Camera SDK Reference Guide

C++

Release Date: June 18, 2025

Version: V1.0

SHANGHAI PERCIPIO TECHNOLOGY LIMITED

Table of Contents

1 Features Under the *Device* Component

1.1 TY_TRIGGER_PARAM_EX

Function:

This feature is used to configure the camera work mode.

Detailed explanation:

The Percipio cameras support multiple work modes. The most used work modes include:

TY_TRIGGER_MODE_OFF: Continuous Capture Mode

TY_TRIGGER_MODE_SLAVE: Software/Hardware Trigger Mode

TY_TRIGGER_MODE_M_SIG: After the master device receives the software trigger signal, it triggers itself and simultaneously outputs a hardware trigger signal through the Output pin to trigger the slave devices.

Example:

```
TY_TRIGGER_PARAM_EX trigger;
trigger.mode = TY_TRIGGER_MODE_OFF;//Set to the required work mode
ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_PARAM_EX, &trigger,
sizeof(trigger)));
```

TY_TRIGGER_MODE_M_PER: The camera captures images at a specific frame rate while outputting synchronization signals through the Output pin to trigger slave devices.

Example:

```
TY_TRIGGER_PARAM_EX param;
param.mode = TY_TRIGGER_MODE_M_PER;
param.fps = 5;
ASSERT_OK(TYSetStruct(cams[count].hDev, TY_COMPONENT_DEVICE, TY_STRUCT_TRIGGER_PARAM_EX,
(void*)&param, sizeof(param)));
```

Note:

- For *TY_TRIGGER_MODE_M_SIG* and *TY_TRIGGER_MODE_M_PER*, use firmware *V3.13.68* or later.
- Legacy systems should use *TY_TRIGGER_PARAM* instead of *TY_TRIGGER_PARAM_EX*.

1.2 TY_INT_FRAME_PER_TRIGGER

Function:

This feature is used to set the number of frames the camera captures after receiving a software/hardware trigger signal.

Note:

The camera is configured to capture a single frame per trigger signal by default. Each capture cycle follows an open-close life cycle pattern:

Example:

```
int32_t value = 2;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_FRAME_PER_TRIGGER, value));
```

1.3 TY_INT_PACKET_DELAY

Function:

This feature is used to set the delay time between packets during camera data transmission, for use in non-ideal network environments.

Example:

```
int32_t value = 10000;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_PACKET_DELAY, value));
```

1.4 TY_INT_PACKET_SIZE

Function:

This feature is used to set the size of camera data packets, for use in non-ideal network environments.

Example:

```
int32_t value = 100;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_PACKET_SIZE, value));
```

1.5 TY_BOOL_GVSP_RESEND

Function:

This feature is used to enable the network camera image retransmission function.

Example:

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_GVSP_RESEND, true));
```

1.6 TY_BOOL_TRIGGER_OUT_IO

Function:

This feature is used to invert the output level of trigger_out.

Example:

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TRIGGER_OUT_IO, false));
```

Note:

This feature is writable but not readable.

1.7 TY_BOOL_KEEP_ALIVE_ONOFF

Function:

This feature is used to set the SDK and camera communication keep-alive mechanism. The default value is true, indicating communication maintenance is on.

Example:

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_KEEP_ALIVE_ONOFF, false));
```

1.8 TY_INT_KEEP_ALIVE_TIMEOUT

Function:

This feature is used to set the SDK and camera communication status keep-alive maintaining time. The default value for USB cameras: 15000ms; for network camera: 3000ms. Unit: ms.

Example:

```
int32_t value = 30000;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_KEEP_ALIVE_TIMEOUT, value));
```

1.9 TY_INT_TRIGGER_DELAY_US

Function:

This feature is used to set the software/hardware trigger delay time. The camera outputs an image after a delay period following the reception of a hardware trigger signal. Unit: us.

Example:

```
int32_t value = 1300000;
```

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_TRIGGER_DELAY_US, value));
```

1.10 TY_INT_TRIGGER_DURATION_US

Function:

This feature is used to set the output signal level hold time. Unit: us.

Example:

```
int32_t value = 100000;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_TRIGGER_DURATION_US, value));
```

1.11 TY_ENUM_STREAM_ASYNC

Function:

This feature is used to set the data stream asynchronous feature.

TY_STREAM_ASYNC_OFF: data stream synchronous.

TY_STREAM_ASYNC_DEPTH: depth stream asynchronous.

TY_STREAM_ASYNC_RGB: RGB stream asynchronous.

TY_STREAM_ASYNC_DEPTH_RGB: depth and RGB streams asynchronous.

TY_STREAM_ASYNC_ALL: all data streams asynchronous.

Example:

```
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_STREAM_ASYNC,

TY_STREAM_ASYNC_RGB));
```

1.12 TY_INT_CAPTURE_TIME_US

Function:

This feature is used to read depth calculation time, only applicable in trigger mode.

Unit: us.

Note:

The depth calculation time in continuous capture mode is 0.

Example:

```
int32_t default_value=0;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_CAPTURE_TIME_US, &default_value));
```

1.13 TY_ENUM_TIME_SYNC_TYPE

Function:

This feature is used to set camera time synchronization function.

TY_TIME_SYNC_TYPE_NONE: No time synchronization.

TY_TIME_SYNC_TYPE_HOST: Camera synchronizes with host computer.

TY_TIME_SYNC_TYPE_NTP: Camera synchronizes with NTP server.

TY_TIME_SYNC_TYPE_PTP: Camera synchronizes with PTP server.

TY_TIME_SYNC_TYPE_CAN: Camera synchronizes with CAN network, only FM862-GDW supports, no validation performed.

TY_TIME_SYNC_TYPE_PTP_MASTER: Set camera as PTP server.

Example:

```
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE,
TY_TIME_SYNC_TYPE_HOST));
while (1) {
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
if (sync_ready) {
break;
}
MSLEEP(10);
}
```

For **NTP time synchronization**, additional validation of the synchronization status after specifying the server IP is required.

Example:

```
const char* ntp_ip = " 119.29.26.206 ";
int32_t ip_i[4];
uint8_t ip_b[4];
int32_t ip32;
sscanf(ntp_ip, "%d.%d.%d.%d", &ip_i[0], &ip_i[1], &ip_i[2], &ip_i[3]);
ip_b[0] = ip_i[0]; ip_b[1] = ip_i[1]; ip_b[2] = ip_i[2]; ip_b[3] = ip_i[3];
ip32 = TYIPv4ToInt(ip_b);
LOGI("Set persistent IP 0x%x(%d.%d.%d.%d)", ip32, ip_b[0], ip_b[1], ip_b[2], ip_b[3]);
```

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_NTP_SERVER_IP, ip32));
ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEVICE,TY_INT_NTP_SERVER_IP, &ip32));
TYIntToIPv4(ip32, ip_b);
LOGD("%d %d %d %d", ip_b[0], ip_b[1], ip_b[2], ip_b[3]);
LOGD("Set type of time sync mechanism");
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TIME_SYNC_TYPE,
TY_TIME_SYNC_TYPE_NTP));
LOGD("Wait for time sync ready");
while (1) {
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
if (sync_ready) {
break;
}
MSLEEP(10);
}
```

1.14 TY_BOOL_TIME_SYNC_READY

Function:

This feature is used to determine whether time synchronization is successful.

Example:

```
bool sync_ready;
ASSERT_OK(TYGetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_TIME_SYNC_READY, &sync_ready));
```

1.15 TY_BOOL_CMOS_SYNC

Function:

This feature is used to set the left and right IR asynchronous exposure switch. true: synchronous exposure; false: asynchronous exposure.

Example:

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEVICE, TY_BOOL_CMOS_SYNC, false));
```

1.16 TY_INT_ACCEPTABLE_PERCENT

Function:

This feature is used to set network packet loss tolerance. Images with a received data packet percentage below this threshold will be discarded by the host computer. Unit: %.

Example:

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEVICE, TY_INT_ACCEPTABLE_PERCENT, 90));
```

1.17 TY_STRUCT_CAM_STATISTICS

Function:

This feature is used to obtain the image transmission status of the network camera.

Example:

```
TY_CAMERA_STATISTICS st;
ASSERT_OK( TYGetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_CAM_STATISTICS, &st, sizeof(st)) );
LOGI("Statistics:");
LOGI("  packetReceived: %" PRIu64 " ", st.packetReceived);
LOGI("  packetLost    : %" PRIu64 " ", st.packetLost);
LOGI("  imageOutputed : %" PRIu64 " ", st.imageOutputed);
LOGI("  imageDropped  : %" PRIu64 " ", st.imageDropped);
```

1.18 TY_ENUM_TEMPERATURE_ID

Function:

This feature is used together with *TY_STRUCT_TEMPERATURE* to read temperature values at specified locations. Currently supported reading positions include: left IR position, right IR position, color position, CPU position, and mainboard position.

Example:

```
uint32_t n = 0;
ASSERT_OK(TYGetEnumEntryCount(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID, &n));
LOGD("===        %14s: entry count %d", "", n);
std::vector<TY_ENUM_ENTRY> feature_info(n);
if (n == 0) {
    LOGD("None temperature sensor exist!\n");
}
else {
```

```
    ASSERT_OK(TYGetEnumEntryInfo(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID,
&feature_info[0], n, &n));
    for (int i = 0; i < n; i++) {
      int ret = TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_TEMPERATURE_ID,
feature_info[i].value);
      if (ret < 0) { LOGD("Set temperature id[%d](%s) failed %d(%s)\n", feature_info[i].value,
feature_info[i].description, ret, TYErrorString(ret));
        break;       }
      TY_TEMP_DATA temp;
      memset(&temp, 0, sizeof(temp));
      ret = TYGetStruct(hDevice, TY_COMPONENT_DEVICE, TY_STRUCT_TEMPERATURE, &temp, sizeof(temp));
      if (ret < 0) { LOGD("Get temperature [%d](%s) failed %d(%s)\n", feature_info[i].value,
feature_info[i].description, ret, TYErrorString(ret));
        break;
      }
      LOGD("Get temperature [%d](%s) temp %s\n", feature_info[i].value, feature_info[i].description,
temp.temp);
    } }
```

1.19 IP Setting Related Features

IP setting involves 3 features:

Set camera IP address: *TY_INT_PERSISTENT_IP*,

Set subnet mask: *TY_INT_PERSISTENT_SUBMASK*,

Set gateway: TY_INT_PERSISTENT_GATEWAY.

2 Features Under the *Laser* Component

2.1 TY_BOOL_LASER_AUTO_CTRL

Function:

This feature is used to control the laser automatic control switch.

Example:

```
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_LASER_AUTO_CTRL, false));
```

2.2 TY_INT_LASER_POWER

Function:

This feature is used to set the laser/structured light projector light source intensity.

Example:

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_LASER_POWER, value));
```

2.3 TY_BOOL_IR_FLASHLIGHT

Function:

This feature is used to enable the IR floodlight source.

Example:

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_IR_FLASHLIGHT, value));
```

2.4 TY_BOOL_RGB_FLASHLIGHT

Function:

This feature is used to enable the RGB floodlight source.

Example:

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_LASER, TY_BOOL_RGB_FLASHLIGHT, value));
```

2.5 TY_INT_IR_FLASHLIGHT_INTENSITY

Function:

This feature is used to set the IR floodlight brightness intensity.

Example:

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_IR_FLASHLIGHT_INTENSITY, value));
```

2.6 TY_INT_RGB_FLASHLIGHT_INTENSITY

Function:

This feature is used to set the RGB floodlight brightness intensity.

Example:

```
int32_t value = 0;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_LASER, TY_INT_RGB_FLASHLIGHT_INTENSITY, value));
```

3 Features Under the *Depth* Component

3.1 TY_INT_SGBM_IMAGE_NUM

Function:

This feature is used to set the number of IR images required for SGBM depth calculation.

Example:

```
int32_t value = 10;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_IMAGE_NUM, value));
```

3.2 TY_INT_SGBM_DISPARITY_NUM

Function:

This feature is used to set the disparity search range for SGBM depth calculation.

Example:

```
int32_t value = 10;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_NUM, value));
```

3.3 TY_FLOAT_SCALE_UNIT

Function:

This feature is used to set the depth value unit.

Example:

```
float value = 0.25;
ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_DEPTH_CAM, TY_FLOAT_SCALE_UNIT,value));
```

3.4 TY_INT_SGBM_DISPARITY_OFFSET

Function:

This feature is used to set the starting disparity value for SGBM depth calculation search.

Example:

```
int32_t value = 10;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_DISPARITY_OFFSET, value));
```

3.5 TY_INT_SGBM_MATCH_WIN_HEIGHT

Function:

This feature is used to set the height of the matching window for SGBM depth calculation.

Example:

```
int32_t value = 10;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_HEIGHT, value));
```

Note:

TY_INT_SGBM_MATCH_WIN_HEIGHT and *TY_INT_SGBM_IMAGE_NUM* have constraint relationships. Please refer to the camera config file for specific constraints.

3.6 TY_INT_SGBM_MATCH_WIN_WIDTH

Function:

This feature is used to set the width of the matching window for SGBM depth calculation.

Example:

```
int32_t value = 10;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MATCH_WIN_WIDTH, value));
```

3.7 TY_INT_SGBM_SEMI_PARAM_P1

Function:

This feature is used to set the penalty parameter for adjacent pixels (+/-1) constraint.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1, value));
```

3.8 TY_INT_SGBM_SEMI_PARAM_P1_SCALE

Function:

This feature is used to set the P1_scale penalty parameter for adjacent pixels (+/-1) constraint.

Example:

```
int32_t value = 1;
```

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1_SCALE, value));
```

3.9 TY_INT_SGBM_SEMI_PARAM_P2

Function:

This feature is used to set the P2 penalty parameter for surrounding pixels constraint.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P2, value));
```

3.10 TY_INT_SGBM_UNIQUE_FACTOR

Function:

This feature is used to set the uniqueness check parameter 2, which represents the difference between the optimal and sub-optimal matching points.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR, value));
```

3.11 TY_INT_SGBM_UNIQUE_ABSDIFF

Function:

This feature is used to set uniqueness check parameter 1, which is the percentage between optimal and sub-optimal matching points.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_UNIQUE_FACTOR, value));
```

3.12 TY_BOOL_SGBM_HFILTER_HALF_WIN

Function:

This feature is used to enable/disable search filtering. Used for further optimizing depth maps, removing noise and discontinuities, making it more friendly to object edge point clouds.

Example:

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_HFILTER_HALF_WIN, value));
```

3.13 TY_BOOL_SGBM_MEDFILTER

Function:

This feature is used to enable/disable median filtering, which eliminates isolated noise points while preserving edge information of the image as much as possible.

Example:

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_MEDFILTER, value));
```

3.14 TY_INT_SGBM_MEDFILTER_THRESH

Function:

This feature is used to set median filter threshold. The larger the setting value, the more noise points are filtered, but may also cause loss of detail information in the depth map.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_MEDFILTER_THRESH, value));
```

3.15 TY_BOOL_SGBM_LRC

Function:

This feature is used to enable/disable left-right consistency check.

Example:

```
bool value = false;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_SGBM_LRC, value));
```

3.16 TY_INT_SGBM_LRC_DIFF

Function:

This feature is used to set median filter threshold. The larger the setting value, the more noise points are filtered, but may also cause loss of detail information in the depth map.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_LRC_DIFF, value));
```

3.17 TY_ENUM_DEPTH_QUALITY

Function:

This feature is used to set ToF camera depth image quality.

Example:

```
uint32_t value = 0;

ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEPTH_CAM, feature_id, value));
```

3.18 TY_INT_TOF_MODULATION_THRESHOLD

Function:

This feature is used to set the threshold for ToF depth camera receiving laser modulated light intensity. Pixels below this threshold do not participate in depth calculation, i.e., the depth value of such pixels is assigned to 0.

Example:

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_MODULATION_THRESHOLD, value));
```

3.19 TY_INT_TOF_JITTER_THRESHOLD

Function:

This feature is used to set ToF depth camera jitter filter threshold. The larger the threshold setting value, the less depth data jitter at depth map edges is filtered.

Example:

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_JITTER_THRESHOLD, value));
```

3.20 TY_INT_FILTER_THRESHOLD

Function:

This feature is used to set ToF depth camera flying point filter threshold. Default value is 0, meaning no filtering. The smaller the filter threshold setting, the more flying points are filtered.

Example:

```
int32_t value = 1;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_FILTER_THRESHOLD, value));
```

3.21 TY_INT_TOF_CHANNEL

Function:

This feature is used to set ToF depth camera modulation channel. Different modulation channels have different modulation frequencies and do not interfere with each other. If multiple ToF depth cameras of the same series need to run in the same scene, ensure they use different modulation channels.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_CHANNEL, value));
```

3.22 TY_INT_TOF_HDR_RATIO

Function:

This feature is used to set high dynamic range ratio threshold, which needs to be used in *TY_ENUM_DEPTH_QUALITY=HIGH* mode.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_HDR_RATIO, value));
```

3.23 TY_INT_TOF_ANTI_SUNLIGHT_INDEX

Function:

This feature is used to set ToF anti-sunlight index.

Example:

```
int32_t value = 1;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_TOF_ANTI_SUNLIGHT_INDEX, value));
```

3.24 TY_INT_MAX_SPECKLE_DIFF

Function:

This feature is used to set speckle filter clustering threshold. Unit: mm. If the depth difference between adjacent pixels is less than this threshold, the adjacent pixels are considered to belong to the same clustering speckle.

Example:

```
int32_t value = 200;
```

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_DIFF, value));
```

3.25 TY_INT_MAX_SPECKLE_SIZE

Function:

This feature is used to set speckle filter area threshold. Unit: pixels. Clustering speckles with area smaller than this threshold will be filtered out.

Example:

```
int32_t value = 200;
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_MAX_SPECKLE_SIZE, value));
```

3.26 TY_BOOL_TOF_ANTI_INTERFERENCE

Function:

This feature is used to enable/disable anti-multi-device interference, applicable to scenarios where the number of dToF cameras is greater than the number of ToF channels, which can effectively avoid multi-device interference.

Example:

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_DEPTH_CAM, TY_BOOL_TOF_ANTI_INTERFERENCE, value));
```

3.27 TY_ENUM_CONFIG_MODE

Function:

This feature is used to set preset parameters for V-series cameras, with different camera accuracy under different modes.

Example:

```
uint32_t value = TY_CONFIG_MODE_PRESET1;
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEVICE, TY_ENUM_CONFIG_MODE, value));
```

3.28 TY_INT_SGBM_TEXTURE_THRESH

Function:

This feature is used to solve abnormal depth image situations when there is no background or distant background behind the measured object.

Example:

```
int32_t value = 500;
```

```
TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_THRESH, value);
```

3.29 TY_INT_SGBM_TEXTURE_OFFSET

Function:

This feature is used to solve poor depth imaging problems when the measured object has background or distant background. This property is read-only.

Example:

```
int32_t value;

TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_TEXTURE_OFFSET, &value);
```

3.30 TY_INT_DEPTH_MIN_MM

Function:

This feature is used to set the minimum cutoff distance for depth values. Unit: mm.

Example for Setting:

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MIN_MM, 1400));
```

Reading for Getting:

```
int min = 0;;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MIN_MM, &min));
```

3.31 TY_INT_DEPTH_MAX_MM

Function:

This feature is used to set the maximum cutoff distance for depth values. Unit: mm.

Example for Setting:

```
ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_MAX_MM, 1500));
```

Reading for Getting:

```
int max = 0;;

ASSERT_OK(TYGetInt(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_DEPTH_ MAX _MM, &max));
```

4 Features Under the *RGB* Component

4.1 TY_INT_ANALOG_GAIN

Function:

This feature is used to set RGB analog gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_ANALOG_GAIN, value));
```

4.2 TY_INT_R_GAIN

Function:

This feature is used to set RGB red channel gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_R_GAIN, value));
```

4.3 TY_INT_G_GAIN

Function:

This feature is used to set RGB green channel gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_G_GAIN, value));
```

4.4 TY_INT_B_GAIN

Function:

This feature is used to set RGB blue channel gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_B_GAIN, value));
```

4.5 TY_INT_EXPOSURE_TIME

Function:

This feature is used to set RGB exposure time.

Example:

```
int32_t value = 100;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_EXPOSURE_TIME, value));
```

4.6 TY_FLOAT_EXPOSURE_TIME_US

Function:

This feature is used to set the actual exposure time of the color sensor. Unit: us.

Example:

```
float value = 23000.0;

ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_RGB_CAM, TY_FLOAT_EXPOSURE_TIME_US, value));
```

4.7 TY_INT_AE_TARGET_Y

Function:

This feature is used to set the target brightness for AEC (Auto Exposure Control) adjustment.

Example:

```
int32_t value = 200;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_RGB_CAM, TY_INT_AE_TARGET_Y, value));
```

4.8 TY_STRUCT_AEC_ROI

Function:

This feature is used to set the AEC region of interest. After setting up, the exposure time will be automatically adjusted based on the brightness of the region of interest.

Example:

```
TY_AEC_ROI_PARAM roi;

roi.x = 0;

roi.y = 0;

roi.w = 100;

roi.h = 100;

ASSERT_OK(TYSetStruct(hDevice, TY_COMPONENT_RGB_CAM, TY_STRUCT_AEC_ROI, &roi, sizeof(roi)));
```

4.9 TY_BOOL_AUTO_EXPOSURE

Function:

This feature is used to enable/disable RGB auto exposure.

Example:

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_EXPOSURE, value));
```

4.10 TY_BOOL_AUTO_AWB

Function:

This feature is used to enable/disable RGB auto white balance.

Example:

```
bool value = true;

ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_RGB_CAM, TY_BOOL_AUTO_AWB, value));
```

4.11 AUTO_ISP

This feature is used for software ISP functionality developed specifically for the specified RGB sensor, involving three APIs: *TYISPCreate()*, *ColorIspInitSetting()*, and *TYISPUpdateDevice()*.

Note:

TY_INT_ANALOG_GAIN will be set to 1 after the auto ISP function is enabled.

5 Features Under the *IR* Component

5.1 TY_INT_EXPOSURE_TIME

Function:

This feature is used to set left and right IR exposure time.

Example:

```
int32_t value = 100;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_EXPOSURE_TIME, value));
```

5.2 TY_FLOAT_EXPOSURE_TIME_US

Function:

This feature is used to set the absolute true value of left and right IR exposure time.

Unit: us.

Example:

```
float value = 23000.0;

ASSERT_OK(TYSetFloat(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_FLOAT_EXPOSURE_TIME_US, value));
```

5.3 TY_INT_ANALOG_GAIN

Function:

This feature is used to set left and right IR analog gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_ANALOG_GAIN, value));
```

5.4 TY_INT_GAIN

Function:

This feature is used to set left and right IR gain.

Example:

```
int32_t value = 2;

ASSERT_OK(TYSetInt(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_INT_GAIN, value));
```

5.5 TY_BOOL_UNDISTORTION

Function:

This feature is used for left and right IR distortion correction switch. When enabled, distortion correction is performed. Default state is disabled.

Example:

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_UNDISTORTION, value));
```

Note:

This feature is linked between the left and right IR components. Modifying this parameter on either component will synchronously affect both IR components. In case of conflicting values, the system automatically applies the last configured setting.

5.6 TY_BOOL_HDR

Function:

This feature is used as left and right IR HDR switch.

Example:

```
bool value = true;
ASSERT_OK(TYSetBool(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BOOL_HDR, value));
```

5.7 TY_BYTEARRAY_HDR_PARAMETER

Function:

This feature is used to set HDR exposure ratio parameters.

Example:

1.Get the size of the HDR array

```
uint32_t hdr_size;
ASSERT_OK(TYGetByteArraySize(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
&hdr_size));
printf("hdr_size %d\n", hdr_size);
```

2.Get default HDR parameter values

```
uint32_t hdr_param[8];
hdr_param[0] = -1;
```

```
hdr_param[1] = -1;
ASSERT_OK(TYGetByteArray(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
(uint8_t*)&hdr_param[0], hdr_size));
printf("default %d  %d\n", hdr_param[0], hdr_param[1]);
```

3.Set HDR parameter values

```
hdr_param[0] = 0;
hdr_param[1] = 0;
ASSERT_OK(TYSetByteArray(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_BYTEARRAY_HDR_PARAMETER,
(uint8_t*)&hdr_param[0], hdr_size));
```

6 Features Under the *Storage* Component

1. Export original calibration parameter file:

```
.\DumpCalibInfo.exe -id 207000151696 -cs 0 -ds 0 -out_json FM855-E1-G.json
```

-cs 0: Set RGB resolution to the first one in the list.

-ds 0: Set depth resolution to the first one in the list.

-out_json: Specify output file name and path.

2. Export current calibration parameter file:

```
.\DumpCalibInfo.exe -id 207000151696 -cs 1 -ds 2  -mode 0 -out_json FM855-E1-G.json
```

-mode 0: Read calibration parameters after current settings. The intrinsic parameters will change with resolution. Other modes use original calibration parameters.

6.1 TY_BYTEARRAY_CUSTOM_BLOCK

Function:

This feature is used to get the storage space size of *custom_block.bin*.

Example:

```
uint32_t block_size;
ASSERT_OK( TYGetByteArraySize(hDevice, TY_COMPONENT_STORAGE, TY_BYTEARRAY_CUSTOM_BLOCK,
&block_size) );
LOGD("custom block bin size %d", block_size);
```

6.2 TY_BYTEARRAY_ISP_BLOCK

Function:

This feature is used to get the storage space size of *isp_block.bin* file.

Example:

```
uint32_t block_size;
ASSERT_OK(TYGetByteArraySize(hDevice, TY_COMPONENT_STORAGE, TY_BYTEARRAY_ISP_BLOCK,
&block_size));
LOGD("isp block bin size %d", block_size);
```

7 Calibration Related Features

7.1 TY_STRUCT_CAM_INTRINSIC

Function:

This feature is used to get camera intrinsic parameters API, which can be used to obtain intrinsic parameters for depth, color, ir_left, and right_ir.

Example:

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_RGB_CAM;
featureID = TY_STRUCT_CAM_INTRINSIC;
TY_CAMERA_INTRINSIC intri;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &intri, sizeof(TY_CAMERA_INTRINSIC)));
LOGD("===%23s%f %f %f", "", intri.data[0], intri.data[1], intri.data[2]);
LOGD("===%23s%f %f %f", "", intri.data[3], intri.data[4], intri.data[5]);
LOGD("===%23s%f %f %f", "", intri.data[6], intri.data[7], intri.data[8]);
```

7.2 TY_STRUCT_EXTRINSIC_TO_DEPTH

Function:

This feature is used to get extrinsic parameters from ir_right/rgb to left ir.

Example:

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_IR_CAM_RIGHT;
featureID = TY_STRUCT_EXTRINSIC_TO_DEPTH;
TY_CAMERA_EXTRINSIC extri;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &extri, sizeof(TY_CAMERA_EXTRINSIC)));
LOGD("===%23s%f %f %f %f", "", extri.data[0], extri.data[1], extri.data[2], extri.data[3]);
LOGD("===%23s%f %f %f %f", "", extri.data[4], extri.data[5], extri.data[6], extri.data[7]);
LOGD("===%23s%f %f %f %f", "", extri.data[8], extri.data[9], extri.data[10], extri.data[11]);
LOGD("===%23s%f %f %f %f", "", extri.data[12], extri.data[13], extri.data[14], extri.data[15]);
```

7.3 TY_STRUCT_CAM_DISTORTION

Function:

This feature is used for distortion parameters, which can be used for distortion correction of rgb, ir_left, and ir_right.

Example:

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_RGB_CAM;
featureID = TY_STRUCT_CAM_DISTORTION;
TY_CAMERA_DISTORTION dist;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &dist, sizeof(TY_CAMERA_DISTORTION)));
LOGD("===%23s%f %f %f %f", "", dist.data[0], dist.data[1], dist.data[2], dist.data[3]);
LOGD("===%23s%f %f %f %f", "", dist.data[4], dist.data[5], dist.data[6], dist.data[7]);
LOGD("===%23s%f %f %f %f", "", dist.data[8], dist.data[9], dist.data[10], dist.data[11]);
```

7.4 TY_STRUCT_CAM_RECTIFIED_INTRI

This feature is used to get rectified intrinsic parameters for ir_left/ir_right.

Example:

```
TY_COMPONENT_ID componentID;
TY_FEATURE_ID featureID;
componentID = TY_COMPONENT_IR_CAM_LEFT;
featureID = TY_STRUCT_CAM_RECTIFIED_INTRI;
TY_CAMERA_INTRINSIC intri;
ASSERT_OK(TYGetStruct(hDevice, componentID, featureID, &intri, sizeof(TY_CAMERA_INTRINSIC)));
LOGD("===%23s%f %f %f", "", intri.data[0], intri.data[1], intri.data[2]);
LOGD("===%23s%f %f %f", "", intri.data[3], intri.data[4], intri.data[5]);
```

7.5 TY_STRUCT_CAM_RECTIFIED_ROTATION

Function:

This feature is used to obtain IR rotation from new config, currently mainly for V-series IR-based calibration.

Example:

```
TY_CAMERA_ROTATION rotation;
  ASSERT_OK(TYGetStruct(hDevice, TY_COMPONENT_IR_CAM_LEFT, TY_STRUCT_CAM_RECTIFIED_ROTATION,
&rotation, sizeof(rotation)));
  LOGD("===%23s%f %f %f", "", rotation.data[0], rotation.data[1], rotation.data[2]);
  LOGD("===%23s%f %f %f", "", rotation.data[3], rotation.data[4], rotation.data[5]);
  LOGD("===%23s%f %f %f", "", rotation.data[6], rotation.data[7], rotation.data[8]);
```

8 Log Related APIs

8.1 TYSetLogLevel()

Function:

This feature is used to set the log output level.

```
typedef enum TY_LOG_LEVEL_LIST{
 TY_LOG_LEVEL_VERBOSE  = 1,
 TY_LOG_LEVEL_DEBUG    = 2,
 TY_LOG_LEVEL_INFO     = 3,
 TY_LOG_LEVEL_WARNING  = 4,
 TY_LOG_LEVEL_ERROR    = 5,
 TY_LOG_LEVEL_NEVER    = 9,
}TY_LOG_LEVEL_LIST;
typedef int32_t TY_LOG_LEVEL;
```

Note:

Definition in this document: VERBOSE is the highest level, ERROR is the lowest level of logging.

Example::

```
ASSERT_OK(TYSetLogLevel(TY_LOG_LEVEL_DEBUG)); //Set LOG level to DEBUG
```

Runing Result:

VERBOSE ⊃ DEBUG ⊃ INFO ⊃ WARNING ⊃ ERROR.

Setting *TY_LOG_LEVEL_NEVER* will not output SDK logs.

8.2 TYSetLogPrefix()

Function:

This API is used to set the log prefix, depends on *TYSetLogLevel* property.

Example::

```
ASSERT_OK(TYSetLogPrefix("PERCIPIO SDK"));
```

8.3 TYAppendLogToFile()

Function:

This API is used to output logs of specified level and below to a specified file.

Example::

```
ASSERT_OK(TYAppendLogToFile("test_log.txt", TY_LOG_LEVEL_DEBUG));
```

8.4 TYAppendLogToServer()

Function:

This API is used to send logs of specified level and below to TCP server.

Example::

```
ASSERT_OK(TYAppendLogToServer("tcp", "192.168.2.70", 8000, TY_LOG_LEVEL_DEBUG));
```

8.5 TYRemoveLogFile()

Function:

This API is used to close log file output and release resources.

Example::

```
ASSERT_OK(TYRemoveLogFile("test_log.txt"));
```

8.6 TYRemoveLogServer()

Function:

This API is used to disconnect from the log server.

Example::

```
ASSERT_OK(TYRemoveLogServer("tcp", "192.168.2.70", 8000));
```

9 Other Common APIs

9.1 TYHasFeature()

Function:

This API is used to check whether the camera has a specific function.

Example:

```
bool has_feature = false;
ASSERT_OK(TYHasFeature(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1,
&has_feature));
```

Note:

If there's no corresponding Component, TYHasFeature reports -1008 error (TY_STATUS_INVALID_COMPONENT).

9.2 TYGetFeatureInfo()

Function:

This feature is used to get feature attributes such as: *isValid*, *accessMode*, *writableAtRun*, *componentID*, *featureID*, etc.

Example:

```
TY_FEATURE_INFO feature_info;
int32_t err = (TYGetFeatureInfo(hDevice, TY_COMPONENT_DEPTH_CAM, TY_INT_SGBM_SEMI_PARAM_P1,
&feature_info));
LOGD("API return %d ,  isValid %d , accessMode %d  writeableatrun %d ,\n", err,feature_info.isValid,
feature_info.accessMode, feature_info.writableAtRun);
```

9.3 TYGetDeviceFeatureNumber()

Function:

This feature is used to get the number of features under a component.

Example:

```
uint32_t feature_number = 0;
ASSERT_OK(TYGetDeviceFeatureNumber(hDevice, TY_COMPONENT_DEPTH_CAM, &feature_number));
LOGD("feature_number %d", feature_number);
```

9.4 TYGetDeviceFeatureInfo()

Function:

This feature is used in combination with *TYGetDeviceFeatureNumber* to get information about all features under a component.

Example:

```
uint32_t feature_number = 0;
ASSERT_OK(TYGetDeviceFeatureNumber(hDevice, TY_COMPONENT_STORAGE, &feature_number));
LOGD("feature_number %d", feature_number);


std::vector <TY_FEATURE_INFO> feature_info(feature_number);
uint32_t entry_count = 0;
ASSERT_OK(TYGetDeviceFeatureInfo(hDevice, TY_COMPONENT_STORAGE, &feature_info[0], feature_number,
&entry_count));
for (uint i = 0; i < feature_number; i++) {
LOGD("feature name [%s], writebleAtRun[%d], TY_FEATURE_ID [%x]", feature_info[i].name,
feature_info[i].writableAtRun, feature_info[i].featureID);
}
```

9.5 write_parameters_to_storage()

Function:

This API is a new function added in *SDK V3.6.51* and above, used to write JSON files exported by *Percipio Viewer* into the camera's *custom_block.bin* file.

Example:

1. Use *Percipio Viewe*r (V*2.5.0* and above) to generate a JSON file of current camera parameters (only saves parameters adjusted after opening the camera with *Percipio Viewer*)
2. Run: *SimpleView_SaveLoadConfig -id 207000139089 -s xxxx.json*

Note:

This step writes the JSON file to the camera.

Verification method:

After powering off and restarting the camera, use *Percipio Viewer* to directly load

parameters and observe if parameters have changed.

Note:

Starting from *SDK V3.6.65*, Huffman compression is used when writing parameters, so directly viewing the file shows garbled text; verification must be done through loading

9.6 load_parameters_from_storage()

Function:

This API is a new function added in *SDK V3.6.51* and above, used to load parameters from *custom_block.bin* file and save them to a local file.

Example:

Run *SimpleView_SaveLoadConfig -id 207000139089 -o xxxxxxx.json*

Note:

This step will load parameters from *custom_block.bin* and output the parameters to the *xxxxxxx.json* file in the program directory

9.7 clear_storage()

Function:

This API is used to clear the contents of the camera's *custom_block.bin* storage area.

Example:

```
ASSERT_OK(clear_storage(hDevice));
```

9.8 selectDevice()

Function:

This API is used to open Percipio cameras using a specified interface, based on ID or IP.

Example:

1. Specify using network interface to open Percipio camera

```
std::vector<TY_DEVICE_BASE_INFO> selected;
  ASSERT_OK( selectDevice(TY_INTERFACE_ETHERNET, ID, IP, 1, selected) );
```

2. Open camera using specified ID or IP

```
// ID = "207000149647";
IP = "192.168.6.150";
std::vector<TY_DEVICE_BASE_INFO> selected;
```

```
ASSERT_OK( selectDevice(TY_INTERFACE_ETHERNET, ID, IP, 1, selected) );
```

9.9 TYOpenInterface()

Function:

This API is used to open a specified network interface.

If the network card MAC address is: *88-a4-c2-b1-35-e3* (view with ipconfig /all), and IP address is: *192.168.6.45* (corresponding little-endian hexadecimal storage is *2d06a8c0*), then when specifying the network card, it needs to be written as "*eth-88-a4-c2-b1-35-e32d06a8c0*", note case sensitivity is required.

Example:

```
char* iface_id = "eth-88-a4-c2-b1-35-e32d06a8c0";

ASSERT_OK(TYOpenInterface(iface_id, &hIface));
```

9.10 TYOpenDevice()

Function:

This API is used to open Percipio cameras. If the camera can be opened normally, the API returns 0. Any other return result indicates an exception.

9.11 parse_firmware_errcode()

Function:

This feature is used to print the error code corresponding to -1024 error.

Example:

```
TY_FW_ERRORCODE err_code;

int32_t TYOpenDevice_err = ( TYOpenDevice(hIface, selectedDev.id, &hDevice ,&err_code) );

printf("TYOpenDevice_err %d\n", TYOpenDevice_err);

parse_firmware_errcode(err_code);
```

9.12 TYGetDeviceXMLSize()

Function:

This feature is used to get the size of the camera's XML file. This API was first added in *SDK V3.6.52*.

Example:

```
uint32_t size;
```

```
TYGetDeviceXMLSize(hDevice, &size);

LOGD("XML size %d", size);
```

9.13 TYGetDeviceXML()

Function:

This feature is used to get the camera's *XML* file. First added in *SDK V3.6.52*.

Example:

```
uint32_t size;

ASSERT_OK(TYGetDeviceXMLSize(hDevice, &size));

LOGD("XML size %d", size);

std::vector<char> xmlBuffer(size);

ASSERT_OK(TYGetDeviceXML(hDevice, xmlBuffer.data(),size,&size));
```

1. Method to print all API content:

```
std::cout << std::string(xmlBuffer.data(), size) << std::endl;
```

2. Print content of specified line:

```
int32_t lineCount = 0;

size_t start = 0;

size_t end = 0;

while (lineCount < 50 && end != std::string::npos) {

end = std::string(xmlBuffer.data(), size).find('\n', start);

if (end != std::string::npos) {

lineCount++;

if (lineCount == 49) {

std::string line(xmlBuffer.data() + start, end - start);

printf("The firmware version is: %s\n", line.c_str());

break;

}

start = end + 1;

}

}
```

9.14 TYImageMode2

Function:

This feature is used when the resolution 1024x768 is not defined in *TY_RESOLUTION_MODE_LIST*, making *TY_IMAGE_MODE_DEPTH16_1024x768* an undefined identifier. In this case, the *TYImageMode2()* interface is needed to construct a parameter.

Example:

```
TY_IMAGE_MODE ImageMode = TYImageMode2(TY_PIXEL_FORMAT_DEPTH16, 1024, 736);
ASSERT_OK(TYSetEnum(hDevice, TY_COMPONENT_DEPTH_CAM, TY_ENUM_IMAGE_MODE, ImageMode));
```

Note:

Usage is not limited to depth image resolution and can be extended to RGB and IR resolutions.

Percipio.XYZ is an industry leading provider of 3D cameras. We provide a broad range of 3D camera products to meet requirements from various applications, such as industrial, automotive, inspection, logistics, medical, education, security and commercial etc. We will continue to develop and optimize our product roadmap to support more 3D vision applications.

Percipio is an independent vendor of 3D machine vision solutions. We provide products and services to system integration customers rather than end users. This marketing strategy allows us to serve multiple sectors and segments, and also means that our success will be based on our customer's success. Together with our customer's industry specific expertise, we can support end users with implementing machine intelligence, which will improve productivity and/or reduce cost.

Affordable 3D Machine Vision

Contact Us

Purchase : info@pcp3d.com
Technical : support@pcp3d.com
Website : www.pcp3d.com
Documentation : doc.percipio.xyz/cam/latest/en/

Statement

* Data mentioned in this document is subject to change without notice.
* The data mentioned in this document may vary due to environmental factors and other reasons. Percipio does not assume any consequences arising therefrom.

YouTube