

Release notes for weCat3D SDK 1.6.0 for C++

Version 1.6.0

Previous version 1.5.2

Release date 24.11.2022

Software description

The weCat3D SDK is the software development kit (SDK) for the easy integration of weCat3D sensors in user applications.

Programming example in programming languages

- C++

Example projects for development environments

- QT

Supported operation systems

- Windows 7 (x64) or higher
- Linux Ubuntu 14.04 (x64) or higher

Content

Content of the SDK package is divided into the latest (newest) SDKs and outdated (older) SDKs. The outdated SDK shows how to use the library in C#, C++ and Visual Studio as development environment.

Name	Description
Latest SDK	
weCat3D_SDK_Windows.1.6.0.478.zip	Newest SDK for Windows and QT example
weCat3D_SDK_Linux.1.6.0.478.tar.gz	Newest SDK for Linux and QT example
weCat3D_TCPIInterface_Windows_VS2013_C++.zip	Example how to use the TCP socket interface without library
Outdated SDKs	
Ethernet.zip	Outdated windows library
Documentation	Documentation of the outdated SDK
Licenses	Licenses used in the outdated SDK
weCat3D_SDK_Windows_VS2013_C#.zip	Outdated example in C# and VS2013
weCat3D_SDK_Windows_VS2013_C++.zip	Outdated example in C++ and VS2013

New Features

Range Image Function

The range image function allows to combine several profiles into a 2.5 D image (the depth information z is encoded into the grey color of the image).

Command `int EthernetScanner_GetRangeImage(void* pEthernetScanner, unsigned short* imageBuffer, int iBuffer, int iTimeoutPerScan, bool* bFrameLost=nullptr, int* picCntBuffer=nullptr, int* encoderBuffer=nullptr, unsigned int* timeStampBuffer = nullptr);`

Parameter 1 `void *`: a handle to the profile sensor returned by the function "EthernetScanner_Connect"

Parameter 2 `unsigned short*` : pointer to a raw buffer (Type "unsigned short") used by the function to write in the 16 Bit Grayscale Range Image.

Parameter 3 `int`: the length of the raw buffer passed in parameter 2. The length of the raw buffer should be larger than the number of measured points returned by the profile sensor. The number of elements of the buffer has to be at least `sensors maximum xResolution* number of profiles per image`.

Parameter 4 `int`: the value of the blocking time to wait for a new measured profile, until the function times out. The value 0 makes the function non-blocking (timeout in ms).

Parameter 5 `bool*`: (optional) pointer to a bool variable with (true) indicating a frame was lost during acquisition.

Parameter 6 `int*`: (optional) pointer to array of type int with the same size set by `nrProfilesPerScan` returning the picture count of each profile.

Parameter 7 `int*`: (optional) pointer to array of type int with the same size set by `nrProfilesPerScan` returning the encoder value of each profile.

Parameter 8 `unsigned int*`: (optional) pointer to array of type unsigned int with the same size set by `nrProfilesPerScan` returning the timestamp value of each profile.

Response `ETHERNETSCANNER_INVALIDHANDLE (-1000)` if the sensor handle (parameter 1) is NULL or invalid. In the case of a success call, the function will return

`ETHERNETSCANNER_OK (0)`. The function will return

`ETHERNETSCANNER_GETXZINONEWSCAN (-1)` if no new profile is available,

`ETHERNETSCANNER_GETXZIINVALIDBUFFER (-3)` if the length of the buffer is shorter than the data to be written,

`ETHERNETSCANNER_GETXZIINVALIDDLINDATA (-2)` if the DLL is not initialized.

Description With the weCat3D Range Image feature you can obtain multiple scans bundled as one ordered 16Bit 2D image, scaling the original z values as the images in intensity.

The settings of the range image can be configured by following ASCII commands:

ASCII commands related to range image function

Command `SetRangeImageNrProfiles=x\r`

Parameter values of x:

1 ... 1000

Default: 1**Description** Sets the number of profiles per image implicitly defining the height of the image.**Command** SetRangeImageXScale=x\r**Parameter** Values of x:

Floating point value 0..1

Default: xRangeMax/2**Description** Scale for mapping the x coordinates to the RangeImage pixels.**Command** SetRangeImageXOffset=x\r**Parameter** Values of x:

- xRangeMax/2 ... xRangeMax/2

Default: xRangeMax/xResolutionSensor**Description** Offset for mapping the x coordinates to the RangeImage pixels.**Command** SetRangeImageZScale=x\r**Parameter** Values of x:

Floating point value 0..1

Default: ZStart**Description** Scale mapping the z coordinates to the RangeImage intensity.**Command** SetRangeImageZOffset=x\r**Parameter** Values of x:

0 ... ZStart + ZRange

Default: ZRange/65535**Description** Offset for mapping the z coordinates to the RangeImage pixels.**NOTE!**

To either modify the coordinate range which is pictured in the range image or recalculate 3D coordinates from the acquired range image an offset value and scale factor are applied both in X and Z, similar to the GigeVision coordinate scale and offset features. These values describe the mapping between 3D coordinates and 2D range image pixels according to:

$$X(i)Coord = x(i)RangeImage * XScale + XOffset$$

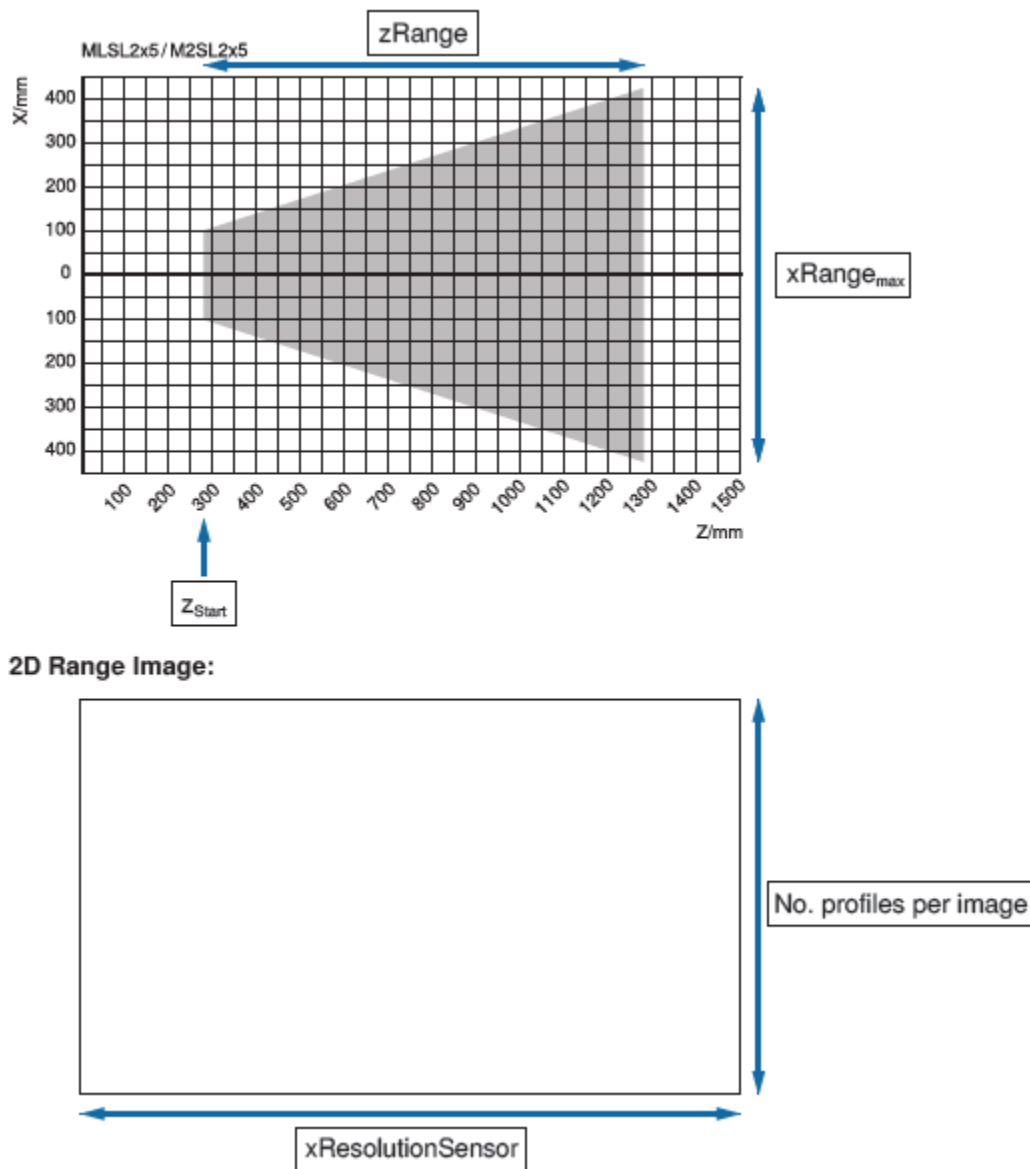
$$Z(i)Coord = l(i)RangeImage * ZScale + ZOffset$$

with X(i): pixelCoord of point i in 2D Image

and l(i): 16Bit Intensity value of point i in 2D Image

(please see figure for visualization the range parameters)

By default ScaleFactor and offset are initialized so that the maximum measurement range in X and the Sensors Z range are displayed completely in the Range Image.



Scales and Offsets can be read out by their corresponding ReadData commands

- GetRangeImageNrProfiles
- GetRangeImageXScale
- GetRangeImageXOffset
- GetRangeImageZScale
- GetRangeImageZOffset

Camera Image Function

The new function allows to receive the camera image of the sensor. With the camera image sources of artifacts in the profile can be better identified.

Command `int EthernetScanner_GetImage(void* pEthernetScanner, char *cBuffer, int iBuffer, unsigned int *puiWidth, unsigned int *puiHeight, unsigned int *puiOffsetX, unsigned int *puiOffsetZ, unsigned int *puiStepX, unsigned int *puiStepZ, unsigned`

```
int iTimeout)
```

Parameter 1 void *: a handle to the profile sensor returned by the function "Ether netScanner_Connect"

Parameter 2 char *: pointer to a 8Bit raw buffer (e.g. unsigned char) used by the function to pass the pixel wise Intensities of the Camera Image. The Buffer should at least have the size of the nr of Pixels read e.g. RoiX_Width *RoiZ_Height.

Parameter 3 int: the length of the raw buffer passed in

Parameter 4 unsigned int*: pointer to variable of type unsigned int to read width of image in pixel

Parameter 5 unsigned int*: pointer to variable of type unsigned int to read height of image in pixel

Parameter 6 unsigned int*: pointer to variable of type unsigned int to read X offset in pixel

Parameter 7 unsigned int*: pointer to variable of type unsigned int to read Z offset in pixel

Parameter 8 unsigned int*: pointer to variable of type unsigned int to read subsampling in X

Parameter 9 unsigned int*: pointer to variable of type unsigned int to read subsampling in Z

Parameter 10 unsigned int: the value of the blocking time to wait for a new measured profile, until the function times out. The value 0 makes the function non-blocking (timeout in ms).

Response ETHERNETSCANNER_INVALIDHANDLE (-1000) if the sensor handle (parameter 1) is NULL or invalid.

In the case of a success call, the function will return the size of the data written to the raw buffer.

The function will return ETHERNETSCANNER_GETXZINONEWSCAN (-1) if no new profile is available,

ETHERNETSCANNER_GETXZIINVALIDBUFFER (-3) if the length of the buffer given in parameter 1 to 5 is shorter than the data to be written,

Description The function is used to read out the camera image of the sensor in the currently defined ROI. Reading Camera Image is only possible in camera mode 1 (=camera image).

Related ASCII command to this function:

ASCII commands related to image function

Command SetCameraMode=x\r

Parameter Values of x:

0: Profile

1: Camera images

Default: 0

Description If camera mode is selected the image of the camera is shown/transferred via the interface. The external GigE Vision interface, uniVision, VisionApp Demo 3D and VisionApp 360 do not support the camera mode. Thus it should be not selected if these programs are used.

UDP Support

The SDK supports now also the data transfer using the UDP protocol introduced in firmware version 2.2.0.

Function to use UDP:

Command `void* EthernetScanner_ConnectUDP(char* chDestIP, char* chDestPort, char* chSrcIP, char* chSrcPort, char* chMode)`

Parameter 1 `char *chDestIP`: IP address of the profile sensor: „192.168.100.1“ \0 terminated

Parameter 2 `char *chDestPort`: Portnumber of the profile sensor: „32001“ \0 terminated

Parameter 3 `char *chSrcIP`: IP address of the networks interface card to which the sensor is connected \0 terminated

Parameter 4 `char *chSrcPort`: Free Port which is used to receive the sensor data \0 terminated

Response `void*` a handle to the profile sensor. A NULL pointer is returned in case of failure

Description This function will create a UDP connection to the weCat3D sensor. The function will return a handle to the profile sensor, which will be used by other functions.

NOTE!

As UDP is a connection less protocol a disconnect won't be detected by the SDK automatically. Furthermore it can't be guaranteed that data sent from the sensor won't get lost during transmission on high network loads.

Related ASCII commands (must not be set if the function `EthernetScanner_ConnectUDP`) are:

ASCII commands related to UDP function
<p>Command <code>SetUDPSocketPort=x\r</code></p> <p>Parameter Values of x: 1024 ... 65000</p> <p>Description Input of the host port number to which the sensors sends the data (reserved 32001/32002).</p>
<p>Command <code>SetUDPSocketIP=x\r</code></p> <p>Description Input of the host IP address to which the sensors sends the data (format aaa.bbb.ccc.ddd).</p>
<p>Command <code>SetUDPSocketStart=x\r</code></p> <p>Parameter Values of x: 0: End of UDP data transmission 1: Start UDP data transmission</p> <p>Description Activating/deactivating UDP data transmission.</p>

Bug fixes

Release notes for weCat3D SDK 1.6.0

Author Dr. Sascha Reinhardt / DE2-PM-PM

Date 21.11.2022 Version 1.0

Description	Solution
Conflicts if different version of neosmart pevents or tinyxml used in same project on Linux	fixed
GetEncoderTriggerFunction and GetEncoderCountDirection not readable from xml	fixed