t*	faces	The faces we detected

4.2.3.5 CVI_TDL_ThermalFace

【Syntax】

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

【Description】

Thermal imaging-based face detection.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_face_t*	faces	The faces we detected

4.2.3.6 CVI_TDL_FLDet3

【Syntax】

```
CVI_S32 CVI_TDL_FLDet3(cvitdl_handle_t handle, VIDEO_FRAME_INFO_S *frame, cvtdl_
↪face_t *faces);
```

【Description】

Determine the face keypoints in the incoming face structure.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	The face detected
Output	cvtdl_face_t*	faces	The face keypoints

4.2.3.7 CVI_TDL_FaceQuality

【Syntax】

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

【Description】

Evaluate the quality of the faces in the given "faces" structure and detect their angles. The quality is affected by the clarity of the face and whether it is occluded. The quality score of a face is stored in "faces->info[i].face_quality" , while the angle is stored in "faces->info[i].head_pose" .

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	face	The faces we detected
Input	bool*	skip	Bool array: Specifies the face quality required for the face. NULL indicates that all faces do this.

4.2.3.8 CVI_TDL_FaceMaskDetection

【Syntax】

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

【Description】

Detect faces wearing masks. The face score is stored in faces->info[i].bbox.score, and the score of faces wearing masks is stored in faces->info[i].mask_score.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	cvtdl_face_t*	faces	The faces we detected

4.2.3.9 CVI_TDL_MaskClassification

【Syntax】

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

【Description】

Check if all faces in the input faces are masked faces. A face detection must be performed before calling this interface. The score of masked face is stored in faces->info[i].mask_score.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	faces	The faces we detected

4.2.4 Face Recognition.

4.2.4.1 CVI_TDL_FaceRecognition

【Syntax】

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

【Description】

Extract facial features. This interface will extract features for all faces in the face variable and store them in faces->info[i].feature.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input/Output	cvtdl_face_t*	faces	The faces we detected

4.2.4.2 CVI_TDL_FaceRecognitionOne

【Syntax】

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

【Description】

Extract face features. This interface will only extract features for the specified face index and store them in faces->info[index].feature.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input/Outpu	cvtdl_face_t*	faces	The faces we detected
Input	int	face_idx	Index of the face to extract features from. Use -1 to extract features from all faces.

4.2.4.3 CVI_TDL_FaceFeatureExtract

【Syntax】

```
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);
```

【Description】

Extract facial features. This interface will only perform feature extraction on the specified 'rgb_pack' position. The extracted features will be stored in 'face_info->feature.ptr'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	const uint8_t*	rgb_pack	Input image pixel starting address
Input	int	width	Input image width
Input	int	height	Input image height
Input	int	stride	Input image stride
Input/Outpu	cvtdl_face_info_t*	face_info	The face feature we detected

4.2.4.4 CVI_TDL_FaceAttribute

【Syntax】

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

【Description】

Extract facial features and attributes. This interface will extract features and attributes, including gender, expression, age, and race, for all faces in the input faces data structure. The results will be stored in faces->info[i].feature, faces->info[i].age, faces->info[i].emotion, faces->info[i].gender, and faces->info[i].race respectively.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input/Output	cvtdl_face_t*	faces	The faces we detected

4.2.4.5 CVI_TDL_FaceAttributeOne

【Syntax】

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

【Description】

Extract facial features and attributes for a specific face index using this interface. The facial features and attributes include gender, emotion, age, and race, and the results will be stored in faces->info[i].feature, faces->info[i].age, faces->info[i].emotion, faces->info[i].gender, faces->info[i].race.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input/Output	cvtdl_face_t*	faces	The faces we detected
Input	int	face_idx	Index of the face to extract features from. Use -1 to extract features from all faces.

4.2.4.6 CVI_TDL_MaskFaceRecognition

【Syntax】

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

【Description】

Extract features of faces wearing masks. This interface will extract features for all faces in the input faces and store them in faces->info[i].feature.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input/Output	cvtdl_face_t*	faces	The faces we detected

4.2.5 Pedestrian Recognition

4.2.5.1 CVI_TDL_OSNet

【Syntax】

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

【Description】

Extract pedestrian features using the person-reid model. This interface will extract features for all Person-class objects in obj and place them in obj->info[i].feature.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_object_t*	obj	Objects detected

4.2.5.2 CVI_TDL_OSNetOne

【Syntax】

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

【Description】

Extract pedestrian features using the person-reid model. This interface only extracts features for the specified object in obj and stores the features in obj->info[i].feature.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_object_t*	obj	Objects detected
Input	int	obj_idx	Index of the object to extract features from. -1 means to extract features from all objects.

4.2.6 Gesture Recognition

4.2.6.1 CVI_TDL_Hand_Detection

【Syntax】

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

【Description】

This function is used for detecting hand bounding boxes. The results will be stored in 'meta->info[i]'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	The handle for the TDL model.
Input	VIDEO_FRAME_INFO_S*	frame	The input image.
Output	cvtdl_object_t*	meta	Detected hand bounding boxes.

4.2.6.2 CVI_TDL_HandClassification

【Syntax】

```
CVI_S32 CVI_TDL_HandClassification(const cvitdl_handle_t handle, VIDEO_FRAME_
↪INFO_S *frame, cvtdl_object_t *meta);
```

【Description】

This gesture classification algorithm is used for recognizing gestures on the specified 'frame'. The results will be stored in 'meta->info[i].name' and 'meta->info[i].bbox.score'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	The handle for the TDL model.
Input	VIDEO_FRAME_INFO_S*	frame	The input image.
Input/Output	cvtdl_object_t*	meta	Detected hand bounding boxes and gesture classifications.

4.2.6.3 CVI_TDL_HandKeypoint

【Syntax】

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

【Description】

This function is used for outputting hand key points and storing them in 'meta->info[i]'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	The handle for the TDL model.
Input	VIDEO_FRAME_INFO_S*	frame	Input image.
Input/Output	cvtdl_handpose21_meta_ts*	meta	Detected hand bounding boxes and 21 hand key points.

4.2.6.4 CVI_TDL_HandKeypointClassification

【Syntax】

```
CVI_S32 CVI_TDL_HandKeypointClassification(const cvitdl_handle_t handle, VIDEO_
↪FRAME_INFO_S *frame, cvtdl_handpose21_meta_t *meta);
```

【Description】

This function is used for extracting hand keypoint information. The keypoint information will be stored in 'meta->info[i]'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	The handle for the TDL model.
Input	VIDEO_FRAME_INFO_S*	frame	Input 21 key points of the hand. The x and y coordinates are sequentially stored in 'frame->stVFrame.pu8VirAddr[0]', 'frame->stVFrame.u32Height=1', 'frame->stVFrame.u32Width=42*sizeof(float)'.
Output	cvtdl_handpose21_meta_t*	meta	The output of hand gestures, including the gesture label in 'meta->label' and the gesture score in 'meta->score'.

4.2.7 Object Tracking

4.2.7.1 CVI_TDL_DeepSORT_Init

【Syntax】

```
CVI_S32 CVI_TDL_DeepSORT_Init(const cvitdl_handle_t handle, bool use_specific_
↪counter);
```

【Description】

Initialize the DeepSORT algorithm.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	bool	use_specific_counter	Whether each object class is assigned its own id

4.2.7.2 CVI_TDL_DeepSORT_GetDefaultConfig

【Syntax】

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

【Description】

Get the default parameters for DeepSORT.

【Parameter】

	Data Type	Parameter	Description
Input	cvtdl_deepsort_config_t*	ds_conf	DeepSORT parameter

4.2.7.3 CVI_TDL_DeepSORT_SetConfig

【Syntax】

```
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);
```

【Description】

Setting DeepSORT parameters.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	cvtdl_deepsort_config_t*	ds_conf	DeepSORT parameter
Input	int	cvi_tdl_ol	"-1" indicates default settings. Non-negative values indicate setting parameters for cvi_tdl_obj_type.
Input	bool	show_conf	Display Settings.

4.2.7.4 CVI_TDL_DeepSORT_GetConfig

【Syntax】

```
CVI_S32 CVI_TDL_DeepSORT_GetConfig(const cvitdl_handle_t handle , cvtdl_
↪deepsort_config_t *ds_conf, int cvi_tdl_obj_type);
```

【Description】

To inquire about the DeepSORT parameters that have been set, please call the corresponding function or method provided by the DeepSORT implementation being used in your system. The specific method or function name may vary depending on the DeepSORT implementation you are using.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	TDL SDK handle
Output	cvtdl_deepsort_config_t*	ds_conf	DeepSORT parameter
Input	int	cvi_tdl_obj	-1 indicates that the default parameter is obtained. Non-1 values represent parameters set against the category of cvi_tdl_obj_type

4.2.7.5 CVI_TDL_DeepSORT_CleanCounter

【Syntax】

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

【Description】

Reset the ID counter recorded in DeepSORT.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle

4.2.7.6 CVI_TDL_DeepSORT_Obj

【Syntax】

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

【Description】

Track objects and update the tracker status. This interface will assign a unique ID to each Object, which can be obtained from obj->info[i].unique_id. The tracker_t will record the tracking status of each object and the current predicted Bounding Box. If you want to use object appearance features for tracking, set use_reid to true, and use CVI_TDL_OSNet for feature extraction before tracking. Currently, feature extraction only supports human-shaped objects.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	cvtdl_object_t*	obj	The object to be tracked
Output	cvtdl_tracker_t*	tracker	The trace state of the object
Input	bool	use_reid	Whether object appearance characteristics are used for tracing

4.2.7.7 CVI_TDL_DeepSORT_Face

【Syntax】

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

【Description】

Track faces and update the Tracker state. This function assigns a unique ID to each face, which can be retrieved from face->info[i].unique_id. The tracker_t will record the tracking status and the predicted bounding box of each face. If you want to use face features for tracking, set use_reid to true and call CVI_TDL_FaceRecognition to compute face features before tracking.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	cvtdl_face_t*	face	The face you want to track
Output	cvtdl_tracker_t*	tracker	The tracking state of a face
Input	bool	use_reid	Whether to use appearance characteristics for tracing. Currently, only false can be set

4.2.8 Motion Detection

4.2.8.1 CVI_TDL_Set_MotionDetection_Background

【Syntax】

```
CVI_S32 CVI_TDL_Set_MotionDetection_Background(const cvitdl_handle_t handle,␣
↪VIDEO_FRAME_INFO_S *frame);
```

【Description】

Set the background for Motion Detection. The first time this function is called, it will initialize the Motion Detection module. Subsequent calls will only update the background. The Motion Detection module in TDL SDK uses frame difference method.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Background

4.2.8.2 CVI_TDL_MotionDetection

【Syntax】

```
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);
```

【Description】

Detect objects using frame difference method. The detection results will be stored in "objects".

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	image
Output	cvtdl_object_t*	object	Motion detection result
Input	uint8_t	threshold	The frame difference threshold must be 0-255
Input	double	min_area	Minimum object area (Pixels). Filters out objects smaller than this value.

4.2.8.3 CVI_TDL_Set_MotionDetection_ROI

【Syntax】

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

【Description】

This function is used for object detection using frame difference method. The detection results will be stored in 'objects'.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	The handle for the TDL model.
Input	MDROI_t*	roi_s	Set the region of interest for motion detection.

4.2.9 License Plate Recognition

4.2.9.1 CVI_TDL_LicensePlateDetection

【Syntax】

```
CVI_S32 CVI_TDL_LicensePlateDetection(cvitdl_handle_t handle, VIDEO_FRAME_INFO_
↪S *frame, cvtdl_object_t *vehicle_meta);
```

【Description】

License plate detection. Before calling this API, vehicle detection must be performed first. This algorithm will perform license plate detection on the already detected objects. The location of the license plate will be stored in obj->info[i].vehicle_properity->license_pts.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	image
Input	cvtdl_object_t*	obj	Object (vehicle) detection result

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);
```

【Description】

Perform license plate recognition (Taiwan) on all vehicles in the input obj. Prior to calling this API, CVI_TDL_LicensePlateDetection must be called once for license plate detection. The license plate number is stored in obj->info[i].vehicle_properity->license_char.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	The input image
Input	cvtdl_object_t*	obj	License plate detection results

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);
```

【Description】

Perform license plate recognition (Mainland China) for all vehicles in the input obj. CVI_TDL_LicensePlateDetection must be called before calling this API for license plate detection. The license plate number is stored in obj->info[i].vehicle_properity->license_char.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	image
Input/Output	cvtdl_object_t*	obj	icense plate detection results

4.2.10 Tamper Detection

4.2.10.1 CVI_TDL_TamperDetection

【Syntax】

```
CVI_S32 CVI_TDL_TamperDetection(const cvitdl_handle_t handle, VIDEO_FRAME_INFO_
↪S *frame, float *moving_score);
```

【Description】

Camera Tampering Detection. This algorithm builds a background model based on the Gaussian model and uses the background subtraction method to calculate the difference as the tampering score (moving_score).

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	image
Output	float*	moving_score	Tamper score

4.2.11 Liveness Detection

4.2.11.1 CVI_TDL_Liveness

【Syntax】

```
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);
```

【Description】

Perform RGB and IR dual-liveness detection. Determine whether the faces in rgb_faces and ir_faces are real-time or not. The liveness score is stored in rgb_face -> info[i].liveness_score.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	rgbFrame	RGB image
Input	VIDEO_FRAME_INFO_S*	irFrame	IR image
Input/Output	cvtdl_face_t*	rgb_meta	Detected RGB face / Living fraction
Input	cvtdl_face_t*	ir_meta	The IR faces we detected

4.2.11.2 CVI_TDL_IrLiveness

【Syntax】

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

【Description】

This function performs IR (infrared) monocular liveness detection.

It determines whether the detected faces in ‘ir_faces’ are real or not.

The liveness scores for each face will be stored in ‘ir_faces->info[i].liveness_score’.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	irFrame	The ir image
Input/Output	cvtdl_face_t*	ir_faces	Detected IR face/ liveness score

4.2.12 Pose Estimation

4.2.12.1 CVI_TDL_AlphaPose

【Syntax】

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

【Description】

Perform inference using the AlphaPose model to predict 17 skeletal keypoints. The detection results are placed in obj->info[i].pedestrian_properity->pose_17.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_object_t*	obj	The person who was detected / 17 skeleton keypoints coordinates

4.2.13 Semantic Segmentation

4.2.13.1 CVI_TDL_DeeplabV3

【Syntax】

```
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);
```

【Description】

Perform semantic segmentation using the DeepLab V3 model.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Output	VIDEO_FRAME_INFO_S*	out_frame	Output image
Input	cvtdl_class_filter_t*	filter	Reserved category

4.2.14 Fall Detection

4.2.14.1 CVI_TDL_Fall

【Syntax】

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

【Description】

Detect falls using the results of object detection and pose estimation.

Prior to running this API, *CVI_TDL_AlphaPose* must be called to obtain the human body keypoints.

The fall detection results are stored in obj->info[i].pedestrian_properity->fall.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	The input image
Input	cvtdl_object_t*	obj	Fall condition outcome

4.2.15 Driver Fatigue Detection

4.2.15.1 CVI_TDL_FaceLandmarker

【Syntax】

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

【Description】

Before using this API, the face detection API should be called to obtain the detection result of 106 facial landmarks, which should be placed in face->dms[i].landmarks_106 and update the 5 facial landmarks in face->dms[i].landmarks_5.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	face	Human face

4.2.15.2 CVI_TDL_EyeClassification

【Syntax】

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

【Description】

The closed state of the eyes is determined based on the results of face detection and facial landmark detection. The scores of the right and left eyes are stored in face->dms[i].reye_score and face->dms[i].leye_score, respectively.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	face	Human face

4.2.15.3 CVI_TDL_YawnClassification

【Syntax】

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

【Description】

Detect yawning based on the results of face detection and facial landmark detection. CVI_FaceRecognition must be called first to obtain the results of face detection and facial landmark detection. The yawning score will be placed in face->dms[i].yawn_score, ranging from 0.0 to 1.0.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	face	Human face

4.2.15.4 CVI_TDL_IncarObjectDetection

【Syntax】

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

【Description】

Detect objects (water cup/mug/phone) around the driver using object detection. The detection results will be outputted in object format and stored in face->dms[i].dms.dms_od.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	face	Human face

4.2.16 Sound Classification

4.2.16.1 CVI_TDL_SoundClassification

【Syntax】

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

【Description】

Classify the audio in the frame into different categories and output the scores of each category in sorted order.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	int*	index	Scores in each category

4.2.16.2 CVI_TDL_Get_SoundClassification_ClassesNum

【Syntax】

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

【Description】

Get the number of audio categories.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle

【Output】

	Data Type	Description
Output	int	The number of category

4.2.16.3 CVI_TDL_Set_SoundClassification_Threshold

【Syntax】

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

【Description】

Set the audio category threshold.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_handle_t	handle	Handle
Input	const float	th	Similarity threshold, the similarity higher than this threshold will be taken out.

4.3 CVI_TDL_Service

4.3.1 Common

4.3.1.1 CVI_TDL_Service_CreateHandle

【Syntax】

```
CVI_S32 CVI_TDL_Service_CreateHandle(cvitdl_service_handle_t *handle, cvtdl_
↪handle tdl_handle);
```

【Description】

Create a Service handle

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t*	handle	handle
Input	cvitdl_handle_t	tdl_handle	cvi_tdl_core handle

4.3.1.2 CVI_TDL_Service_DestroyHandle

【Syntax】

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

【Description】

Destroy the Service handle

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t*	handle	Handle

4.3.1.3 CVI_TDL_Service_Polygon_SetTarget

【Syntax】

```
CVI_S32 CVI_TDL_Service_Polygon_SetTarget(cvitdl_service_handle_t handle, const␣
↪cvtdl_pts_t *pts);
```

【Description】

Set up a region of interest for intrusion detection with pts as the coordinates of the convex polygon. The order of the points should be clockwise or counterclockwise. Call CVI_TDL_Service_Polygon_Intersect to check if a bounding box intersects with the defined region.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	cvtdl_pts_t*	pts	Convex polygon point

4.3.1.4 CVI_TDL_Service_Polygon_Intersect

【Syntax】

```
CVI_S32 CVI_TDL_Service_Polygon_Intersect(cvitdl_service_handle_t handle, const␣
↪cvtdl_bbox_t *bbox, bool *has_intersect);
```

【Description】

Detect if a given bounding box is within the targeted intrusion area previously set by calling CVI_TDL_Service_Polygon_SetTarget.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	cvtdl_bbox_t*	bbox	Bounding box
Output	bool*	has_intersect	Intrusion or not

4.3.1.5 CVI_TDL_Service_RegisterFeatureArray

【Syntax】

```
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);
```

【Description】

Register a feature database by precomputing and loading the features in the featureArray into memory.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const cvtdl_service_feature_array_t	featureArray	Feature array structure
Input	const cvtdl_service_feature_matching_e	method	Currently, only COS_SIMILARITY is supported

4.3.1.6 CVI_TDL_Service_CalculateSimilarity

【Syntax】

```
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);
```

【Description】

Compute the cosine similarity between two feature vectors using the RISC-V. The cosine similarity is calculated using the following formula:

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

The Cosine Similarity between two features of length n is calculated using the following formula. Currently, only INT8 features are supported.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const cvtdl_feature_t*	feature_rhs	The first feature
Input	const cvtdl_feature_t*	feature_lhs	Second feature
Output	float*	score	Similarity

4.3.1.7 CVI_TDL_Service_ObjectInfoMatching

【Syntax】

```
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);
```

【Description】

Compute the cosine similarity between the object feature in **object_info** and the pre-calculated object features in the registered object feature library. Retrieve the top-K similarities that are greater than **threshold**. The calculation formula is as follows:

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

Calculate the Cosine Similarity between the object feature in object_info and the registered object feature in the feature library.

The top-K similarities greater than the given threshold will be returned.

If the number of features in the library is less than 1000, RISC-V will be used for calculation, otherwise TPU will be launched for computation.

Feature registration can be done through CVI_TDL_Service_RegisterFeatureArray.

Note that the feature length n must be consistent and INT8 features are currently supported.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const cvtdl_object_info_t*	object_info	Object Info
Input	const uint32_t	topk	Take topk similarity
Output	float	threshold	Similarity threshold, above which the similarity will be taken out
Output	uint32_t*	indices	The Index of the similarity that satisfies the condition in the library
Output	float*	sims	The similarity of the condition
Output	uint32_t*	size	The number of similarities that are finally extracted

4.3.1.8 CVI_TDL_Service_FaceInfoMatching

【Syntax】

```
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);
```

【Description】

Calculate the Cosine Similarity between the face features in face_info and the registered face feature library, and output the top-K similarities that are greater than the given threshold.

The calculation formula is as follows:

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

Where n is the feature length.

If the number of features in the library is less than 1000, the calculation will be performed on the RISC-V.

Otherwise, the TPU will be used for calculation.

Feature registration requires calling CVI_TDL_Service_RegisterFeatureArray.

Currently, only INT8 features are supported.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const cvtdl_face_info_t*	face_info	Face info
Input	const uint32_t	topk	Take topk similarity
Output	float	threshold	Similarity threshold, above which the similarity will be taken out
Output	uint32_t*	indices	The Index of the similarity that satisfies the condition in the library
Output	float*	sims	The similarity of the condition
Output	uint32_t*	size	The number of similarities that are finally extracted

4.3.1.9 CVI_TDL_Service_RawMatching

【Syntax】

```
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);
```

【Description】

Compute the Cosine Similarity between a given feature and the registered feature library.

Retrieve the top-K similarities greater than the specified threshold.

The computation is done using the following formula:

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

where n is the length of the feature.

If the number of registered feature arrays is less than 1000, the calculation will be performed on RISC-V.

Otherwise, TPU will be launched for calculation.

The feature arrays need to be registered using *CVI_TDL_Service_RegisterFeatureArray*.

Unlike *CVI_TDL_Service_FaceInfoMatching* and *CVI_TDL_Service_ObjectInfoMatching*, this API directly uses the feature array for comparison and does not require *cvtdl_face_info_t* or *cvtdl_object_info_t* to be passed in.

This API limits the feature type to be the same as the feature type in the feature arrays.

Currently, only INT8 features are supported.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const void*	feature	Feature array
Input	const feature_type_e	type	Feature type, currently only TYPE_INT8 is supported
Input	const uint32_t	topk	Take topk similarity
Output	float	threshold	Similarity threshold, above which the similarity will be taken out
Output	uint32_t*	indices	The Index of the similarity that satisfies the condition in the library
Output	float*	scores	The similarity of the condition
Output	uint32_t*	size	The number of similarities that are finally extracted

4.3.1.10 CVI_TDL_Service_FaceAngle

【Syntax】

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

【Description】

Estimate the pose of a single face.

【Parameter】

	Data Type	Parameter	Description
Input	cvtdl_pts_t*	pts	Face landmark
Output	cvtdl_head_pose_t*	hp	Face posture

4.3.1.11 CVI_TDL_Service_FaceAngleForAll

【Syntax】

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

【Description】

Calculate pose for multiple faces

【Parameter】

	Data Type	Parameter	Description
Input	cvtdl_face_t*	meta	Face data

4.3.2 Image Scaling

4.3.2.1 CVI_TDL_Service_FaceDigitalZoom

【Syntax】

```
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);
```

【Description】

Resizing (zooming in) the detected face.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	inFrame	Input image
Input	cvtdl_face_t*	meta	Face data
Input	float	face_skip_ratio	Neglect ratio
Input	float	trans_ratio	Amplification ratio
Input	float	padding_ratio	Expand bounding box ratio
Output	VIDEO_FRAME_INFO_S*	outFrame	Output image

4.3.2.2 CVI_TDL_Service_ObjectDigitalZoom

【Syntax】

```
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);
```

【Description】

The object detected in the detection result is zoomed in.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const VIDEO_FRAME_INFO_S*	inFrame	Input image
Input	const cvtdl_object_t*	meta	Object data
Input	const float	obj_skip_ratio	Neglect ratio
Input	const float	trans_ratio	Amplification ratio
Input	const float	padding_ratio	Expand bounding box ratio
Output	VIDEO_FRAME_INFO_S*	outFrame	Output image

4.3.2.3 CVI_TDL_Service_ObjectDitgitalZoomExt

【Syntax】

```
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);
```

【Description】

Zoom in on the object in the detection results

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	VIDEO_FRAME_INFO_S*	inFrame	Input image
Input	const cvtdl_object_t*	meta	Object data
Input	const float	obj_skip_ratio	Neglect ratio
Input	const float	trans_ratio	Amplification ratio
Input	const float	pad_ratio_left	Expansion rate (left)
Input	const float	pad_ratio_right	Expansion rate (right)
Input	const float	pad_ratio_top	Expansion rate (top)
Input	const float	pad_ratio_bottom	Expansion rate (lower)
Output	VIDEO_FRAME_INFO_S*	outFrame	Output image

4.3.3 Image drawing

4.3.3.1 CVI_TDL_Service_FaceDrawPts

【Syntax】

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

【Description】

Drawing face landmarks.

【Parameter】

	Data Type	Parameter	Description
Input	cvtdl_pts_t*	pts	Face landmark
Input	VIDEO_FRAME_INFO_S*	hp	Input/output image

4.3.3.2 CVI_TDL_Service_FaceDrawRect

【Syntax】

```
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);
```

【Description】

Drawing a bounding box around a detected face.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	cvtdl_face_t*	meta	Face data
Input	VIDEO_FRAME_INFO_S*	frame	Input/output image
Input	bool	drawText	Whether to draw face names
Input	cvtdl_service_brush_t	brush	color

4.3.3.3 CVI_TDL_Service_ObjectDrawPose

【Syntax】

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

【Description】

Draw 17 skeletal points detected by pose estimation.

【Parameter】

	Data Type	Parameter	Description
Input	cvtdl_object_t*	meta	Skeletal points detection result
Input	VIDEO_FRAME_INFO_S*	frame	Input image

4.3.3.4 CVI_TDL_Service_ObjectDrawRect

【Syntax】

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

【Description】

Draw bounding box of detected objects

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	cvtdl_object_t*	meta	Object detection result
Input	VIDEO_FRAME_INFO_S*	frame	Input/output image
Input	const bool	drawText	Whether to draw class text

4.3.3.5 CVI_TDL_Service_ObjectWriteText

【Syntax】

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

【Description】

Draw specified text

【Parameter】

	Data Type	Parameter	Description
Input	char*	name	Drawn text
Input	int	x	The x coordinate is drawn
Input	int	y	The y coordinate is drawn
Input/Output	IDEO_FRAME_INFO_S*	rame	Input/output image
Input	float	r	Draw the color r channel value
Input	float	g	Draw the color g channel value
Input	float	b	Draw the color b channel value

4.3.3.6 CVI_TDL_Service_Incar_ObjectDrawRect

【Syntax】

```
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);
```

【Description】

Draw specified text

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_service_handle_t	handle	handle
Input	const cvtdl_dms_od_t*	meta	Object detection result
Input/Outpu	IDEO_FRAME_INFO_S*	rame	Input/output image
Input	const bool	drawText	Whether to draw class text
Input	cvtdl_service_brush_t	brush	color

5 Application（APP）

5.1 Objective

Cvitek TDL Application (APP) is a solution designed for different client applications based on TDL SDK.

The APP integrates TDL SDK and provides customers with more convenient operation APIs.

The APP code is open source and can be used as a reference for client development.

【compile】

1. Download TDL SDK and its dependent SDK: MW, TPU, IVE.
2. Move to the module/app directory of the TDL SDK
3. Execute the following commands:

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

make install

make clean
```

The compiled lib will be placed in the tmp install directory of the TDL SDK

5.2 API

5.2.1 Handle

【Syntax】

```
typedef struct {

cvitdl_handle_t tdl_handle;

IVE_HANDLE ive_handle;

```

(continues on next page)

(continued from previous page)

```
face_capture_t *face_cpt_info;

} cvitdl_app_context_t;

typedef cvitdl_app_context *cvitdl_app_handle_t;
```

【Description】

The cvitdl_app_handle_t is a pointer type of cvitdl_app_context, which includes the TDL handle, IVE handle, and other data structures for the application.

5.2.1.1 CVI_TDL_APP_CreateHandle

【Syntax】

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

【Description】

Create the metrics required for using the APP, which require input of TDL handle and IVE handle.

【Parameter】

	Data Type	Parameter	Description
Output	cvitdl_app_handle_t*	handle	Handle pointer input
Input	cvitdl_handle_t	a i_handle	TDL handle

5.2.1.2 CVI_TDL_APP_DestroyHandle

【Syntax】

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

【Description】

Destroy the created handle cvitdl_app_handle_t. Only the data used by individual applications will be destroyed, and it does not affect the tdl handle and ive handle.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input

5.2.2 Face Capture

Face Capture is a function that combines face detection, multi-object tracking, and face quality assessment to capture (or extract) facial images of different individuals in an image or video. The capture conditions can be adjusted using a configuration file, such as capture time, face quality assessment algorithm, face angle threshold, etc.

【Configuration File】

Parameter	Default Value	Description
miss_time_limit:	40	Face loss time limit. When the APP cannot trace a face continuously, it will determine that the face has left. [Unit: frame]
threshold_size_min	32	Minimum/maximum acceptable face size, if either side of the face bbox is less than/greater than this threshold, quality will be forced to 0.
threshold_size_max	512	
quality_assessment_method	0	If face evaluation does not use FQNet, enable the built-in quality detection algorithm 0: based on face size and Angle 1: Based on eye distance
threshold_quality	0.1	Face quality threshold. If the quality of the new face is greater than this threshold and higher than the quality of the currently captured face, the face data in the staging area will be captured and updated.
threshold_quality_high	0.95	Face quality threshold (high). If the quality of a face in the temporary area is higher than this threshold, the face is judged to be of high quality and will not be updated later. (level 2,3 only)
threshold_yaw	0.25	Face Angle threshold, if the Angle is greater than this threshold, quality will be forced to set to 0. (One unit is 90 degrees)
threshold_pitch	0.25	
threshold_roll	0.25	
fast_mode_interval	10	FAST mode capture interval. [Unit: frame]
fast_mode_capture_num	3	Number of captures in FAST mode.
cycle_mode_interval	20	CYCLE mode Snapshot interval. [Unit: frame]
auto_mode_time_limit	0	Duration of the last output in AUTO mode. [Unit: frame]
auto_mode_fast_cap	1	AUTO Mode Whether to output one quick snapshot.
capture_aligned_face	0	Whether the captured/captured face is corrected.

【Face Quality Assessment Algorithm】

#	Algorithm	Computation
0	Based on face size and Angle	1. Face Area Score 1. Definition of standard face size A_base = 112 * 112 2. Calculate the detected face area A_face = length * width 3. Calculate MIN(1.0, A_face/A_base) as a fraction 2. Face Pose Score 1. Calculate face Angle yaw, pitch and roll respectively and take their absolute values 2. Calculate 1 - (yaw + pitch + roll) / 3 as a score 3. Face Quality = Face Area Score * Face Pose Score
1	Eye distance based	1 Define standard pupil distance D = 80 2 Calculate the distance d between the eyes 3 Calculate MIN(1.0, d/D) as a fraction

5.2.2.1 CVI_TDL_APP_FaceCapture_Init

【Syntax】

```
CVI_S32 CVI_TDL_APP_FaceCapture_Init(const cvitdl_app_handle_t handle, uint32_t␣
↪buffer_size);
```

【Description】

Initialize the data structure for face capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	uint32_t	buffer_size	Face staging area size

5.2.2.2 CVI_TDL_APP_FaceCapture_QuickSetUp

【Syntax】

```
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);
```

【Description】

Quickly set up face capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	int	fd_model_id	Face detection Model ID
Input	int	fr_model_id	Face recognition model ID
Input	const char*	fd_model_pat	Face detection model Path
Input	const char*	fr_model_patl	Face recognition model path
Input	const char*	fq_model_pat	Face quality detection model path
Input	const char*	fl_model_patl	Face key point detection model path

5.2.2.3 CVI_TDL_APP_FaceCapture_FusePedSetup

【Syntax】

```
CVI_S32 CVI_TDL_APP_FaceCapture_FusePedSetup(const cvitdl_app_handle_t handle,␣
↪int ped_model_id, const char *ped_model_path);
```

【Description】

Quickly set up face capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	int	fd_model_id	Pedestrian detection model ID
Input	const char*	fl_model_patl	Pedestrian detection model path

5.2.2.4 CVI_TDL_APP_FaceCapture_GetDefaultConfig

【Syntax】

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

【Description】

Get the preset parameters for face capture.

【Parameter】

	Data Type	Parameter	Description
Output	face_capture_config_t*	cfg	Face capture parameters

5.2.2.5 CVI_TDL_APP_FaceCapture_SetConfig

【Syntax】

```
CVI_S32 CVI_TDL_APP_FaceCapture_SetConfig(const cvitdl_app_handle_t handle,␣
↪face_capture_config_t *cfg);
```

【Description】

Setting face capture parameters.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	face_capture_config_t*	cfg	Face capture parameters.

5.2.2.6 CVI_TDL_APP_FaceCapture_FDFR

【Syntax】

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

【Description】

Set face capture parameters.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	VIDEO_FRAME_INFO_S*	frame	Input image
Input	cvtdl_face_t*	p_face	Face capture output result

5.2.2.7 CVI_TDL_APP_FaceCapture_SetMode

【Syntax】

```
CVI_S32 CVI_TDL_APP_FaceCapture_SetMode(const cvitdl_app_handle_t handle,␣
↪capture_mode_e mode);
```

【Description】

Set face capture mode.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	capture_mode_e	mode	Face capture mode

5.2.2.8 CVI_TDL_APP_FaceCapture_Run

【Syntax】

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

【Description】

Perform face capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	VIDEO_FRAME_INFO_S*	frame	Input image

5.2.2.9 CVI_TDL_APP_FaceCapture_CleanAll

【Syntax】

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

【Description】

Clear all the data in the temporary storage area for face capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input

5.2.3 Humanoid Capture

Human body capture is a function that combines human body detection, multi-object tracking, and face quality detection to capture (or extract) photos of different people' s faces in an image.

The capture conditions can be adjusted using a configuration file, such as the capture time, face quality detection algorithm, face angle threshold, etc.

【Configuration File】

Parameter	Default Value	Description
miss_time_limit:	40	Face loss time limit. When the APP cannot trace a face continuously, it will determine that the face has left. [Unit: frame]
threshold_size_min	32	Minimum/maximum acceptable face size, if either side of the face bbox is less than/greater than this threshold, quality will be forced to 0.
threshold_size_max	512	
quality_assessment_method	0	If face evaluation does not use FQNet, enable the built-in quality detection algorithm 0: based on face size and Angle 1: Based on eye distance
threshold_quality	0.1	Face quality threshold. If the quality of the new face is greater than this threshold and higher than the quality of the currently captured face, the face data in the staging area will be captured and updated.
threshold_quality_high	0.95	Face quality threshold (high). If the quality of a face in the temporary area is higher than this threshold, the face is judged to be of high quality and will not be updated later. (level 2,3 only)
threshold_yaw	0.25	Face Angle threshold, if the Angle is greater than this threshold, quality will be forced to set to 0. (One unit is 90 degrees)
threshold_pitch	0.25	
threshold_roll	0.25	
fast_mode_interval	10	FAST mode capture interval. [Unit: frame]
fast_mode_capture_num	3	Number of captures in FAST mode.
cycle_mode_interval	20	CYCLE mode Snapshot interval. [Unit: frame]
auto_mode_time_limit	0	Duration of the last output in AUTO mode. [Unit: frame]
auto_mode_fast_cap	1	AUTO Mode Whether to output one quick snapshot.
capture_aligned_face	0	Whether the captured/captured face is corrected.

【Face Quality Assessment Algorithm】

#	Algorithm	Computation
0	Based on face size and Angle	1. Face Area Score 1. Definition of standard face size A_base = 112 * 112 2. Calculate the detected face area A_face = length * width 3. Calculate MIN(1.0, A_face/A_base) as a fraction 2. Face Pose Score 1. Calculate face Angle yaw, pitch and roll respectively and take their absolute values 2. Calculate 1 - (yaw + pitch + roll) / 3 as a score 3. Face Quality = Face Area Score * Face Pose Score
1	Eye distance based	1 Define standard pupil distance D = 80 2 Calculate the distance d between the eyes 3 Calculate MIN(1.0, d/D) as a fraction

5.2.3.1 CVI_TDL_APP_PersonCapture_Init

【Syntax】

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

【Description】

Initialize the data structure for human body capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	uint32_t	buffer_size	Face staging area size

5.2.3.2 CVI_TDL_APP_PersonCapture_QuickSetUp

【Syntax】

```
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);
```

【Description】

Quickly set up human body capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	const char*	od_model_name	Model name for human detection
Input	const char*	od_model_path	Path of human detection model
Input	const char*	reid_model_path	Path of ReID model

5.2.3.3 CVI_TDL_APP_FaceCapture_GetDefaultConfig

【Syntax】

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

【Description】

Get the preset parameters for human body capture.

【Parameter】

	Data Type	Parameter	Description
Output	person_capture_config_t*	cfg	Human body capture parameters

5.2.3.4 CVI_TDL_APP_PersonCapture_SetConfig

【Syntax】

```
CVI_S32 CVI_TDL_APP_PersonCapture_SetConfig(const cvitdl_app_handle_t handle,␣
↪person_capture_config_t *cfg);
```

【Description】

Setting the parameters for the person capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	person_capture_config_t*	cfg	The parameters for human pose capture

5.2.3.5 CVI_TDL_APP_PersonCapture_SetMode

【Syntax】

```
CVI_S32 CVI_TDL_APP_PersonCapture_SetMode(const cvitdl_app_handle_t handle,␣
↪capture_mode_e mode);
```

【Description】

Setting the mode of person capture

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	capture_mode_e	mode	Person capture mode

5.2.3.6 CVI_TDL_APP_PersonCapture_Run

【Syntax】

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

【Description】

Perform human body capture

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	VIDEO_FRAME_INFO_S*	frame	Input image

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】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input
Input	VIDEO_FRAME_INFO_S*	frame	Input image

5.2.3.8 CVI_TDL_APP_PersonCapture_CleanAll

【Syntax】

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

【Description】

Clear all data in the temporary storage area for human form capture.

【Parameter】

	Data Type	Parameter	Description
Input	cvitdl_app_handle_t	handle	Handle pointer input

6 Data Types

6.1 CVI_TDL_Core

6.1.1 CVI_TDL_SUPPORTED_MODEL_E

【Description】

This enum defines all Deep Learning Models in the TDL SDK.

The following table shows each model Id and its corresponding model functionality:

Model Index	Model ID	Description
0	CVI_TDL_SUPPORTED_MODEL_RETINAFACE	Face detection
1	CVI_TDL_SUPPORTED_MODEL_RETINAFACE_IR	IR Face detection(RetinaFace)
2	CVI_TDL_SUPPORTED_MODEL_RETINAFACE_HARDHAT	Hard hat detection(RetinaFace)
3	CVI_TDL_SUPPORTED_MODEL_SCRFDFACE	Face detection(ScrFD Face)
4	CVI_TDL_SUPPORTED_MODEL_THERMALFACE	Thermal face detection
5	CVI_TDL_SUPPORTED_MODEL_THERMALPERSON	Thermal human type detection
6	CVI_TDL_SUPPORTED_MODEL_FACEATTRIBUTE	Face attributes and face recognition
7	CVI_TDL_SUPPORTED_MODEL_FACERECOGNITION	FaceRecognition
8	CVI_TDL_SUPPORTED_MODEL_MASKFACERECOGNITION	FaceRecognition by wearing a mask
9	CVI_TDL_SUPPORTED_MODEL_FACEQUALITY	Face quality
10	CVI_TDL_SUPPORTED_MODEL_MASKCLASSIFICATION	Face mask recognition
11	CVI_TDL_SUPPORTED_MODEL_HANDCLASSIFICATION	Gesture recognition

continues on next page

Table 6.1 – continued from previous page

Model Index	Model ID	Description
12	CVI_TDL_SUPPORTED_MODEL _HAND_KEYPOINT	Hand keypoints detection
13	CVI_TDL_SUPPORTED_MODEL _HAND_KEYPOINT_CLASSIFICATION	Hand keypoints recognition
14	CVI_TDL_SUPPORTED_MODEL _LIVENESS	Binocular vivisection recognition
15	CVI_TDL_SUPPORTED_MODEL _HAND_DETECTION	Hand detection
16	CVI_TDL_SUPPORTED_MODEL _MOBILEDETV2_PERSON_VEHICLE	human type and vehicle detection
17	CVI_TDL_SUPPORTED_MODEL _MOBILEDETV2_VEHICLE	Vehicle detection
18	CVI_TDL_SUPPORTED_MODEL _MOBILEDETV2_PEDESTRIAN	Pedestrian detection
19	CVI_TDL_SUPPORTED_MODEL _MOBILEDETV2_PERSON_PETS	Cat, dog and human type detection
20	CVI_TDL_SUPPORTED_MODEL _MOBILEDETV2_COCO80	80 types of object detection
21	CVI_TDL_SUPPORTED_MODEL _YOLOV3	80 types of object detection
22	CVI_TDL_SUPPORTED_MODEL _YOLOV5	80 types of object detection
23	CVI_TDL_SUPPORTED_MODEL _YOLOX	80 types of object detection
24	CVI_TDL_SUPPORTED_MODEL _OSNET	Pedestrian rerecognition
25	CVI_TDL_SUPPORTED_MODEL _SOUNDCLASSIFICATION	Sound recognition
26	CVI_TDL_SUPPORTED_MODEL _SOUNDCLASSIFICATION_V2	Sound recognition V2
27	CVI_TDL_SUPPORTED_MODEL _WPODNET	License plate detection
28	CVI_TDL_SUPPORTED_MODEL _LPRNET_TW	License plate recognition in Taiwan
29	CVI_TDL_SUPPORTED_MODEL _LPRNET_CN	License plate recognition in Mainland China
30	CVI_TDL_SUPPORTED_MODEL _DEEPLABV3	Semantic segmentation
31	CVI_TDL_SUPPORTED_MODEL _ALPHAPOSE	Human keypoints detection
32	CVI_TDL_SUPPORTED_MODEL _EYECLASSIFICATION	Closed eye recognition
33	CVI_TDL_SUPPORTED_MODEL _YAWNCLASSIFICATION	Yawn recognition
34	CVI_TDL_SUPPORTED_MODEL _FACELANDMARKER	Face keypoints detection

continues on next page

Table 6.1 – continued from previous page

Model Index	Model ID	Description
35	CVI_TDL_SUPPORTED_MODEL _FACELANDMARKERDET2	Face keypoints detection V2
36	CVI_TDL_SUPPORTED_MODEL _INCAROBJECTDETECTION	Vehicle object recognition
37	CVI_TDL_SUPPORTED_MODEL _SMOKECLASSIFICATION	Smoking recognition
38	CVI_TDL_SUPPORTED_MODEL _FACEMASKDETECTION	Mask face detection
39	CVI_TDL_SUPPORTED_MODEL _IRLIVENESS	IR liveness detection
40	CVI_TDL_SUPPORTED_MODEL _PERSON_PETS_DETECTION	Human type, cat and dog detection
41	CVI_TDL_SUPPORTED_MODEL _PERSON_VEHICLE_DETECTION	Human type and Vehicle detection
42	CVI_TDL_SUPPORTED_MODEL _HAND_FACE_PERSON_DETECTION	Hand, face and Human type detection
43	CVI_TDL_SUPPORTED_MODEL _HEAD_PERSON_DETECTION	Hand and Human type detection
44	CVI_TDL_SUPPORTED_MODEL _YOLOV8POSE	Pose detection
45	CVI_TDL_SUPPORTED_MODEL _SIMCC_POSE	Pose detection
46	CVI_TDL_SUPPORTED_MODEL _LANDMARK_DET3	Face keypoints detection

Here is a table of model IDs, corresponding model files, and the inference functions used:

Model Index	Inference Function	Model file
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

continues on next page

Table 6.2 – continued from previous page

Model Index	Inference Function	Model file
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
33	CVI_TDL_YawnClassification	yawn_v1_bf16.cvimodel
34	CVI_TDL_FaceLandmarker	face_landmark_bf16.cvimodel

continues on next page

Table 6.2 – continued from previous page

Model Index	Inference Function	Model file
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

【Description】

This enum defines the categories of object detection, with each category belonging to a group.

Category	Category Group
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	
CVI_TDL_DET_TYPE_DOG	
CVI_TDL_DET_TYPE_HORSE	
CVI_TDL_DET_TYPE_SHEEP	
CVI_TDL_DET_TYPE_COW	

continues on next page

Table 6.3 – continued from previous page

Category	Category Group
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	
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	

continues on next page

Table 6.3 – continued from previous page

Category	Category Group
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

【Description】

This enum defines the object category groups.

Category Group	Description
CVI_TDL_DET_GROUP_ALL	Full class
CVI_TDL_DET_GROUP_PERSON	Human form
CVI_TDL_DET_GROUP_VEHICLE	Means of transportation
CVI_TDL_DET_GROUP_OUTDOOR	Outdoors
CVI_TDL_DET_GROUP_ANIMAL	Animal
CVI_TDL_DET_GROUP_ACCESSORY	Accessories
CVI_TDL_DET_GROUP_SPORTS	Movement
CVI_TDL_DET_GROUP_KITCHEN	Kitchen
CVI_TDL_DET_GROUP_FOOD	Food
CVI_TDL_DET_GROUP_FURNITURE	Furniture
CVI_TDL_DET_GROUP_ELECTRONIC	Electronic equipment
CVI_TDL_DET_GROUP_APPLIANCE	Appliance
CVI_TDL_DET_GROUP_INDOOR	Indoor articles
CVI_TDL_DET_GROUP_MASK_HEAD	Custom category
CVI_TDL_DET_GROUP_MASK_START	Custom category start
CVI_TDL_DET_GROUP_MASK_END	Custom category end

6.1.4 feature_type_e

【enum】

Value	Parameter	Description
0	TYPE_INT8	int8_t feature type
1	TYPE_UINT8	uint8_t feature type
2	TYPE_INT16	int16_t feature type
3	TYPE_UINT16	uint16_t feature type
4	TYPE_INT32	int32_t feature type
5	TYPE_UINT32	uint32_t feature type
6	TYPE_BF16	bf16 feature type
7	TYPE_FLOAT	float feature type

6.1.5 meta_rescale_type_e

【enum】

Value	Parameter	Description
0	RESCALE_UNKNOWN	Unknown
1	RESCALE_NOASPECT	Direct adjustment without proportional scaling
2	RESCALE_CENTER	Padding around all sides
3	RESCALE_RB	Padding on the bottom right side

6.1.6 cvtdl_bbox_t

Data Type	Parameter	Description
float	x1	The x value of the coordinate at the top left of the detection frame
float	y1	The y value of the coordinate on the upper left of the detection frame
float	x2	The x value of the lower right coordinate of the detection box
float	y2	The y value of the lower right coordinate of the detection box
float	score	The confidence level of the detection box

6.1.7 cvtdl_feature_t

Data Type	Parameter	Description
int8_t*	ptr	Address
uint32_t	size	Characteristic dimension
feature_type_e	type	Characteristic pattern

6.1.8 cvtdl_pts_t

Data Type	Parameter	Description
float*	x	Coordinate x
float*	y	Coordinate y
uint32_t	size	Number of coordinate points

6.1.9 cvtdl_4_pts_t

Data Type	Parameter	Description
float	x[4]	The x-coordinate values of the four coordinate points
float	y[4]	The y coordinate values of the four coordinate points

6.1.10 cvtdl_vpssconfig_t

Data Type	Parameter	Description
VPSS_SCALE_COEF_E	chn_coeff	Rescale mode
VPSS_CHN_ATTR_S	chn_attr	VPSS attribute data

6.1.11 cvtdl_tracker_t

Data Type	Parameter	Description
uint32_t	size	Number of trace messages
cvtdl_tracker_info_t*	info	Trace message structure

6.1.12 cvtdl_tracker_info_t

Data Type	Parameter	Description
cvtdl_trk_state_type_t	state	Tracking state
cvtdl_bbox_t	bbox	Tracking the Bounding Box of predictions

6.1.13 cvtdl_trk_state_type_t

【enum】

Value	Parameter	Description
0	CVI_TRACKER_NEW	The tracking status is new
1	CVI_TRACKER_UNSTABLE	The tracking state is unstable
2	CVI_TRACKER_STABLE	The tracking state is stable

6.1.14 cvtdl_deepsort_config_t

Data Type	Parameter	Description
float	max_distance_iou	Maximum IOU distance for BBox matching
float	ma x_distance_consine	Maximum consine distance during Feature matching
int	max_unmatched_times _for_bbox_matching	The number of maximum unmatched times of the target participating in BBox matching
bool	enable_internal_FQ	Enable internal feature quality
cvtdl_kalman_filter_config_t	kfilter_conf	Kalman filter settings
cvtdl_kalman_tracker_config_t	ktracker_conf	Kalman tracker settings

6.1.15 cvtdl_kalman_filter_config_t

Data Type	Parameter	Description
bool	enable_X_constraint_0	Enable 0th X constraint
bool	enable_X_constraint_1	Enable 1st X constraint
float	X_constraint_min[8]	X Constraint lower bound
float	X_constraint_max[8]	X Constraint upper limit
bool	enable_bounding_stay	Keep boundaries
mahalanobis _confidence_e	confidence_level	Mahalanobis distance confidence
float	chi2_threshold	Chi-square threshold
float	Q_std_alpha[8]	Process Noise parameter
float	Q_std_beta[8]	Process Noise parameter
int	Q_std_x_idx[8]	Process Noise parameter
float	R_std_alpha[4]	Measurement Noise parameter
float	R_std_beta[4]	Measurement Noise parameter
int	R_std_x_idx[4]	Measurement Noise parameter

【Description】

For the motion state X of a tracked target···

Process Nose, Q, where

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

Measurement Nose, R, The formula for motion deviation is the same as follows:

6.1.16 cvtdl_kalman_tracker_config_t

Data Type	Parameter	Description
int	max_unmatched_num	Track the maximum number of lost targets
int	accreditation_threshold	The threshold at which the tracking state changes to stable
int	feature_budget_size	Save the maximum number of tracking target features
int	feature_update_interval	Time interval of updating feature
bool	enable_QA_feature_init	Enable QA feature initialization
bool	enable_QA_feature_update	Enable QA feature updates
float	feature_init_quality_threshold	Feature initialization quality threshold
float	feature_update_quality_threshold	Feature update quality threshold
float	P_std_alpha[8]	Initial Covariance parameter
float	P_std_beta[8]	Initial Covariance parameter
int	P_std_x_idx[8]	Initial Covariance parameter

【Description】

Initial Covariance, P, The formula for updating the initial covariance is similar to that of the process noise covariance:

6.1.17 cvtdl_liveness_ir_position_e

【enum】

Value	Parameter	Description
0	LIVENESS_IR_LEFT	The IR lens is to the left of the RGB lens
1	LIVENESS_IR_RIGHT	The IR lens is to the right of the RGB lens

6.1.18 cvtdl_head_pose_t

Data Type	Parameter	Description
float	yaw	Yaw angle
float	pitch	Pitch angle
float	roll	Angle of roll
float	facialUnitNormalVector[3]	The face is oriented towards the bearing

6.1.19 cvtdl_face_info_t

Data Type	Parameter	Description
char	name[128]	Face name
uint64_t	unique_id	Face ID
cvtdl_bbox_t	bbox	Face detection frame
cvtdl_pts_t	pts	Face feature point
cvtdl_feature_t	feature	Facial features
cvtdl_face_emotion_e	emotion	Expression
cvtdl_face_gender_e	gender	Gender
cvtdl_face_race_e	race	Race
float	score	Score
float	age	Age
float	liveness_score	Live probability
float	hardhat_score	Hard hat score
float	mask_score	Probability of wearing face mask
float	recog_score	Recognition score
float	face_quality	Human face quality
float	pose_score	Pose score
float	pose_score1	Pose score 1
float	sharpness_score	Sharpness score
float	blurness	Blurness
cvtdl_head_pose_t	head_pose	Facial angle information
int	track_state	Track state

6.1.20 cvtdl_face_t

Data Type	Parameter	Description
uint32_t	size	Number of faces
uint32_t	width	The width of the original picture
uint32_t	height	The height of the original picture
meta_rescale_type_e*	rescale_type	Rescale type
cvtdl_face_info_t*	info	Face synthesis information
cvtdl_dms_t*	dms	Driver monitoring system

6.1.21 cvtdl_pose17_meta_t

Data Type	Parameter	Description
float	x[17]	x-coordinate of 17 skeletal keypoints
float	y[17]	y-coordinate of 17 skeletal keypoints
float	score[17]	Predicted confidence value of 17 skeletal keypoints

6.1.22 cvtdl_vehicle_meta

Data Type	Parameter	Description
cvtdl_4_pts_t	license_pts	Coordinates of the 4 corners of the license plate
cvtdl_bbox_t	license_bbox	Bounding box of the license plate
char[125]	license_char	License plate number

【Description】

The four corner coordinates of the license plate are upper left, upper right, lower right to lower left in order.

6.1.23 cvtdl_class_filter_t

Data Type	Parameter	Description
uint32_t*	preserved_class_ids	The category id to keep
uint32_t	num_preserved_classes	The number of category ids to keep

6.1.24 cvtdl_dms_t

Data Type	Parameter	Description
float	reye_score	Right eye opening and closing score
float	leye_score	Left eye opening and closing score
float	yawn_score	Mouth closure score
float	phone_score	Talking on the phone score
float	smoke_score	Smoking score
cvtdl_pts_t	landmarks_106	106 feature points
cvtdl_pts_t	landmarks_5	Five feature points
cvtdl_head_pose_t	head_pose	Face angle calculated from 106 feature points
cvtdl_dms_od_t	dms_od	Object detection results in the vehicle

6.1.25 cvtdl_dms_od_t

Data Type	Parameter	Description
uint32_t	size	There are a couple of things
uint32_t	width	Width
uint32_t	height	Height
meta_rescale_type_e	rescale_type	The shape of rescale
cvtdl_dms_od_info_t*	info	Information about objects

6.1.26 cvtdl_dms_od_info_t

Data Type	Parameter	Description
char[128]	name	Name of object
int	classes	Category of objects
cvtdl_bbox_t	bbox	Object bounding Box

6.1.27 cvtdl_face_emotion_e

【Description】

Emotion

Emotion	Description
EMOTION_UNKNOWN	unknown
EMOTION_HAPPY	Be happy
EMOTION_SURPRISE	Surprise
EMOTION_FEAR	Fear of
EMOTION_DISGUST	Aversion to
EMOTION_SAD	Be sad
EMOTION_ANGER	Get angry
EMOTION_NEUTRAL	Nature

6.1.28 cvtdl_face_race_e

Race	Description
RACE_UNKNOWN	Unknown
RACE_CAUCASIAN	The Caucasian
RACE_BLACK	Black people
RACE_ASIAN	The Asians

6.1.29 cvtdl_pedestrian_meta

Data Type	Parameter	Description
cvtdl_pose17_meta_t	pose17	Human 17 keypoints
bool	fall	Whether fallen or not

6.1.30 cvtdl_object_info_t

Data Type	Parameter	Description
char	name	Object class name
uint64_t	unique_id	Unique id
cvtdl_box_t	bbox	Bounding box
cvtdl_feature_t	feature	Feature of objects
int	classes	ID of category
cvtdl_vehicle_meta	vehicle_property	Property of vehicle
cvtdl_pedestrian_meta	pedestrian_property	Attributes of pedestrians
int	track_state	Track state

6.1.31 cvtdl_object_t

Data Type	Parameter	Description
uint32_t	size	The number of objects in info
uint32_t	width	The width of the original image
uint32_t	height	The height of the original picture
uint32_t	entry_num	Number of entries
uint32_t	miss_num	Number of miss
meta_rescale_type_e	rescale_type	resize method used in model preprocessing
cvtdl_object_info_t*	info	Object information

6.1.32 cvtdl_handpose21_meta_t

Data Type	Parameter	Description
float	xn[21]	normalize x points
float	x[21]	x points
float	yn[21]	normalize y points
float	y[21]	y points
float	bbox_x	the x-coordinate of the box
float	bbox_y	the y-coordinate of the box
float	bbox_w	The width of box
float	bbox_h	The height of box
int	label	Gesture category
float	score	Gesture score

6.1.33 cvtdl_handpose21_meta_ts

Data Type	Parameter	Description
uint32_t	size	The number of hands detected
uint32_t	width	The width of image
uint32_t	height	The height of image
cvtdl_handpose21_meta_t*	info	Hand keypoints

6.1.34 Yolov5PreParam

Data Type	Parameter	Description
float	factor[3]	Scaling factor
float	mean[3]	Image mean
meta_rescale_type_e	rescale_type	Scaling mode
bool*	pad_reverse	Reverse padding
bool*	keep_aspect_ratio	Keep aspect ratio
bool*	use_quantize_scale	Qantization scaling
bool*	use_crop	Crop to resize an image
VPSS_SCALE_COEF_E*	resize_method	Scaling method
PIXEL_FORMAT_E*	format	Image format

6.1.35 YOLOV5AlgParam

Data Type	Parameter	Description
uint32_t	anchors[3][3][2]	model anchors
float	conf_thresh	confidence threshold
float	nms_thresh	RMS Threshold

6.2 CVI_TDL_Service

6.2.1 cvtdl_service_feature_matching_e

【Description】

The method used for feature comparison calculation, currently only supports Cosine Similarity.

【Member】

Parameter	Description
COS_SIMILARITY	Cosine similarity

6.2.2 cvtdl_service_feature_array_t

【Description】

Feature array, this structure contains the pointer to the feature array, length, number of features, and feature type information.

This structure needs to be passed in when registering the feature library.

【Member】

Data Type	Parameter	Description
int8_t*	ptr	Feature array pointer
uint32_t	feature_length	Single characteristic length
uint32_t	data_num	Number of features
feature_type_e	type	Type of feature

6.2.3 cvtdl_service_brush_t

【Description】

Brush structure for drawing, which specifies the RGB color and brush size to be used.

【Member】

Data Type	Parameter	Description
Inner structure	color	The RGB value to use
uint32_t	size	Brush size

6.2.4 cvtdl_area_detect_e

【enum】

Data Type	Parameter	Description
0	UNKNOWN	The int8_t feature type
1	NO_INTERSECT	Not intersect
2	ON_LINE	On line
3	CROSS_LINE_POS	Forward cross
4	CROSS_LINE_NEG	Negative cross
5	INSIDE_POLYGON	Inside the polygon
6	OUTSIDE_POLYGON	Outside the polygon

7 Error Codes

Error Code	Macro Definition	Description
0xFFFFFFFF	CVI_TDL_FAILURE	API call failed
0xC0010101	CVI_TDL_ERR_INVALID_MODEL_PATH	Incorrect model path
0xC0010102	CVI_TDL_ERR_OPEN_MODEL	Failed to start the model
0xC0010103	CVI_TDL_ERR_CLOSE_MODEL	Failed to close the model
0xC0010104	CVI_TDL_ERR_GET_VPSS_CHN_CONFIG	Failed to obtain VPSS CHN Settings
0xC0010105	CVI_TDL_ERR_INFERENCE	Model inference failure
0xC0010106	CVI_TDL_ERR_INVALID_ARGS	Incorrect parameters
0xC0010107	CVI_TDL_ERR_INIT_VPSS	Failed to initialize VPSS
0xC0010108	CVI_TDL_ERR_VPSS_SEND_FRAME	Failed to send Frame to VPSS
0xC0010109	CVI_TDL_ERR_VPSS_GET_FRAME	Failed to get Frame from VPSS
0xC001010A	CVI_TDL_ERR_MODEL_INITIALIZED	The model is not open
0xC001010B	CVI_TDL_ERR_NOT_YET_INITIALIZED	The function is not initialized
0xC001010C	CVI_TDL_ERR_NOT_YET_IMPLEMENTED	The function has not been implemented
0xC001010D	CVI_TDL_ERR_ALLOC_ION_FAIL	Failed to allocate ION memory
0xC0010201	CVI_TDL_ERR_MD_OPERATION_FAILED	Failed to run Motion Detection