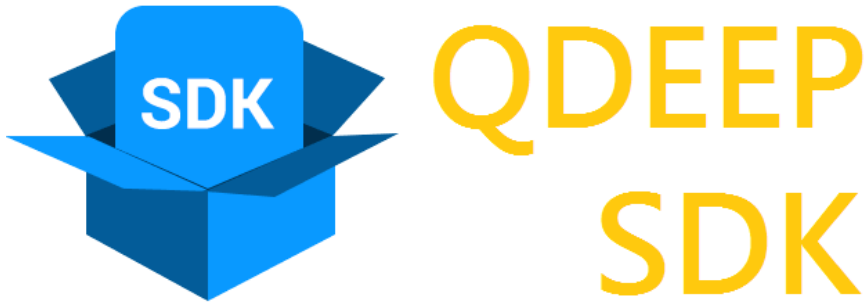
version: 202.0

1 SDK Introduction

Introduction



QDEEP SDK

The QDEEP SDK technology provides the user a simple way to use deep learning (AI),can use different model to detect different object.

1.1 Overview

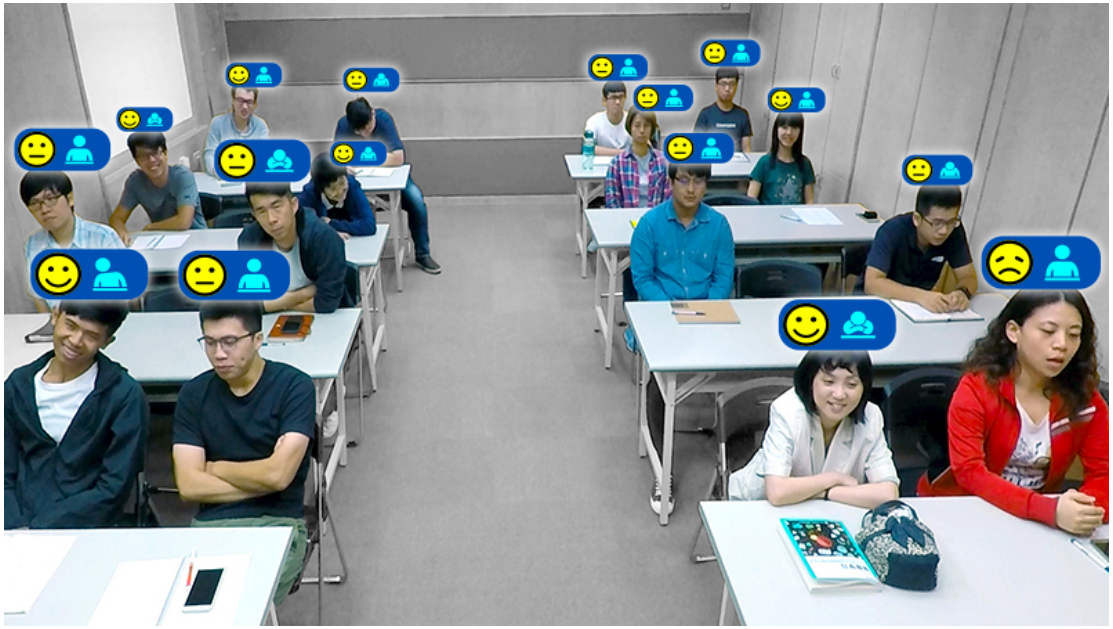
The QDEEP SDK more high-level APIs for a developer to easily access our capture card to use deep learning and to develop a rich client application. It supports following features:

- QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_GENERAL_DEFECT_DETECTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_PCB_DEFECT_DETECTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_GAUGE_READER_DETECTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_BOAT
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_DEPTH_MAP_3D_EX
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_DRIVING_DISTRACTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_EDUCATION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_HEAD_BODY
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_5_KEYPOINTS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_3D_EX
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_FACE_BEAUTY_EX
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_MODAL_ANALYTICS_EX
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_FLAME
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HAND_LANDMARK_21_KEYPOINTS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_BLURRING
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_REMOVAL
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_AUTO_FRAMING
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_FACE_LAYOUT
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_SPEAKER_TRACKING
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_HANDWRITE_EXTRACTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SAFETY_INSPECTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SKELETON_17_KEYPOINTS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SKELETON_136_KEYPOINTS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_PARKING
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_LAW_ENFORCEMENT
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_MISSING_OBJECT
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_OPTICAL_CHARACTER_RECOGNITION

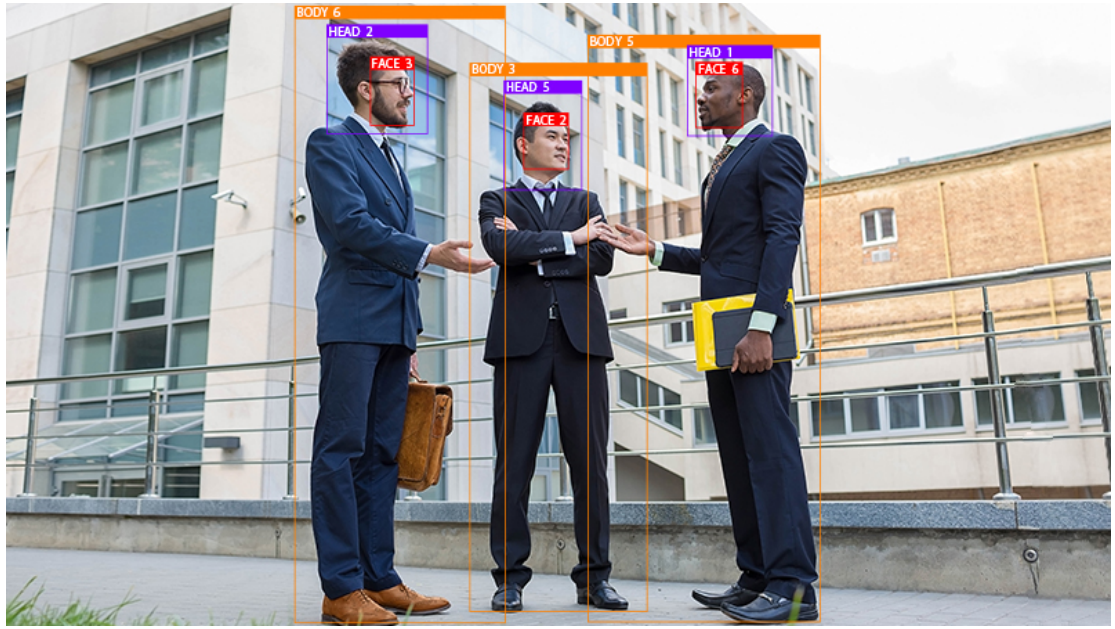
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_RETOUCHING
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_SUPER_RESOLUTION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_RETAIL_PRODUCT_RECOGNITION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_SEGMENTATION
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_04
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_08
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_UNATTENDED_OBJECT
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_XRAY_INSPECTION_SYSTEM
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_LITE
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MEDICAL_GRADE
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MULTI_LABELS
- QDEEP_OBJECT_DETECT_CONFIG_MODEL_NVIDIA_CLARA_AGX

1.1.1 QDEEP_OBJECT_DETECT_CONFIG_MODEL_DRIVING_DISTRACTION

1.1.2 QDEEP_OBJECT_DETECT_CONFIG_MODEL_EDUCATION

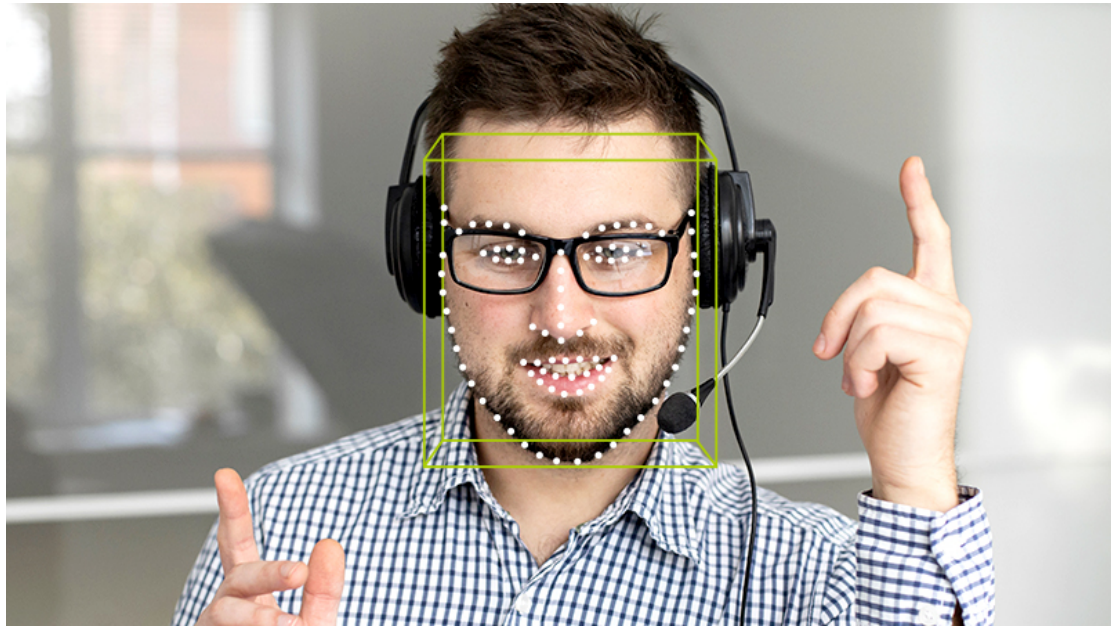


1.1.3 QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_HEAD_BODY



1.1.4 QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_5_KEYPOINTS

1.1.5 QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS



1.1.6 QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_FACE_BEAUTY_EX

1.1.7 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_BLURRING

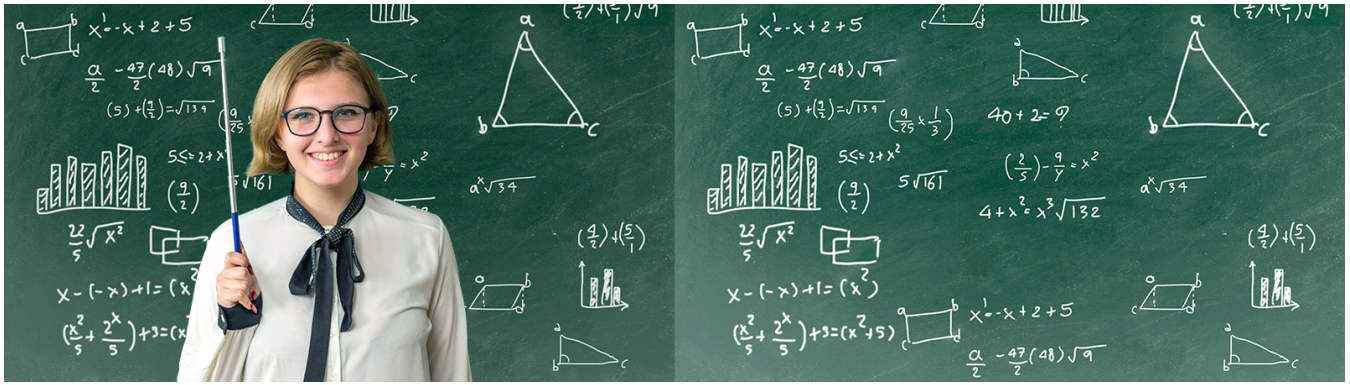
1.1.8 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_REMOVAL

1.1.9 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_AUTO_FRAMING

1.1.10 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_FACE_LAYOUT

1.1.11 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_SPEAKER_TRACKING

1.1.12 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_HANDWRITE_EXTRACTION



1.1.13 QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SKELETON_17_KEYPOINTS

1.1.14 QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_PARKING

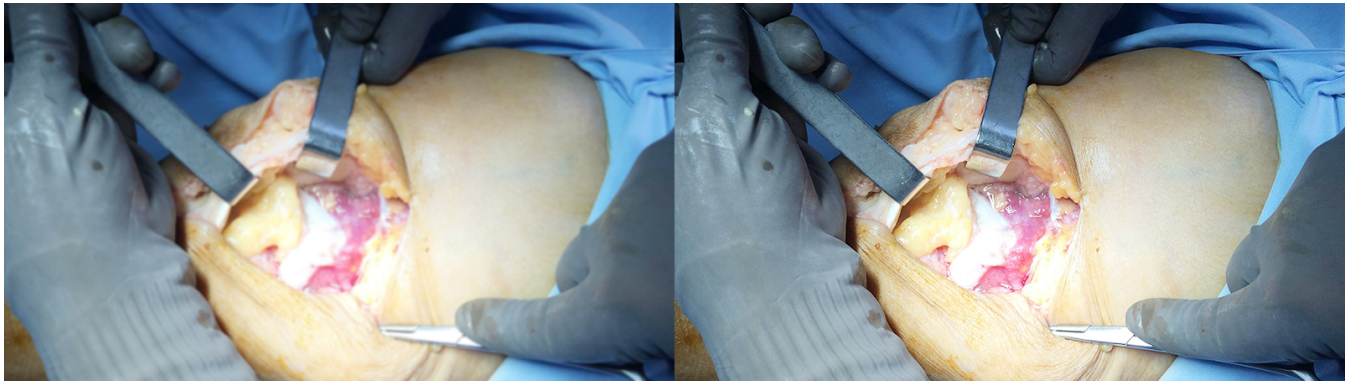
1.1.15 QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_LAW_ENFORCEMENT

1.1.16 QDEEP_OBJECT_DETECT_CONFIG_MODEL_MISSING_OBJECT

1.1.17 QDEEP_OBJECT_DETECT_CONFIG_MODEL_OPTICAL_CHARACTER_RECOGNITION

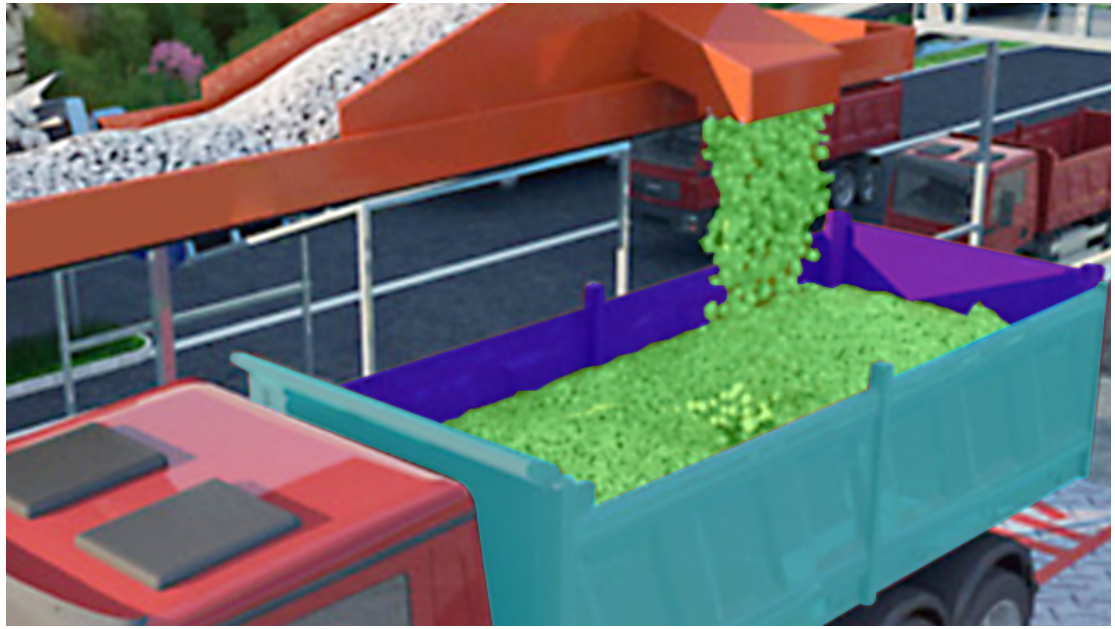
1.1.18 QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_RETOUCHING

1.1.19 QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_SUPER_RESOLUTION



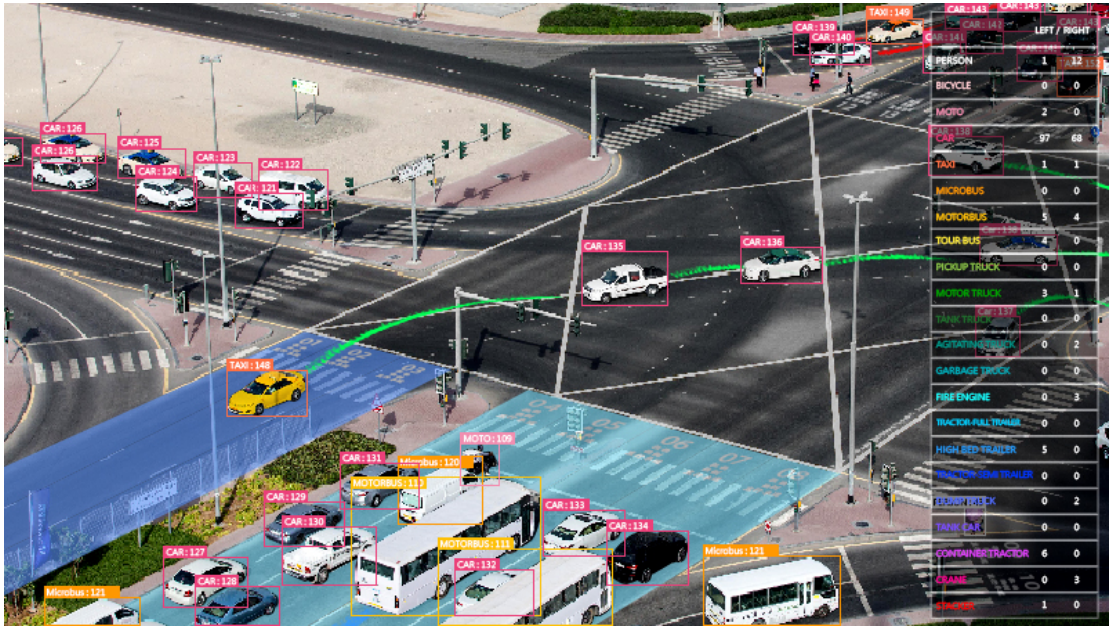
1.1.20 QDEEP_OBJECT_DETECT_CONFIG_MODEL_RETAIL_PRODUCT_RECOGNITION

1.1.21 QDEEP_OBJECT_DETECT_CONFIG_MODEL_SEGMENTATION

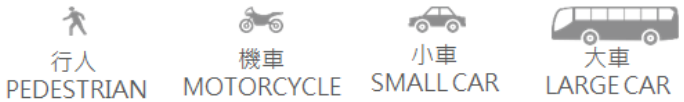


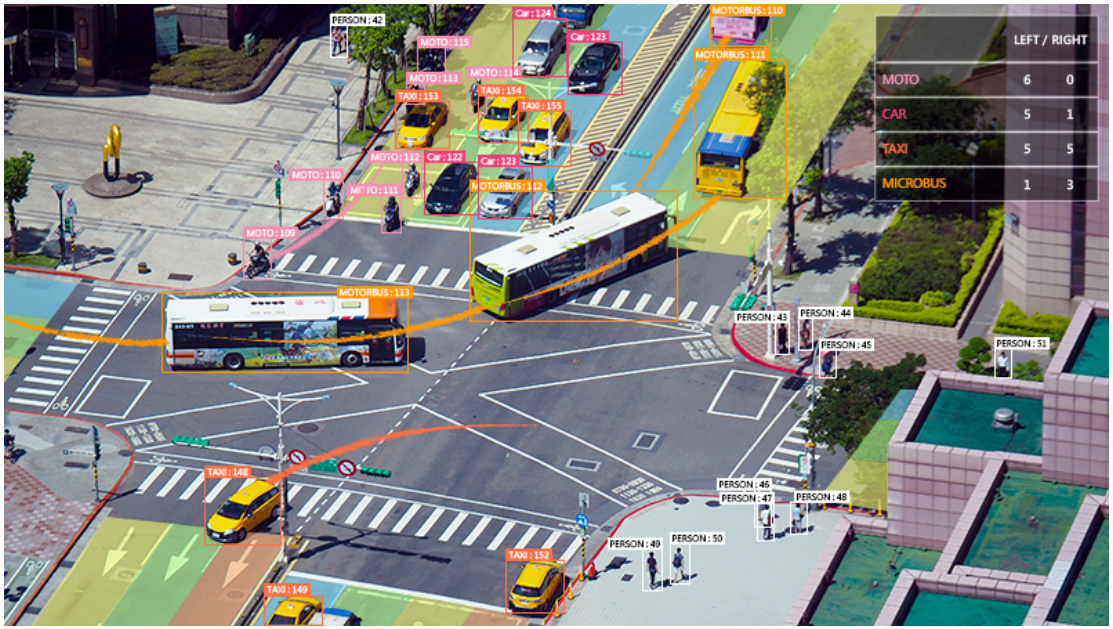
1.1.22 QDEEP_OBJECT_DETECT_CONFIG_MODEL_SMOG

1.1.23 QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC



1.1.24 QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_04

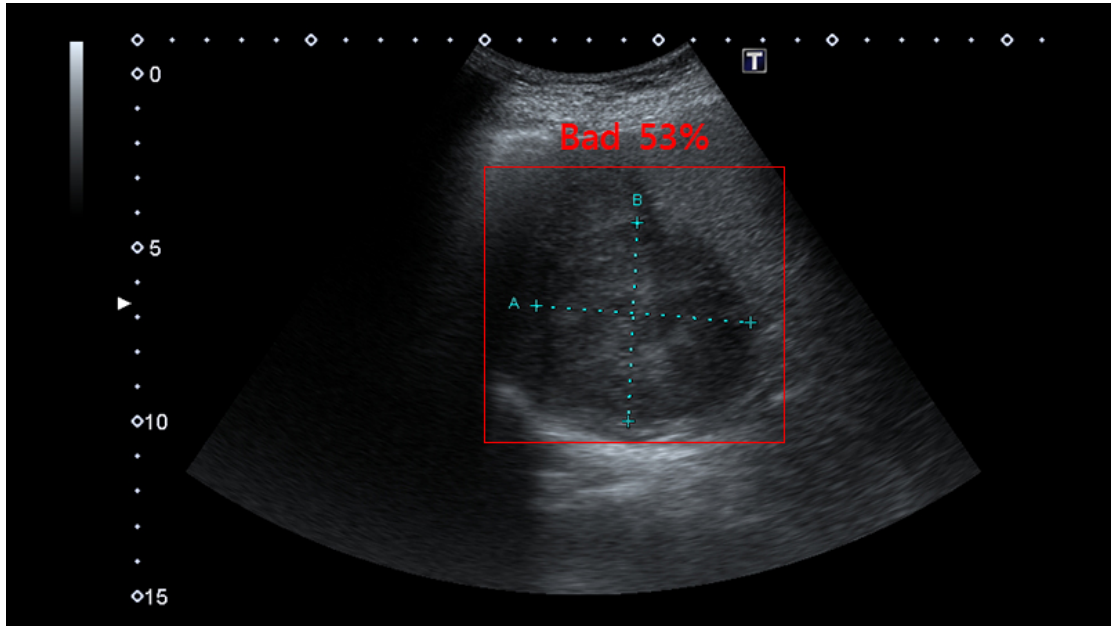




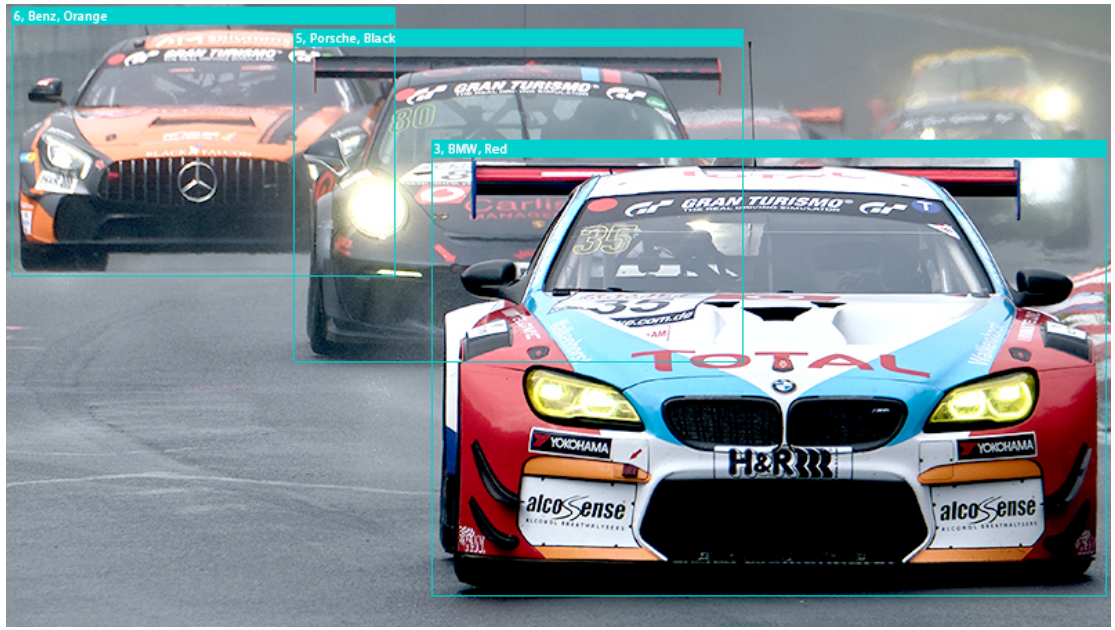
1.1.25 QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_08

1.1.26 QDEEP_OBJECT_DETECT_CONFIG_MODEL_UNATTENDED_OBJECT

1.1.27 QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MEDICAL_GRADE



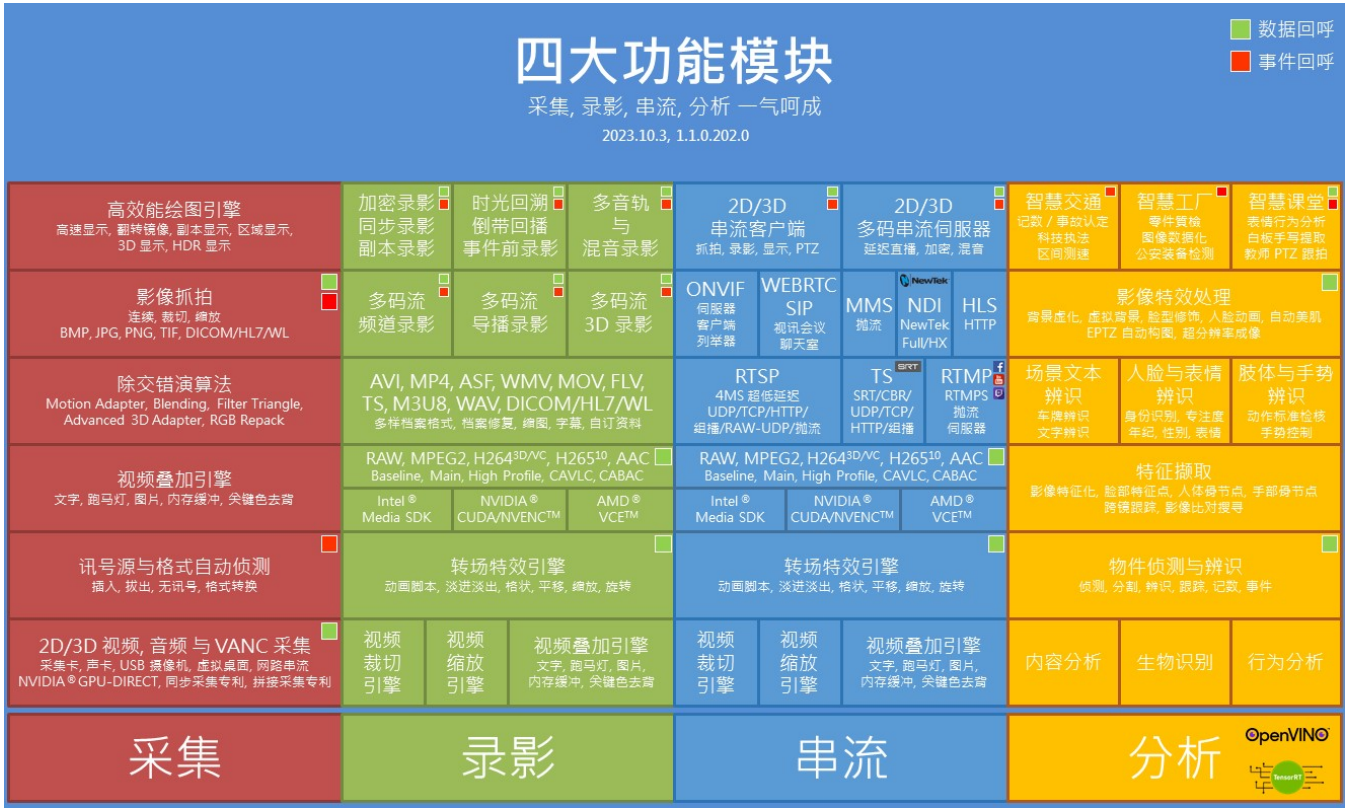
1.1.28 QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MULTI_LABELS

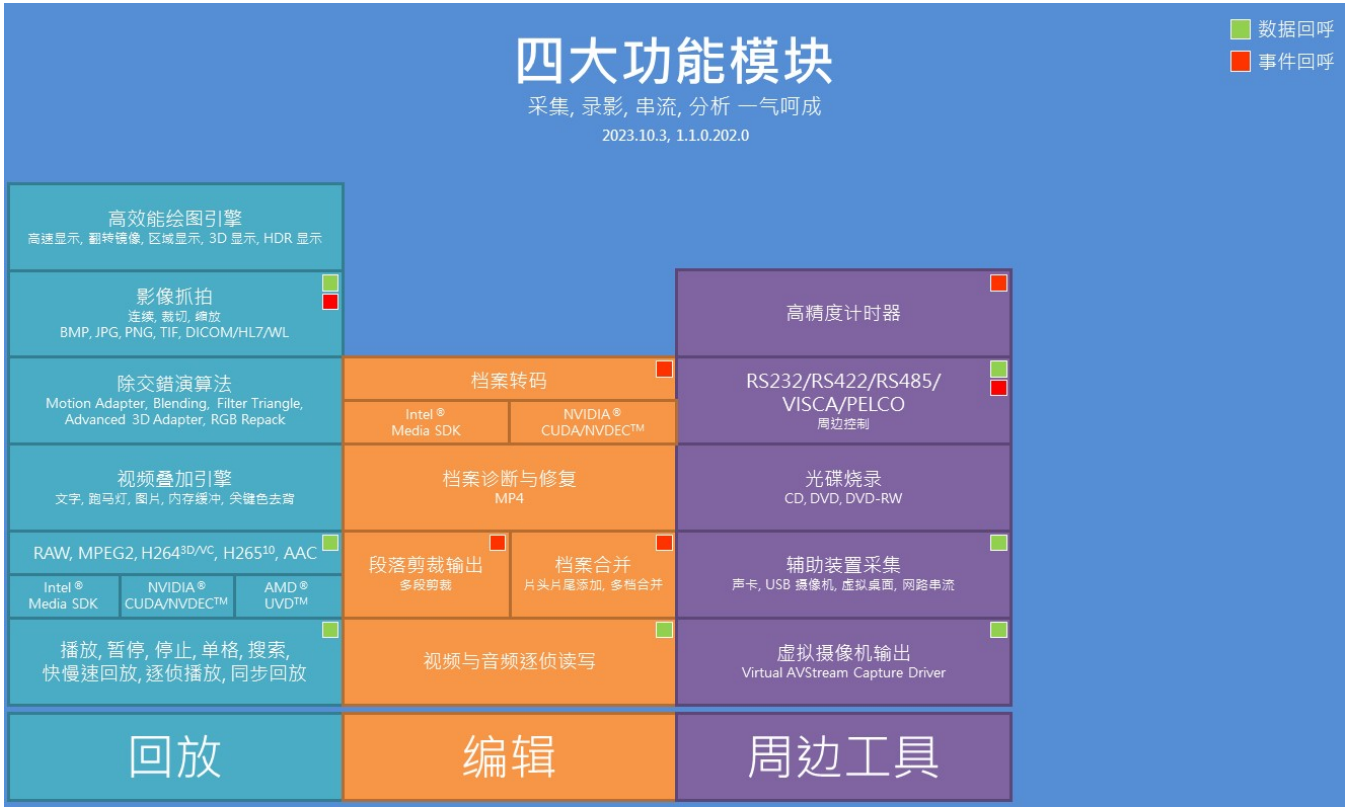


1.2 SDK Function Blocks

The QDEEP SDK is a **Object Detection, object tracking, human keypoints detection and face recognition in One Library.**

Note : When developers use QDEEP SDK Development, it must be an X64 OS.

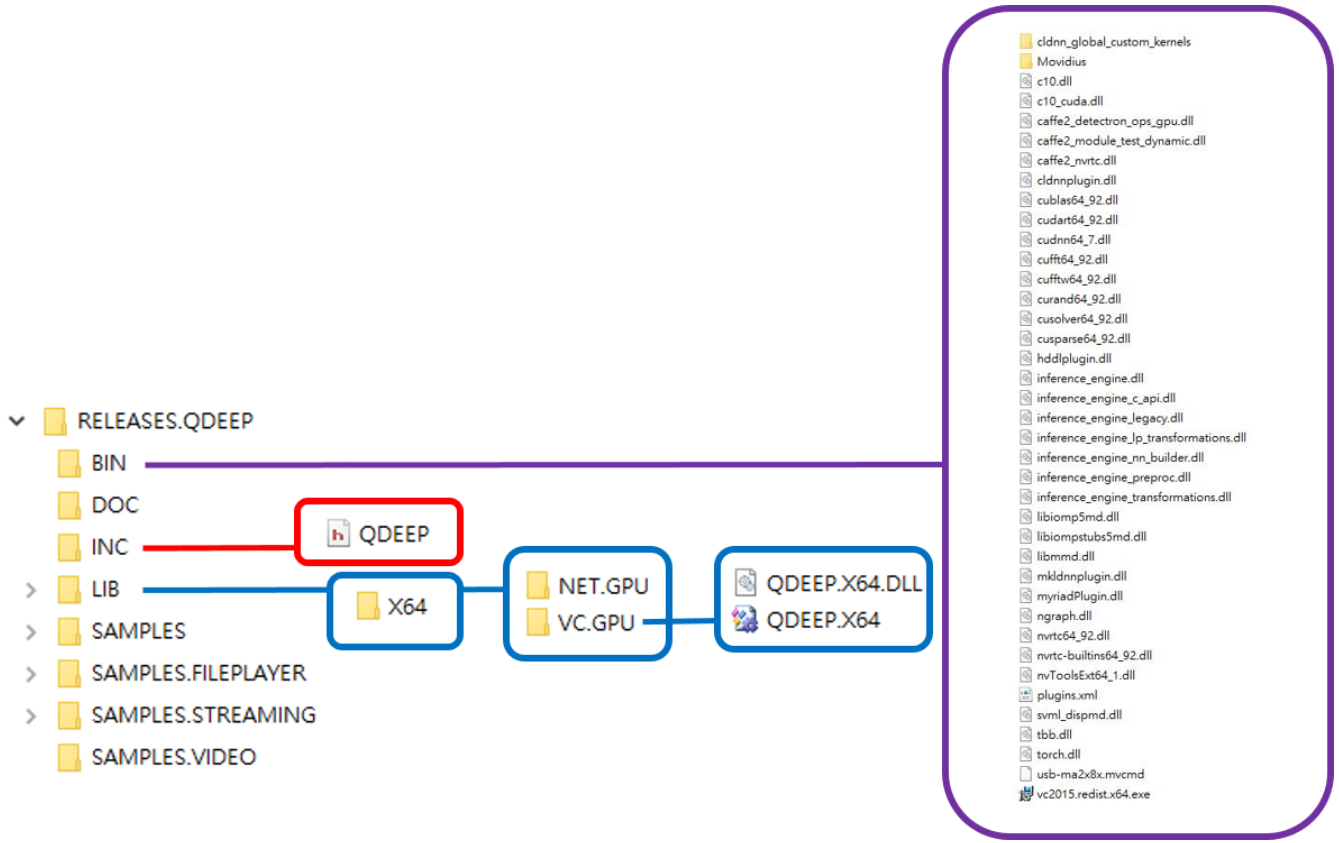


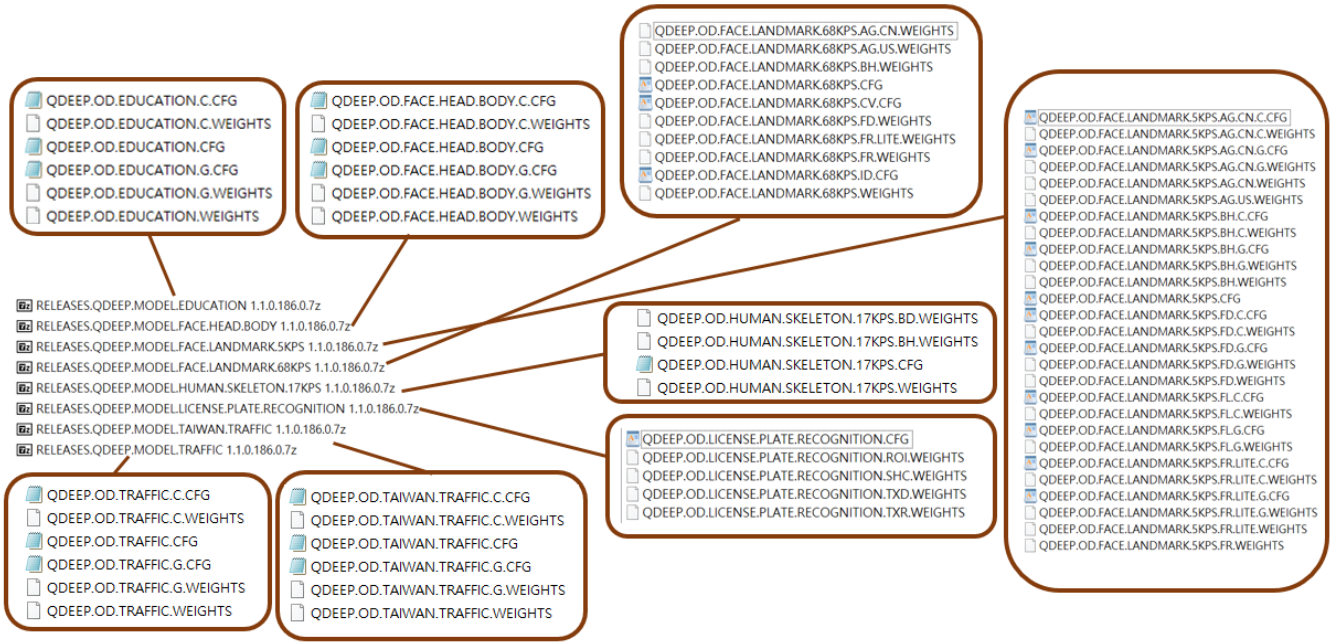

四大功能模块
采集, 录影, 串流, 分析 一气呵成
2023.10.3, 1.1.0.202.0
数据回呼
事件回呼
高效能绘图引擎
高速显示, 翻转镜像, 区域显示, 3D 显示, HDR 显示
影像抓拍
连续, 裁切, 缩放
BMP, JPG, PNG, TIF, DICOM/HL7/WL
除交錯演算法
Motion Adapter, Blending, Filter Triangle, Advanced 3D Adapter, RGB Repack
视频叠加引擎
文字, 跑马灯, 图片, 内存缓冲, 关键色去背
RAW, MPEG2, H264 3D/VC, H265 10, AAC
Intel ® Media SDK
NVIDIA ® CUDA/NVDEC™
AMD ® UVD™
播放, 暂停, 停止, 单格, 搜索, 快慢速回放, 逐侦播放, 同步回放
回放
档案转码
Intel ® Media SDK
NVIDIA ® CUDA/NVDEC™
档案诊断与修复
MP4
段落剪裁输出
多段剪裁
档案合并
片头片尾添加, 多档合并
视频与音频逐侦读写
编辑
高精度计时器
RS232/RS422/RS485/ VISCA/PELCO
周边控制
光碟烧录
CD, DVD, DVD-RW
辅助装置采集
声卡, USB 摄像机, 虚拟桌面, 网路串流
虚拟摄像机输出
Virtual AVStream Capture Driver
周边工具

1.3 Using the SDK

In this section, we will guide and teach users how step by step to use QDEEP SDK functions. The step-by-step instructions:

- Install Codec by executing **CODECS 1.1.0.exe**
  - Please make sure there is no *question marks* on the Video Graphics Array device icon.
- Unarchive **RELEASES.QDEEP 1.1.0.7z** to **QDEEP_SDK**
  - Please use **7-Zip** (*www.7-zip.org*) to extract .**7z** file.
- Setting QDEEP Header/Static/Run-time Libraries to your sample folder:
  - Copy **QDEEP.H** from **QDEEP_SDK\INC** to your project directory
  - Copy **QDEEP.X64.LIB** from **QDEEP_SDK\LIB\X64\VC.GPU** to your project directory
- Compile a sample code and produce an executable file
  - Copy all files to the same directory as your executable file from **QDEEP_SDK\BIN** to the directory of your executable file
  - According to functions needed, copy **.CFG**、**.WEIGHTS** from **RELEASES.QDEEP.MODEL.X** to the directory of your executable file
  - Copy **QDEEP.X64.DLL** from **QDEEP_SDK\LIB\X64\VC.GPU** to the directory of your executable file
  - Make sure the QDEEP.X64.DLL and your executable file in the same directory!
- Run your executable file


RELEASES.QDEEP
BIN
DOC
INC
LIB
SAMPLES
SAMPLES.FILEPLAYER
SAMPLES.STREAMING
SAMPLES.VIDEO
QDEEP
X64
NET.GPU
VC.GPU
QDEEP.X64.DLL
QDEEP.X64
cldnn_global_custom_kernels
Movidius
c10.dll
c10_cuda.dll
caffe2_detectron_ops_gpu.dll
caffe2_module_test_dynamic.dll
caffe2_nvrtc.dll
cldnnplugin.dll
cublas64_92.dll
cudart64_92.dll
cudnn64_7.dll
cufft64_92.dll
cufftw64_92.dll
curand64_92.dll
cusolver64_92.dll
cusparse64_92.dll
hddlplugin.dll
inference_engine.dll
inference_engine_c_api.dll
inference_engine_legacy.dll
inference_engine_lp_transformations.dll
inference_engine_nn_builder.dll
inference_engine_preproc.dll
inference_engine_transformations.dll
libiomp5md.dll
libiompstubs5md.dll
libmmd.dll
mkldnnplugin.dll
myriadPlugin.dll
ngraph.dll
nvrtc64_92.dll
nvrtc-builtins64_92.dll
nvToolsExt64_1.dll
plugins.xml
svml_dispmd.dll
tbb.dll
torch.dll
usb-ma2x8x.mvcmd
vc2015.redist.x64.exe


QDEEP.OD.EDUCATION.C.CFG
QDEEP.OD.EDUCATION.C.WEIGHTS
QDEEP.OD.EDUCATION.CFG
QDEEP.OD.EDUCATION.G.CFG
QDEEP.OD.EDUCATION.G.WEIGHTS
QDEEP.OD.EDUCATION.WEIGHTS
QDEEP.OD.FACE.HEAD.BODY.C.CFG
QDEEP.OD.FACE.HEAD.BODY.C.WEIGHTS
QDEEP.OD.FACE.HEAD.BODY.CFG
QDEEP.OD.FACE.HEAD.BODY.G.CFG
QDEEP.OD.FACE.HEAD.BODY.G.WEIGHTS
QDEEP.OD.FACE.HEAD.BODY.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.AG.CN.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.AG.US.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.BH.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.CFG
QDEEP.OD.FACE.LANDMARK.68KPS.CV.CFG
QDEEP.OD.FACE.LANDMARK.68KPS.FD.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.FR.LITE.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.FR.WEIGHTS
QDEEP.OD.FACE.LANDMARK.68KPS.ID.CFG
QDEEP.OD.FACE.LANDMARK.68KPS.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.AG.CN.C.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.AG.CN.C.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.AG.CN.G.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.AG.CN.G.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.AG.CN.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.AG.US.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.BH.C.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.BH.C.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.BH.G.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.BH.G.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.BH.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FD.C.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FD.C.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FD.G.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FD.G.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FD.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FL.C.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FL.C.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FL.G.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FL.G.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FR.LITE.C.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FR.LITE.C.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FR.LITE.G.CFG
QDEEP.OD.FACE.LANDMARK.5KPS.FR.LITE.G.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FR.LITE.WEIGHTS
QDEEP.OD.FACE.LANDMARK.5KPS.FR.WEIGHTS
QDEEP.OD.HUMAN.SKELETON.17KPS.BD.WEIGHTS
QDEEP.OD.HUMAN.SKELETON.17KPS.BH.WEIGHTS
QDEEP.OD.HUMAN.SKELETON.17KPS.CFG
QDEEP.OD.HUMAN.SKELETON.17KPS.WEIGHTS
QDEEP.OD.LICENSE.PLATE.RECOGNITION.CFG
QDEEP.OD.LICENSE.PLATE.RECOGNITION.ROI.WEIGHTS
QDEEP.OD.LICENSE.PLATE.RECOGNITION.SHC.WEIGHTS
QDEEP.OD.LICENSE.PLATE.RECOGNITION.TXD.WEIGHTS
QDEEP.OD.LICENSE.PLATE.RECOGNITION.TXR.WEIGHTS
RELEASES.QDEEP.MODEL.EDUCATION 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.FACE.HEAD.BODY 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.FACE.LANDMARK.5KPS 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.FACE.LANDMARK.68KPS 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.HUMAN.SKELETON.17KPS 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.LICENSE.PLATE.RECOGNITION 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.TAIWAN.TRAFFIC 1.1.0.186.0.7z
RELEASES.QDEEP.MODEL.TRAFFIC 1.1.0.186.0.7z
QDEEP.OD.TRAFFIC.C.CFG
QDEEP.OD.TRAFFIC.C.WEIGHTS
QDEEP.OD.TRAFFIC.CFG
QDEEP.OD.TRAFFIC.G.CFG
QDEEP.OD.TRAFFIC.G.WEIGHTS
QDEEP.OD.TRAFFIC.WEIGHTS
QDEEP.OD.TAIWAN.TRAFFIC.C.CFG
QDEEP.OD.TAIWAN.TRAFFIC.C.WEIGHTS
QDEEP.OD.TAIWAN.TRAFFIC.CFG
QDEEP.OD.TAIWAN.TRAFFIC.G.CFG
QDEEP.OD.TAIWAN.TRAFFIC.G.WEIGHTS
QDEEP.OD.TAIWAN.TRAFFIC.WEIGHTS

1.4 Platform and Thread Safety

SDK for X64 Windows Vista, Windows 7, Windows 8, Windows 10 and Windows Server 2008 are all supported by this QDEEP API calling can be used in the multiple threads and the multiple processes.

Intel CPU/ GPU / VPU MOVIDIUS of GPU type is only For **Windows 10**

1.5 SDK Easy Programming Guide

This section will give users a first glance of typical QDEEP SDK program samples.

1.5.1 Initialize Detector object APIs

```c
PVOID m_pDetector = NULL;

QDEEP_CREATE_OBJECT_DETECT(     QDEEP_GPU_TYPE_NVIDIA,
                                0,
                                QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_HEAD_BODY,
                                "QDEEP.OD.FACE.HEAD.BODY.CFG",
                                &m_pDetector);

QDEEP_START_OBJECT_DETECT(m_pDetector);
```

1.5.2 Uninitialize Detector object APIs

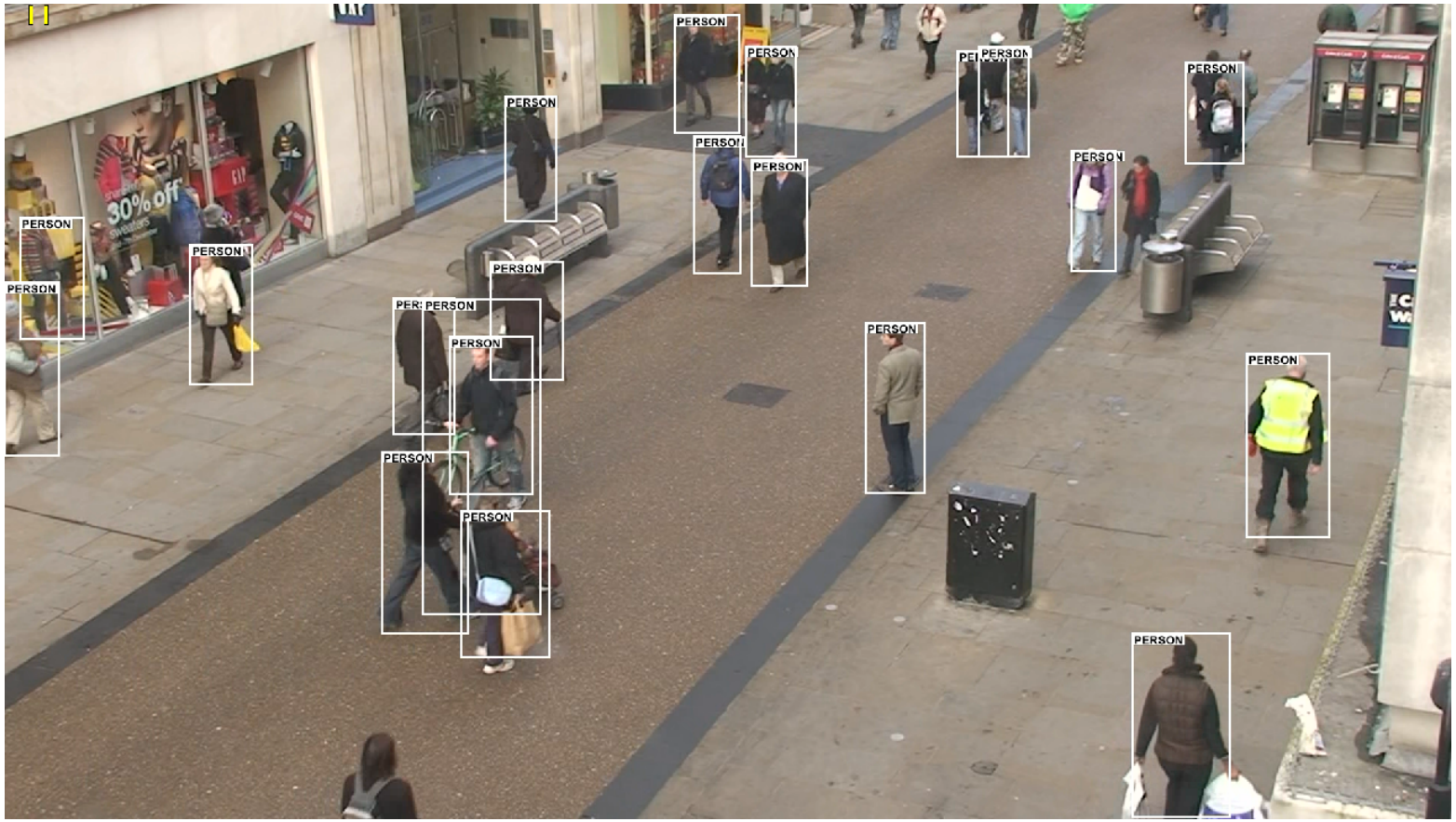
```c
QDEEP_STOP_OBJECT_DETECT(m_pDetector);

QDEEP_DESTROY_OBJECT_DETECT(m_pDetector);
```

1.5.3 SET BUFFER APIs

```c
ULONG m_pObjectSize = 1000;

QDEEP_OBJECT_DETECT_BOUNDING_BOX* m_pObjectList =
                                        (QDEEP_OBJECT_DETECT_BOUNDING_BOX*)malloc( m_pObjectSize * sizeof(QDEEP_OBJECT_DETECT_BOUNDING_BOX) );

QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER( m_pDetector,
                                                    QDEEP_COLORSPACE_TYEP_YV12,
                                                    nVideoWidth,
                                                    nVideoHeight,
                                                    pFrameBuffer,
                                                    nFrameBufferLen,
                                                    m_pObjectList,
                                                    &m_pObjectSize );
```

2 Object Detection and Object Tracking Function API

Introduction



This chapter provides the user an interface to use QDEEP SDK to detect and tracking.

2.1 Object Detect Creating Function

2.1.1 QDEEP_CREATE_OBJECT_DETECT

2.1.2 QDEEP_CREATE_OBJECT_DETECT_EX

Introduction

The user can use this function to create a Detector .

Intel CPU / GPU / VPU MOVIDIUS is only For **Object detection**

Parameters

type	parameter	I/O	descriptions
ULONG	nGpuType	IN	Specify the type of GPU QDEEP_GPU_TYPE_DEFAULT QDEEP_GPU_TYPE_NVIDIA QDEEP_GPU_TYPE_INTEL_CPU QDEEP_GPU_TYPE_INTEL_GPU QDEEP_GPU_TYPE_INTEL_VPU_MOVIDIUS
UINT	iGpuNum	IN	Specify Video Graphics Array index , start from 0

type	parameter	I/O	descriptions
ULONG	nConfigModel	IN	Specify Model config QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_GENERAL_DEFECT_DETECTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_PCB_DEFECT_DETECTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_BOAT QDEEP_OBJECT_DETECT_CONFIG_MODEL_DEPTH_MAP_3D_EX QDEEP_OBJECT_DETECT_CONFIG_MODEL_AOI_GAUGE_READER_DETECTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_DRIVING_DISTRACTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_EDUCATION QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_HEAD_BODY QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_5_KEYPOINTS QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_3D_EX QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_FACE_BEAUTY_EX QDEEP_OBJECT_DETECT_CONFIG_MODEL_FACE_LANDMARK_68_KEYPOINTS_MODAL_ANALYTICS_EX QDEEP_OBJECT_DETECT_CONFIG_MODEL_FLAME QDEEP_OBJECT_DETECT_CONFIG_MODEL_HAND_LANDMARK_21_KEYPOINTS QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_BLURRING QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_BACKGROUND_REMOVAL QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_AUTO_FRAMING QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_FACE_LAYOUT QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_EPTZ_SPEAKER_TRACKING QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_HANDWRITE_EXTRACTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SAFETY_INSPECTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SKELETON_17_KEYPOINTS QDEEP_OBJECT_DETECT_CONFIG_MODEL_HUMAN_SKELETON_136_KEYPOINTS QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_PARKING QDEEP_OBJECT_DETECT_CONFIG_MODEL_LICENSE_PLATE_RECOGNITION_LAW_ENFORCEMENT QDEEP_OBJECT_DETECT_CONFIG_MODEL_MISSING_OBJECT QDEEP_OBJECT_DETECT_CONFIG_MODEL_OPTICAL_CHARACTER_RECOGNITION QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_RETOUCHING QDEEP_OBJECT_DETECT_CONFIG_MODEL_PHOTO_SUPER_RESOLUTION QDEEP_OBJECT_DETECT_CONFIG_MODEL_RETAIL_PRODUCT_RECOGNITION QDEEP_OBJECT_DETECT_CONFIG_MODEL_SEGMENTATION QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_04 QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC_TAIWAN_08 QDEEP_OBJECT_DETECT_CONFIG_MODEL_UNATTENDED_OBJECT QDEEP_OBJECT_DETECT_CONFIG_MODEL_XRAY_INSPECTION_SYSTEM QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_LITE

type	parameter	I/O	descriptions
			QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MEDICAL_GRADE QDEEP_OBJECT_DETECT_CONFIG_MODEL_CUSTOMIZED_MULTI_LABELS QDEEP_OBJECT_DETECT_CONFIG_MODEL_NVIDIA_CLARA_AGX
CHAR *	pszConfigFileName	IN	**default NULL** Specify Model config
PVOID *	ppDetector	OUT	Handle of the detector object
float *	pModeVersion	OUT	Return the version of the model **Only in QDEEP_CREATE_OBJECT_DETECT_EX()**
DWORD	dwFlags	IN	**default QDEEP_OBJECT_DETECT_FLAG_FULL** Flexible switch flags
PVOID	pUserData	IN	**default NULL** Pointer to custom user data
CHAR *	pszEncryptionKey	IN	The decrypted key of the model **Only in QDEEP_CREATE_OBJECT_DETECT_EX()**

Return values

Name	Description
QDEEP_RS_SUCCESSFUL	Success
QDEEP_RS_ERROR_GENERAL	Fail due to some errors
QDEEP_RS_ERROR_OUT_OF_MEMORY	Fail due to system memory is not enough to allocate the capture object
QDEEP_RS_ERROR_OUT_OF_RESOURCE	Fail due to out of resource
QDEEP_RS_ERROR_INVALID_DEVICE	Cannot open video or audio capture device
QDEEP_RS_ERROR_INVALID_PARAMETER	Fail due to invalid input parameter
QDEEP_RS_ERROR_NON_SUPPORT	Fail due to encoder type doesn’t support

Examples

```c
PVOID m_pDetector = NULL;

QDEEP_CREATE_OBJECT_DETECT(QDEEP_GPU_TYPE_NVIDIA, 0, QDEEP_OBJECT_DETECT_CONFIG_MODEL_TRAFFIC, "QDEEP.OD.TRAFFIC.CFG", &m_pDetector);
```

2.1.3 CFG File Introduction

Parameters of CFG File

parameter	*callback descriptions*
SCOPE.SIZE	Specify object size can be detected
CLASS.SIZE	Specify the number of detect class
EXTRA.CFG.FILEPATH	Specify special configuration file

2.2 QDEEP_START_OBJECT_DETECT

Introduction

The user can use this function to start object detect .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_START_OBJECT_DETECT(m_pDetector);
```

2.3 QDEEP_STOP_OBJECT_DETECT

Introduction

The user can use this function to stop object detect .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_STOP_OBJECT_DETECT(m_pDetector);
```

2.4 QDEEP_DESTROY_OBJECT_DETECT

Introduction

This function can destroy the object detect and release its system resource.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```
QDEEP_DESTROY_OBJECT_DETECT(m_pDetector);
```

2.5 Object Detect Buffer Function

2.5.1 QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER

2.5.2 QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX

Introduction

The user can call this function to push an uncompressed video buffer into object detect engine.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
ULONG	nColorSpaceType	IN	Specify the input source buffer's color space type: QDEEP_COLORSPACE_TYEP_RGB24 QDEEP_COLORSPACE_TYEP_BGR24 QDEEP_COLORSPACE_TYEP_ARGB32 QDEEP_COLORSPACE_TYEP_ABGR32 QDEEP_COLORSPACE_TYEP_YUY2 QDEEP_COLORSPACE_TYEP_YV12
ULONG	nWidth	IN	Specify the input source buffer's width
ULONG	nHeight	IN	Specify the input source buffer's height
BYTE *	pFrameBuffer	IN	Specify the input source buffer
ULONG	nFrameBufferLen	IN	Specify the input source buffer's size
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	OUT	Pointer to the object buffer
ULONG	pObjectSize	IN/OUT	Pointer to the object number
ULONG	nCropX	IN	Specify the input source buffer's width **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX()**

type	parameter	I/O	descriptions
ULONG	nCropY	IN	Specify the input source buffer's height **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX()**
ULONG	nCropW	IN	Specify the width of crop **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX()**
ULONG	nCropH	IN	Specify the height of crop **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX()**
DWORD	dwPostFlags	IN	**default QDEEP_OBJECT_DETECT_FLAG_FULL** Support post flag to access all sub-functions

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pObjectSize = 1000;

QDEEP_OBJECT_DETECT_BOUNDING_BOX* pObjectList = (QDEEP_OBJECT_DETECT_BOUNDING_BOX*)
                                            malloc( m_DetBOXLen * sizeof(QDEEP_OBJECT_DETECT_BOUNDING_BOX) );

QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER( pDetector,
                                                    QDEEP_COLORSPACE_TYEP_YV12,
                                                    nVideoWidth,
                                                    nVideoHeight,
                                                    pFrameBuffer,
                                                    nFrameBufferLen,
                                                    pObjectList,
                                                    &pObjectSize );

QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_EX( pDetector ,
                                                       QDEEP_COLORSPACE_TYEP_YV12,
                                                       nVideoWidth,
                                                       nVideoHeight,
                                                       pFrameBuffer,
                                                       nFrameBufferLen,
                                                       pObjectList,
                                                       &pObjectSize,
                                                       0,
                                                       0,
                                                       960,
                                                       540);
```

2.5.3 QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D

2.5.4 QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX

Introduction

The user can call this function to push an uncompressed video buffer into object detect engine.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.

type	parameter	I/O	descriptions
ULONG	nColorSpaceType	IN	Specify the input source buffer's color space type: QDEEP_COLORSPACE_TYEP_RGB24 QDEEP_COLORSPACE_TYEP_BGR24 QDEEP_COLORSPACE_TYEP_ARGB32 QDEEP_COLORSPACE_TYEP_ABGR32 QDEEP_COLORSPACE_TYEP_YUY2 QDEEP_COLORSPACE_TYEP_YV12
ULONG	nWidth	IN	Specify the input source buffer's width
ULONG	nHeight	IN	Specify the input source buffer's height
BYTE *	pFrameBufferL	IN	Specify the left input source buffer
ULONG	nFrameBufferLenL	IN	Specify the left input source buffer's size
BYTE *	pFrameBufferR	IN	Specify the right input source buffer
ULONG	nFrameBufferLenR	IN	Specify the right input source buffer's size
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	OUT	Pointer to the object buffer
ULONG	pObjectSize	IN/OUT	Pointer to the object number
ULONG	nCropX	IN	Specify the input source buffer's width **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX()**
ULONG	nCropY	IN	Specify the input source buffer's height **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX()**
ULONG	nCropW	IN	Specify the width of crop **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX()**
ULONG	nCropH	IN	Specify the height of crop **Only in QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX()**
DWORD	dwPostFlags	IN	**default QDEEP_OBJECT_DETECT_FLAG_FULL** Support post flag to access all sub-functions

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pObjectSize = 1000;

QDEEP_OBJECT_DETECT_BOUNDING_BOX* pObjectList = (QDEEP_OBJECT_DETECT_BOUNDING_BOX*)
                                            malloc( m_DetBOXLen * sizeof(QDEEP_OBJECT_DETECT_BOUNDING_BOX) );

QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D( pDetector,
                                                       QDEEP_COLORSPACE_TYEP_YV12,
                                                       nVideoWidth,
                                                       nVideoHeight,
                                                       pFrameBufferL,
                                                       nFrameBufferLenL,
                                                       pFrameBufferR,
                                                       nFrameBufferLenR,
                                                       pObjectList,
                                                       &pObjectSize );

QDEEP_SET_VIDEO_OBJECT_DETECT_UNCOMPRESSION_BUFFER_3D_EX( pDetector ,
                                                          QDEEP_COLORSPACE_TYEP_YV12,
                                                          nVideoWidth,
                                                          nVideoHeight,
                                                          pFrameBufferL,
                                                          nFrameBufferLenL,
                                                          pFrameBufferR,
                                                          nFrameBufferLenR,
                                                          pObjectList,
                                                          &pObjectSize,
                                                          0,
                                                          0,
                                                          960,
                                                          540);
```

2.6 Object Detect Tracking Property

2.6.1 QDEEP_SET_OBJECT_DETECT_PROPERTY

Introduction

This function can set the Threshold of object detect.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
float	fObjectDetectionThreshold	IN	**default -1.0** Specify the Threshold of object detect

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
QDEEP_SET_OBJECT_DETECT_PROPERTY( m_pDetector , -1.0);
```

2.6.2 QDEEP_GET_OBJECT_DETECT_PROPERTY

Introduction

This function can get the Threshold of object detect.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
float *	pObjectDetectionThreshold	OUT	**default NULL** Pointer to the Threshold of object detect

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
float m_pObjectDetectionThreshold = 0;

QDEEP_GET_OBJECT_DETECT_PROPERTY ( m_pDetector, &m_pObjectDetectionThreshold );
```

2.6.3 QDEEP_SET_OBJECT_DETECT_TRACKING_PROPERTY

Introduction

This function can set tracking parameter which is life cycle and scope.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
float	fLifeCycle	IN	**default 5** Specify object tracking life cycle
float	fHitThreshold	IN	**default 2** Specify object tracking threshold
float	fScoreThreshold	IN	**default 0.5** Specify score threshold
BOOL	pFeatureVectorUpdate	IN	**default FALSE** Specify whether updating the feature vectors or not

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
QDEEP_SET_OBJECT_DETECT_TRACKING_PROPERTY( m_pDetector, 5, 2, 0.5, FALSE);
```

2.6.4 QDEEP_GET_OBJECT_DETECT_TRACKING_PROPERTY

Introduction

This function can get tracking parameter which is life cycle and scope.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
float *	pLifeCycle	OUT	**default NULL** Return object tracking life cycle
float *	pHitThreshold	OUT	**default NULL** Return object tracking threshold
float *	pScoreThreshold	OUT	**default NULL** Return object score threshold
BOOL *	pFeatureVectorUpdate	OUT	**default NULL** Return whether the feature vector is updating or not

Return value

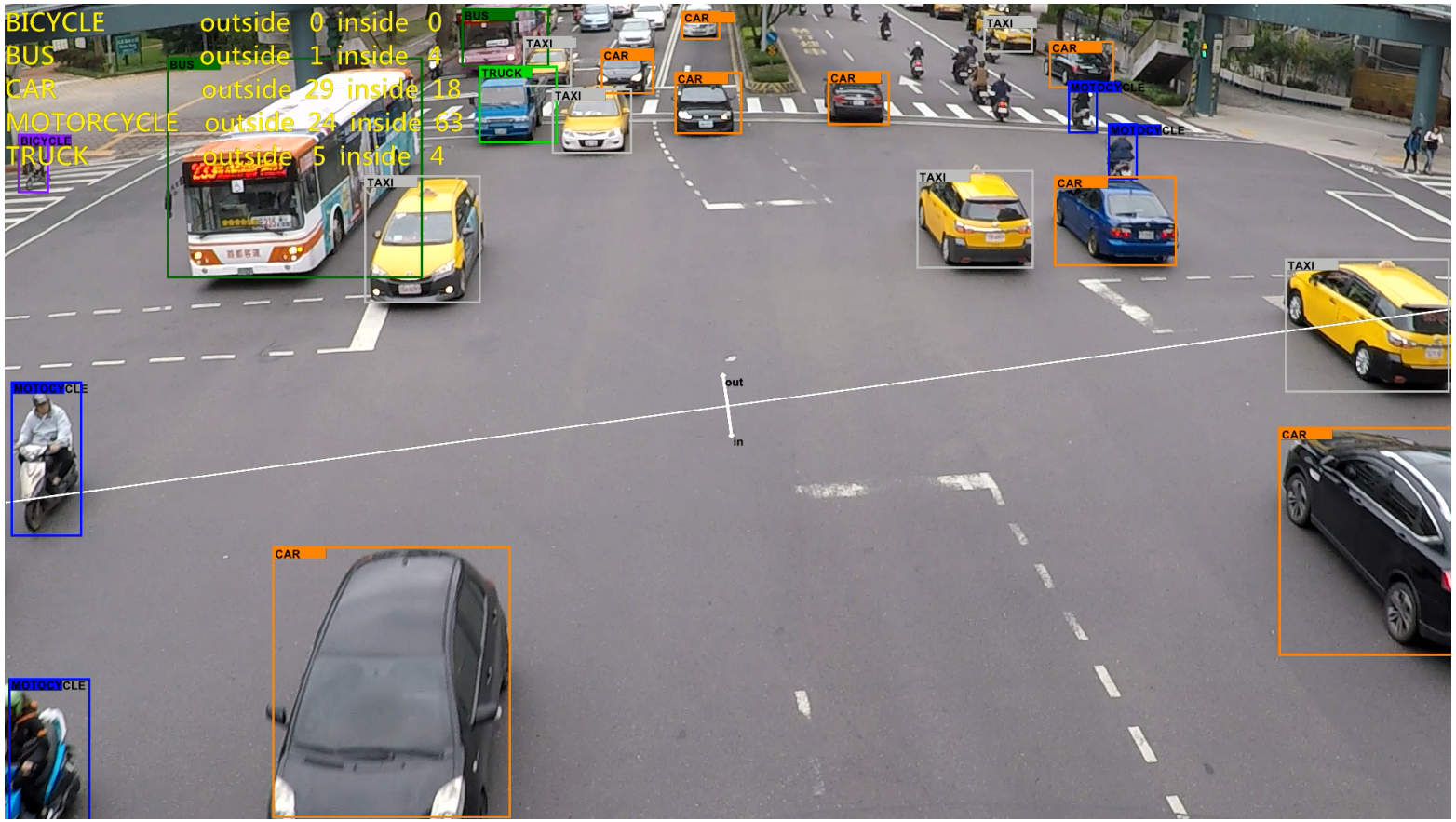
Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
float m_pLifeCycle = 0;

float m_pHitThreshold = 0;

float m_pScoreThreshold = 0;

BOOL pFeatureVectorUpdate = FALSE;

QDEEP_GET_OBJECT_DETECT_TRACKING_PROPERTY( m_pDetector, &m_pLifeCycle, &m_pHitThreshold, &m_pScoreThreshold, &pFeatureVectorUpdate );
```

3 Object Counting Function API

Introduction



This chapter provides the user an interface to use QDEEP SDK to line counting. in/out area counting and inside area counting.

3.1 QDEEP_ADD_OBJECT_DETECT_COUNTING_LINE

Introduction

The user can use this function to add a line which counting .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iLineNum	IN	Specify Line ID.
ULONG	nPoint1_X	IN	Set Starting Point X Coordinate.
ULONG	nPoint1_Y	IN	Set Starting Point Y Coordinate.
ULONG	nPoint2_X	IN	Set End X Coordinate.
ULONG	nPoint2_Y	IN	Set End Y Coordinate.
ULONG	nIntersectionMode	IN	Specify which point of the object crossing the line counts as the object crossed the line. QDEEP_COUNTING_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_COUNTING_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_COUNTING_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_COUNTING_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_ADD_OBJECT_DETECT_COUNTING_LINE(m_pDetector, 0, 0, 0, 1920, 1080, QDEEP_COUNTING_PROP_INTERSECTION_MODE_BOTTOM_POINT );
```

3.2 QDEEP_DEL_OBJECT_DETECT_COUNTING_LINE

Introduction

The user can use this function to delete line which counting .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iLineNum	IN	Specify Line ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_DEL_OBJECT_DETECT_COUNTING_LINE(m_pDetector, 0 );
```

3.3 QDEEP_RESET_OBJECT_DETECT_COUNTING_LINE

Introduction

The user can use this function to reset counting value by line .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iLineNum	IN	Specify Line ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_RESET_OBJECT_DETECT_COUNTING_LINE(m_pDetector, 0 );
```

3.4 QDEEP_GET_OBJECT_DETECT_COUNTING_LINE_STATUS

Introduction

The user can use this function to reset counting value by line .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iLineNum	IN	Specify Line ID.
ULONG	nClassID	IN	Specify Class Counting.
ULONG	pClassCrossCount_IN	OUT	Return Class Count A Direction Of The Line .
ULONG	pClassCrossCount_OUT	OUT	Return Class Count B Direction Of The Line .
ULONG	pTotalCrossCount_IN	OUT	Return Total Class A Direction Of The Line .
ULONG	pTotalCrossCount_OUT	OUT	Return Total Class B Direction Of The Line .

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_GET_OBJECT_DETECT_COUNTING_LINE_STATUS(m_pDetector,
                                             0,
                                             QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_PERSON,
                                             pClassCrossCount_IN,
                                             pClassCrossCount_OUT,
                                             pTotalCrossCount_IN,
                                             pTotalCrossCount_OUT);
```

3.5 QDEEP_ADD_OBJECT_DETECT_COUNTING_AREA

Introduction

The user can use this function to add a area which counting .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iAreaNum	IN	Specify Line ID.
ULONG	nTotalPoints	IN	Set Area Total Points .
ULONG*	pPointsArray_X	IN	Set Total Point X Coordinate.
ULONG*	pPointsArray_Y	IN	Set Total Point Y Coordinate.
ULONG	nIntersectionMode	IN	Specify which point of the object crossing the area counts as the object crossed the area. QDEEP_COUNTING_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_COUNTING_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_COUNTING_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_COUNTING_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_COUNTING_AREA(m_pDetector, 0, 4, pPointsArray_X, pPointsArray_Y, QDEEP_COUNTING_PROP_INTERSECTION_MODE_BOTTOM_POINT );
```

3.6 QDEEP_DEL_OBJECT_DETECT_COUNTING_AREA

Introduction

The user can use this function to delete area which counting .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iAreaNum	IN	Specify Area ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_DEL_OBJECT_DETECT_COUNTING_AREA(m_pDetector, 0 );
```

3.7 QDEEP_RESET_OBJECT_DETECT_COUNTING_AREA

Introduction

The user can use this function to reset counting value by area .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iAreaNum	IN	Specify Area ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
QDEEP_RESET_OBJECT_DETECT_COUNTING_AREA(m_pDetector, 0 );
```

3.8 QDEEP_GET_OBJECT_DETECT_COUNTING_AREA_STATUS

Introduction

The user can use this function to reset counting value by area .

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iAreaNum	IN	Specify Area ID.
ULONG	nClassID	IN	Specify Class Counting.
ULONG	pClassCrossCount_IN	OUT	Return class count enter the area .
ULONG	pClassCrossCount_OUT	OUT	Return class count go out area.
ULONG	pTotalCrossCount_IN	OUT	Return total class enter the area .
ULONG	pTotalCrossCount_OUT	OUT	Return total class go out area .

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_GET_OBJECT_DETECT_COUNTING_AREA_STATUS(m_pDetector,
                                             0,
                                             QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_PERSON,
                                             pClassCrossCount_IN,
                                             pClassCrossCount_OUT,
                                             pTotalCrossCount_IN,
                                             pTotalCrossCount_OUT);
```

3.9 Counting Callback

3.9.1 QDEEP_REGISTER_OBJECT_DETECT_COUNTING_LINE_CALLBACK

3.9.2 QDEEP_REGISTER_OBJECT_DETECT_COUNTING_AREA_CALLBACK

Introduction

This callback function will be called when a object cross line or enter/go out the area. The user can get the object information and counting.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the detector object
PF_OBJECT_DETECT_COUNTING_LINE_CALLBACK PF_OBJECT_DETECT_COUNTING_AREA_CALLBACK	pCB	IN	Callback function
PVOID	pUserData	IN	User defined data

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

PF_OBJECT_DETECT_COUNTING_LINE_CALLBACK

PF_OBJECT_DETECT_COUNTING_AREA_CALLBACK

Parameters of Callback

type	*parameter*	*callback descriptions*
PVOID	pDetector	Handle of the detector object
UINT	iLineNum iAreaNum	Line ID+
ULONG	nCrossDirection	Indicates direction
ULONG	nClassID	Indicates Class
ULONG	nObjectID	Indicates Object
ULONG	nClassCrossCount_IN	Return class cross A direction counting by line or enter the area.
ULONG	nClassCrossCount_OUT	Return class cross B direction counting by line or go out area.
ULONG	nTotalCrossCount_IN	Return total cross A direction counting or enter the area.

type	*parameter*	*callback descriptions*
ULONG	nTotalCrossCount_OUT	Return total cross B direction counting by line or go out area.
PVOID	pUserData	User defined data

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
// COUNT CALLBACK
QRETURN OBJECT_DETECT_COUNTING_LINE_CALLBACK( PVOID pDetector, UINT iLineNum, ULONG nCrossDirection, ULONG nClassID, ULONG nObjectID, ULONG
nClassCrossCount_IN, ULONG nClassCrossCount_OUT, ULONG nTotalCrossCount_IN, ULONG nTotalCrossCount_OUT, PVOID pUserData );
{
        m_nLineCount[ iLineNum * m_pMainDialog->m_nTotalClass * 2 + nClassID * 2 + nCrossDirection] ++ ;
}

void test_callback()
{
        QDEEP_REGISTER_OBJECT_DETECT_COUNTING_LINE_CALLBACK (m_pDetector, OBJECT_DETECT_COUNTING_LINE_CALLBACK, pUserData );
}
```

4 Face Recognition Function API

This chapter provides the user an interface to face detection and recognition by using QDEEP SDK.

4.1 QDEEP_GET_OBJECT_RECOGNITION_COMPARISON

Introduction

The user can use this function to face detection and recognition.

Parameters

type	parameter	I/O	descriptions
float	pFeatureVectorA	IN	Specify the feature vector of object A.
float	pFeatureVectorB	IN	Specify the feature vector of object B.
float	pSimilarity	OUT	Point to the similarity between object A and object B.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
float m_pSimilarity = 0;

QDEEP_GET_OBJECT_RECOGNITION_COMPARISON(m_pFeatureVectorA, m_pFeatureVectorB, &m_pSimilarity);
```

4.2 QDEEP_SET_OBJECT_DETECT_POST_PROCESSING_UPDATE

Introduction

The user can use this function to perform asynchronous post processing update function.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
ULONG	nGpuType	IN	Specify the type of GPU QDEEP_GPU_TYPE_NVIDIA QDEEP_GPU_TYPE_INTEL_CPU QDEEP_GPU_TYPE_INTEL_GPU QDEEP_GPU_TYPE_INTEL_VPU_MOVIDIUS
UINT	iGpuNum	IN	Specify Video Graphics Array index , start from 0
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	IN	Specify the object buffer
ULONG	pObjectSize	IN	Specify the object number
DWORD	dwPostFlags	IN	**default QDEEP_OBJECT_DETECT_FLAG_FEATURE_VECTOR** Support post flags to access some sub-functions

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_SET_OBJECT_DETECT_POST_PROCESSING_UPDATE(m_pDetector, QDEEP_GPU_TYPE_NVIDIA, 0,
                                               pObjectList, pObjectSize,
                                               QDEEP_OBJECT_DETECT_FLAG_FEATURE_VECTOR);
```

4.3 Post Processing Update Callback

4.3.1 QDEEP_REGISTER_OBJECT_DETECT_POST_PROCESSING_UPDATE_CALLBACK

Introduction

This callback function will be called when a object updates its face information.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the detector object
PF_OBJECT_DETECT_POST_PROCESSING_UPDATE_CALLBACK	pCB	IN	Callback function
PVOID	pUserData	IN	User defined data

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

PF_OBJECT_DETECT_POST_PROCESSING_UPDATE_CALLBACK

Parameters of Callback

type	*parameter*	*callback descriptions*
PVOID	pDetector	Handle of the detector object
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	Return the object buffer
PVOID	pUserData	User defined data

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

5 Event Function API

This chapter provides the user an interface to detect object-event by using QDEEP SDK.

5.1 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_CROSSING_LINE

Introduction

The user can use this event function to detect object crossing-line event. When the number of crossing-line objects is more than the specified threshold, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

PVOID	pDetector	IN	Handle of the Detector object
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nPoint1_X	IN	Set Starting Point X coordinate.
ULONG	nPoint1_Y	IN	Set Starting Point Y coordinate.
ULONG	nPoint2_X	IN	Set End X coordinate.
ULONG	nPoint2_Y	IN	Set End Y coordinate.
ULONG	nCrossDirection	IN	Specify Cross-direction.(0:counterclockwise, 1:clockwise)
ULONG	nObjectSizeThreshold	IN	Specify Cross-line object number.
ULONG	nDurationThreshold	IN	Specify Cross-line object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object crossing the line counts as the object crossed the line. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_CROSSING_LINE(pDetector, 0, pSelectClassIDs, 3, 0, 0, 1920, 1080, 0, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT );
```

5.2 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_IN_AREA

Introduction

The user can use this event function to detect objects which are in the area. When the number of objects in area is more than the specified threshold, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object is in the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT );
```

5.3 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_ENTER_AREA

Introduction

The user can use this event function to detect objects which enter the area. When the number of entering-area objects is more than the specified threshold, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object entered the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_ENTER_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.4 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_LEAVE_AREA

Introduction

The user can use this event function to detect objects which leaft the area. When the number of leaving-area objects is more than the threshold, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object left the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_LEAVE_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.5 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_STOP_IN_AREA

Introduction

Users can utilize this event function to detect objects that have stopped in the area. When the number of stopped objects exceeds the threshold, the PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object stopped in the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_STOP_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.6 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_STOP_IN_AREA_WITHOUT_WARNING_SIGN

Introduction

Users can utilize this event function to detect objects that have stopped in the area when there isn't a stop sign. When the number of stopped objects exceeds the threshold, the PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG *	pWarningSignClassIDs	IN	Specify warning sign class ID buffer.
ULONG	nWarningSignClassSize	IN	Specify warning sign class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object stopped in the area without a warning sign. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG nWarningSignClassIDs[1] = { 0 };

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_STOP_IN_AREA_WITHOUT_WARNING_SIGN(pDetector, 0, pSelectClassIDs, 3, nWarningSignClassIDs, 1, 4,
pPointsArray_X, pPointsArray_Y, 1, 500, QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.7 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_COLLIDE_IN_AREA

Introduction

Users can utilize this event function to detect objects that have collided in the area. When the number of stopped objects exceeds the threshold, the PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nObjectSizeThreshold	IN	Specify object number.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as two objects collided in the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECTS_COLLIDE_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 1, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.8 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_APPEAR_IN_AREA

Introduction

The user can use this event function to detect objects which appear in the area simultaneously. When the objects appear in area, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object appeared in the area. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_APPEAR_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.9 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_ABSENT_IN_AREA

Introduction

The user can use this event function to detect objects which disappear in the area. When there is not object in area, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object is absent in the area QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_ABSENT_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.10 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_LOITERING_IN_AREA

Introduction

The user can use this event function to detect objects which loiter in the area. When there are objects loiter in the area more than a certain period of time, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nDurationThreshold	IN	Specify object wandering time threashold. (ms)
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object is wandering in the area QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_LOITERING_IN_AREA(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 500,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.11 QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_DIRECTION_REVERSED

Introduction

The user can use this event function to detect objects in the area but the forward direction is reversed. When there is an object in the area that doesn't follow the direction of the specified angle, PF_OBJECT_DETECT_EVENT_CALLBACK will be triggered.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
ULONG *	pSelectClassIDs	IN	Specify Class ID buffer.
ULONG	nSelectClassSize	IN	Specify Class size.
ULONG	nTotalPoints	IN	Set Area Total Points.
ULONG *	pPointsArray_X	IN	Set Total Point X coordinate.
ULONG *	pPointsArray_Y	IN	Set Total Point Y coordinate.
ULONG	nDurationThreshold	IN	Specify object duration time threashold.
float	fDirAngle	IN	Specified angle(angle: 0.0 ~ 360.0)
float	fToleranceAngle	IN	Specified tolerance angle(angle: 0.0 ~ 360.0)
ULONG	nIntersectionMode	IN	Specify which point of the object hit the area counts as the object reversed its direction. QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT QDEEP_EVENT_PROP_INTERSECTION_MODE_UPPER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_LOWER_BOUNDING_BOX QDEEP_EVENT_PROP_INTERSECTION_MODE_FULL_BOUNDING_BOX

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
ULONG pSelectClassIDs[3] = {QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_BIKE,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_MOTOR,
                            QDEEP_OBJECT_DETECT_CLASS_TRAFFIC_TAIWAN_08_TAXI};

ULONG pPointsArray_X[4] = {0, 1920, 1920, 0};

ULONG pPointsArray_Y[4] = {0, 0, 1080, 1080};

QDEEP_ADD_OBJECT_DETECT_EVENT_OBJECT_DIRECTION_REVERSED(pDetector, 0, pSelectClassIDs, 3, 4, pPointsArray_X, pPointsArray_Y, 500, 120, 10,
QDEEP_EVENT_PROP_INTERSECTION_MODE_BOTTOM_POINT);
```

5.12 QDEEP_DEL_OBJECT_DETECT_EVENT

Introduction

The user can use this function to delete the event.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_DEL_OBJECT_DETECT_EVENT(pDetector, 0);
```

5.13 QDEEP_RESET_OBJECT_DETECT_EVENT

Introduction

The user can use this function to reset the event.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```c
QDEEP_RESET_OBJECT_DETECT_EVENT(pDetector, 0);
```

5.14 QDEEP_GET_OBJECT_DETECT_EVENT_CURRENT_STATUS

Introduction

The user can use this function to get the information of the objects that match the specified event.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the Detector object.
UINT	iEventNum	IN	Specify Event ID.
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	OUT	Returns pointer to object buffer matching the event.
ULONG *	pObjectSize	IN/OUT	Returns the number of objects matching the event.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

Examples

```C
ULONG pObjectSize = 1000;

QDEEP_OBJECT_DETECT_BOUNDING_BOX* pObjectList = (QDEEP_OBJECT_DETECT_BOUNDING_BOX*)
                                          malloc( m_DetBOXLen * sizeof(QDEEP_OBJECT_DETECT_BOUNDING_BOX) );

QDEEP_GET_OBJECT_DETECT_EVENT_CURRENT_STATUS(pDetector, 0, pObjectList, &pObjectSize);
```

5.15 Object Event Callback

5.15.1 QDEEP_REGISTER_OBJECT_DETECT_EVENT_CALLBACK

Introduction

This callback function will be called when the event happens.

Parameters

type	parameter	I/O	descriptions
PVOID	pDetector	IN	Handle of the detector object.
PF_OBJECT_DETECT_EVENT_CALLBACK	pCB	IN	Callback function.
PVOID	pUserData	IN	User defined data.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

PF_OBJECT_DETECT_EVENT_CALLBACK

Parameters of Callback

type	*parameter*	*callback descriptions*
PVOID	pDetector	Handle of the detector object.
UINT	iEventNum	Return the event ID.
QDEEP_OBJECT_DETECT_BOUNDING_BOX *	pObjectList	Returns pointer to object buffer matching the event.
ULONG	nObjectSize	Returns the number of objects matching the event.
PVOID	pUserData	User defined data.

Return value

Returns **QDEEP_RS_SUCCESSFUL** if OK, otherwise an error occurred.

範例程式

```c
// COUNT CALLBACK
QRETURN QDEEP_OBJECT_DETECT_EVENT_CALLBACK( PVOID pDetector, UINT iEventNum, QDEEP_OBJECT_DETECT_BOUNDING_BOX *pObjectList, ULONG nObjectSize,
PVOID pUserData );
{
        printf("[iEventNum %d]", iEventNum);
        for(int i = 0; i < nObjectSize; ++i){
            printf("[%d], nClassID: %d", i, pObjectList[i].nClassID);
        }
}

void test_callback()
{
        QDEEP_REGISTER_OBJECT_DETECT_EVENT_CALLBACK (m_pDetector, QDEEP_OBJECT_DETECT_EVENT_CALLBACK, pUserData );
}
```