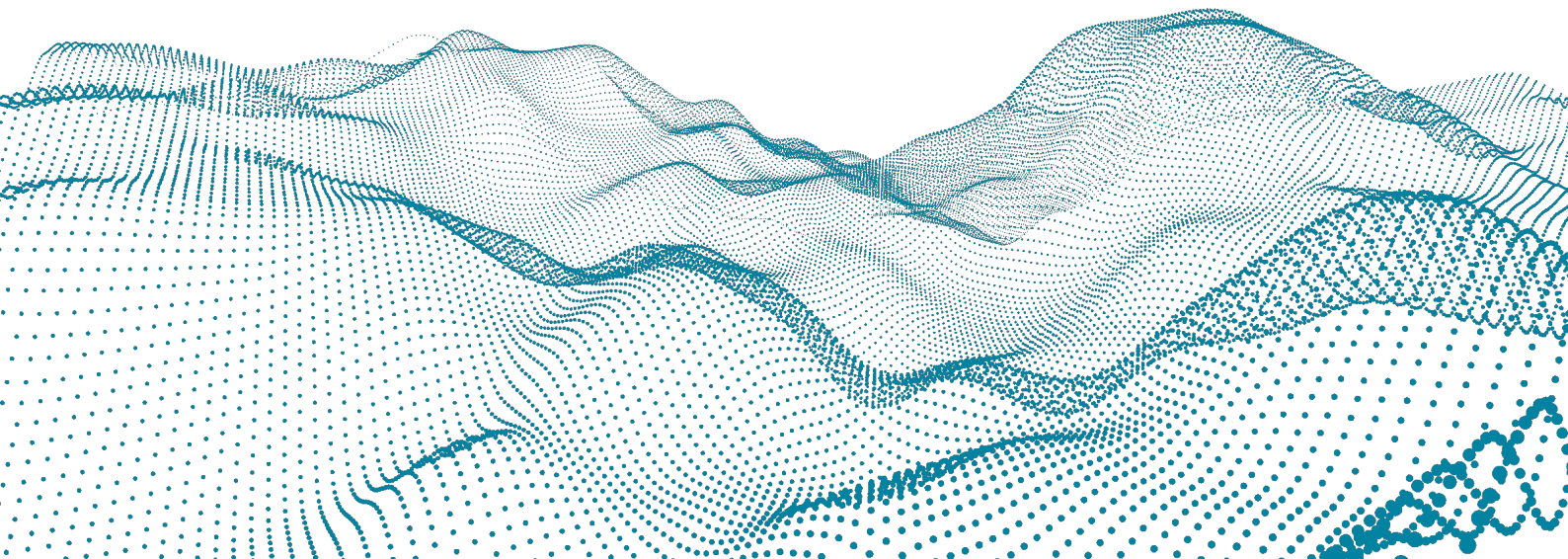
roboception

Roboception GmbH | July 2024

SGM® Producer

Release 24.07.0



Contents

1 COPYRIGHT

2 Introduction

The SGM® Producer is a software library for running stereo matching on a graphics card of a host computer for significantly increasing the frequency and reducing the latency for computing disparity images. It supports the rc_visard and rc_viscore. About 12.5 Hz can be reached for matching 1.2 MPixel images on an Nvidia GeForce RTX 2070. 4.6 Hz can be reached for matching 3 MPixel images on the same GPU. Higher frame rates are possible on more powerful Nvidia GPUs. Stereo matching without a graphics card on the CPU is also possible but significantly slower.

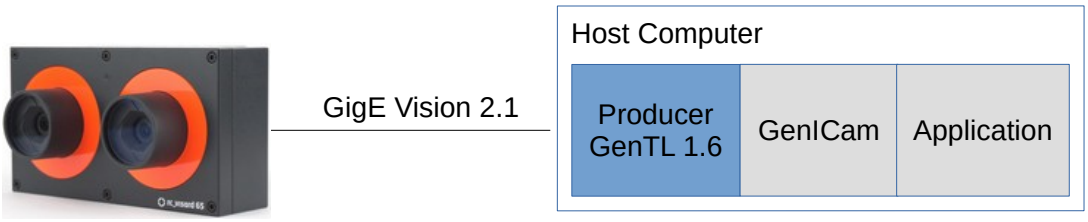


Figure 1: Overview of the rc_visard and host computer. The producer is highlighted in blue.

In a GenICam application, the producer is responsible for communicating with the device, which in case of the rc_visard uses the GigE Vision 2.1 standard.

The SGM® Producer replaces a standard GigE Vision producer. It is compatible with all GenICam applications that use the GenTL 1.6 standard, e.g. Halcon from MVTec.

2.1 Prerequisites

The requirements on the host computer are:

- Supported operating systems include Windows 10 (64 bit), Ubuntu 20.04 (64 bit), Ubuntu 22.04 (64 bit) and other 64 bit Linux distributions
- On x86 architecture, the CPU must support the AVX2 instruction set. Almost all x86 CPUs that were manufactured from 2014 onward, especially the more powerful ones, provide this support.
- Nvidia GPU with compute capability 3.5 (i.e. Kepler architecture) or higher is highly recommended. The full functionality is also available without GPU support, but stereo matching will be very slow in this case.
- Supported camera that is connected to the host computer.

Supported cameras are:

- rc_visard with firmware version 21.04.0 or higher
- rc_viscore

The amount of required GPU memory and possible frame rates depend on the camera resolution, the configured depth quality, depth range and memory limitation using the RC_SGM_MAXMEM environment variable.

For rc_visard, a depth image frame rate of 25 Hz can be reached on mid-range Nvidia GPUs. The maximum amount of required GPU memory is below 1200 MB in all modes.

For a 12 Mpixel rc_viscore with 16 mm lenses, the following table shows the required GPU memory and resulting frame rates on an Nvidia GeForce RTX 2070 card for some example configurations. rc_viscore has an image resolution of 4112 x 3008 pixels and can deliver images with a maximum frame rate of 9 Hz.

DepthQuality	DepthMinDepth	DepthMaxDepth	RC_SGM_MAXMEM	GPU Memory	FPS
Low				553 MB	9 Hz
Medium				1057 MB	9 Hz
High				6043 MB	2.5 Hz
High	0.75 m			4343 MB	3.4 Hz
High	0.75 m	3 m	3312 MB	3639 MB	4.0 Hz
Full	1.6 m	3 m	7500 MB	7789 MB	1.8 Hz

Using HDR mode requires about 400 MB more GPU memory for rc_viscore.

Please note that stereo matching in full quality (i.e. 12 Mpixel) is in most situations not needed and not recommended. Instead, we recommend using the high quality setting with appropriate limitations on the depth range and maximum SGM memory as shown in the table above.

3 Installation

3.1 Windows

The software for Windows is provided as a ZIP archive that can be downloaded from https://roboception.com/en/download/.

The ZIP archive can be unpacked anywhere on the local hard drive. The main directory of the ZIP archive contains the file rc_sgm_producer.cti. This directory must be added to an environment variable named GENICAM_GENTL64_PATH so that third party applications can find the producer.

- Type “env” in the Windows search field (on a German Windows system type “Umgebung”)
- Select “Edit the system environment variables” (if you have administrator rights) or “Edit environment variables for your account)”
- Check if GENICAM_GENTL64_PATH exists and create it as new if it does not exist
- Add the path to the directory in which rc_sgm_producer.cti is located to this environment variable
- Several paths can be specified with ‘;’ as separation character
- Leave the dialog with “OK”

Important: The main directory must also be added to the ‘PATH’ environment variable so that dependencies of the producer are found.

3.2 Ubuntu

The software for Ubuntu LTS consists of the debian package rc−sgm−producer*.deb. It can be downloaded from https://roboception.com/en/download/. The package can be installed with:

```
sudo dpkg −i rc−sgm−producer*.deb
```

If there are errors due to missing packages, they can be installed by calling:

```
sudo apt install −−fix−broken
```

The directory /opt/rc_sgm_producer/lib contains the file rc_sgm_producer.cti, which should be included in the environment variable GENICAM_GENTL64_PATH so that third party applications can find the producer. In the bash shell, this can be done by:

```
export GENICAM_GENTL64_PATH=$GENICAM_GENTL64_PATH:/opt/rc_sgm_producer/lib
```

The command line can be added to .bashrc for permanently including the path.

3.3 Other Linux Distributions

For other Linux distributions, the producer can be installed from the archive file rc_sgm_producer*.tgz. The producer in the TGZ package was built with minimal dependencies and GUI tools are provided as AppImages to support a wide range of Linux distributions. It can be downloaded from https://roboception.com/en/download/.

The archive can be unpacked anywhere. It is recommended to unpack it under /opt as user root, e.g.

```
cd /opt
sudo tar xvfz <download−directory>/rc_sgm_producer*.tgz
```

The sub-directory lib contains the file rc_sgm_producer.cti, which must be included in the environment variable GENICAM_GENTL64_PATH. In the bash shell, this can be done by:

```
export GENICAM_GENTL64_PATH=$GENICAM_GENTL64_PATH:/opt/rc_sgm_producer/lib
```

The command line can be added to .bashrc for permanently including the path.

3.4 Environment Variables

The following environment variables influence the behavior of the producer:

- RC_SGM_PRODUCER_LEVEL: This variable enables printing log information. By default, logging is done on stdout of the application that uses the producer. See RC_SGM_PRODUCER_LOG for logging into a file. The possible log levels are given below. Higher levels include lower ones.
  - 0: Off (default)
  - 1: Fatal
  - 2: Error
  - 3: Warning
  - 4: Info
  - 5: Debug
  - 6: Trace (not recommended for general use as it can slow down the producer significantly)
- RC_SGM_PRODUCER_LOG: This variable can be defined with the full path and name of the log file. Logging into this file is done according to the log level (see RC_SGM_PRODUCER_LEVEL).
- RC_SGM_MAXMEM: Maximum amount of memory in MB that can be used as temporary memory for stereo matching. If a compatible GPU is available, this is memory on the GPU. The depth range will be reduced if this amount of memory is reached. The default is 0, which means unlimited.

4 Tools for setup, calibration and testing

Note: The tools are not contained in the TGZ package for Linux ARM64. Use the Linux x86_64 or Windows packages for using these tools.

4.1 rc_check tool (rc_viscore only)

The tool rc_check can be used for checking the network connection and basic setup of an rc_viscore device. In case of a misconfiguration, the tool also helps to reset the rc_viscore to factory defaults. The usage is self explanatory.

4.2 rc_calib tool (rc_viscore only)

The tool rc_calib can be used for checking and adjusting the focus and calibration of rc_viscore devices. rc_visard devices allow calibration in the Web GUI.

After starting the tool, select *File* / *Connect camera ...*, choose the rc_viscore from the list and specify the size of the calibration grid. Optionally limit the throughput, in case the cameras are connected via a 1 Gbit switch. After confirming the dialog, the tool will show the live images from the left and right cameras.

For checking and adjusting the focus, select *Adjust Focus* and follow the instructions in the rc_viscore documentation at https://doc.rc-viscore.com/en/installation.html#adjust-focus.

For checking the calibration and recalibration, select the appropriate calibration option. The procedure is the same as described in the rc_cube manual at https://doc.rc-cube.com/latest/en/camera_calibration.html#verify-calibration.

It is mandatory to always check the calibration after mounting the rc_viscore, changing the focus or aperture.

4.3 rc_viewer tool

This is a live 3D viewer for rc_visard and rc_viscore devices. After starting the tool, select *File* / *Connect camera ...*. The dialog permits to choose between SGM producer and standard producer. For an rc_viscore, the SGM Producer must be used. For an rc_visard, stereo processing is done on the host computer with the SGM producer and on-board the rc_visard with a standard producer. Optionally, for an rc_viscore the bandwidth can be limited in case the cameras are connected via a 1 Gbit switch.

After confirming the connection dialog, the tool will show the live 3D view. If this is not the case, press the Reset to defaults button on the camera and depth tab panel or check for connection errors.

Best results are retrieved by supporting 3D acquisition with the random dot projector. For continuous streaming, select ExposureAlterateActive for Out1 / Projector on the depth tab panel. For single shot acquisition with projector, select SingleFrameOut1 as Acquisition mode on the depth tab panel.

The framerate that is reached with rc_viewer tool can be lower than with the SGM® Producer alone, especially for rc_viscore, due to computational overhead for on-the-fly 3D mesh creation and rendering.

5 GenICam Nodemap

The producer offers the same nodemap parameters for rc_visard and rc_viscore devices, although some may be disabled, depending on device capabilities. Most of the parameters are standard GenICam features. Their meaning and usage is described in the GenICam Standard Features Naming Convention (SFNC), e.g. https://www.emva.org/wp-content/uploads/GenICam_SFNC_v2_7.pdf. Proprietary parameters are described below.

The appendix shows the full nodemap of an rc_viscore device as an example.

5.1 Proprietary GenICam Parameters

5.1.1 Requesting Images

- *ComponentSelector*: Selects an image type that can then be enabled or disabled with the boolean parameter *ComponentEnable*.
  - *Intensity*: Rectified left camera image. By default, only this component is enabled.
  - *IntensityCombined*: Synchronized rectified left and right camera images stacked vertically.
  - *Disparity*: 16 bit disparity image (see 'Computing Point Cloud' below).
  - *Confidence*: 8 bit confidence image. 0 means no confidence and 255 means full confidence.
  - *Error*: 8 bit error image.
  - *RawCombined*: Synchronized raw left and right camera images stacked vertically. This component is only available for *AcquisitionMultiPartMode=SingleComponent*. It is not available for rc_visard.
  - *Calibration*: Annotated raw camera images, used for calibration (see 'Calibration Interface' below). This component is only available for *AcquisitionMultiPartMode=SingleComponent*. It is not available for rc_visard.
- *ComponentEnable*: Enables or disables the selected component with *True* or *False*

5.1.2 Projector Control and Synchronization

It is assume that there is a random dot projector connected to GPIO Out1, which is optional for rc_visard, but always the case for rc_viscore. After setting *LineSelector=Out1*, the projector can be turned on and off by setting *LineSource* to *High* or *Low*. Setting *LineSource* to *ExposureActive* turns the projector on for every image, while *ExposureAlternateActive* turns it on for every second image. If *ExposureAlternateActive* is on, disparity images will only be computed from images where the projector was on.

- *RcLineRatio* (not available for rc_visard): Defines the fraction of the exposure time for which the projector is turned on, e.g. 0.8 means that the projector is on for 80% of the exposure time. This feature can be used to reduce the brightness of the projector.
- *AcquisitionMultiPartMode*: Defines how images (e.g. left camera and disparity image) are synchronized.
  - *SingleComponent*: Means that all images are provided in individual buffers whenever they become available, e.g. a camera image will be provided as soon as it is rectified and the corresponding disparity image will be provided later. The application can synchronize both by their timestamp.
  - *SynchronizedComponents*: Means that all images from all components will be synchronized by their timestamp and provided in a multi-part buffer. If there is only one component enabled, then this is the same as *SingleComponent*.
  - *SynchronizedAlternateComponents*: This is the same as SynchronizedComponents, except that if the projector is on for every second image (i.e. *LinceSource* is *ExposureAlternateActive* for *Out1*) and if camera images and disparity images are requested (i.e. *Intensity* or *IntensityCombined* is enabled and at least one of *Disparity*, *Confidence* or *Error*is enabled), then *Intensity* or *IntensityCombined* images will be captured without projector, after the images with projection that are used for computing the disparity image. The timestamp of the disparity image is used for

the multi-part buffer. This mode is useful in most applications, because the disparity image benefits from the projected pattern, but the camera image is not disturbed by the pattern.

- *AcquisitionAlternateFilter*: Defines filtering of *Intensity* or *IntensityCombined* images. This only has an effect if *LineSource* is set to *ExposureAlternateActive* and if *AcquisitionMultiPartMode* is either set to *SingleComponent* or if there is only the component *Intensity* or *IntensityCombined* enabled.
  - *Off*: Disables the filter.
  - *OnlyHigh*: Provides only images when the projector is on, i.e. *Out1* is high.
  - *OnlyLow*: Provides only *Intensity* or *IntensityCombined* images when the projector is off, i.e. *Out1* is low. *Disparity*, *Confidence* or *Error* images are still computed and provided from images with projection.

5.1.3 Exposure Control

- *ExposureAuto*: Is a standard GenICam parameter to control the auto exposure. The rc_sgm_producer provides some custom modes.
  - *Off*: No auto exposure. Exposure time and gain can be manually controlled via standard parameter.
  - *Continuous*: Normal auto exposure mode is enabled.
  - *AdaptiveOut1*: Tracks the exposure difference between images with *Out1* low and high (i.e. projector off and on). When *Out1* is low, the images are under-exposed by this exposure difference to avoid over-exposure for when *Out1* triggers an external projector. The resulting exposure difference is given in *ChunkRcOut1Reduction*. This exposure mode is recommended for using the depth acquisition mode *SingleFrameOut1* with an external projector when changes in the brightness of the scene should be considered at all times.
  - *Out1High*: Adapts the exposure time using only images with *Out1* high. Images where *Out1* is low are not considered at all, which means, that the exposure time does not change when only images with *Out1* low are acquired. This exposure mode is recommended for using the depth acquisition mode *SingleFrameOut1* with an external projector when changes in the brightness of the scene should only be considered when *Out1* is high.
  - *HDR*: The HDR mode computes high-dynamic-range images by combining images with different exposure times to avoid under-exposed and over-exposed areas. This decreases the frame rate and is only suitable for static scenes.
- *ExposureTimeAutoMax*: In auto-exposure mode, the maximum exposure time is specified. If the maximum exposure time is reached, then the sensor increases the gain to increase the images' brightness. Limiting the exposure time is useful for avoiding or reducing motion blur during fast movements. However, larger gains introduce noise into the image. The best trade-off depends on the application.
- *ExposureRegionOffsetX*, *ExposureRegionOffsetY*, *ExposureRegionWidth*, *ExposureRegionHeight*: Region in the image that is used by the auto exposure function to determine the image brightness.
- *RcExposureAutoAverageMin*, *RcExposureAutoAverageMax*: The auto exposure tries to set the exposure time and gain factor such that the average image intensity is between a maximum and minimum brightness. The maximum can be reached if there is no saturation (e.g. due to reflections). To avoid saturation, the auto exposure can reduce the brightness to the given minimum.

5.1.4 Depth Image Computation

- *DepthAcquisitionMode*:
  - *SingleFrame*: Stereo matching is performed for the next image pair after calling *DepthAcquisitionTrigger*.
  - *SingleFrameOut1*: Works similar to *SingleFrame*, but in addition the GPIO Out1 of the sensor is set to high for the time of capture to trigger a projector.
  - *Continuous*: Continuous computation of depth images.
- *DepthExposureAdaptTimeout*: Maximum time in seconds to wait after triggering an acquisition in *SingleFrame* or *SingleFrameOut1* mode until auto exposure has finished adjustments.

- *DepthQuality*: The quality can be *Full*, *High*, *Medium* or *Low*. Reducing the quality reduces the resolution of depth images, but increases the possible frame rate.
- *DepthDoubleShot*: Enabling this option will lead to denser disparity images, but increases processing time. For scenes recorded with a projector in *SingleFrameOut1* acquisition mode, or in continuous acquisition mode with the projector in *ExposureAlternateActive* mode, holes (e.g. caused by reflections of the projector) are filled with depth information computed from an image without projector pattern. This is only suitable for scenes that do not change during capturing! If *ExposureAlternateActive* is not used, holes are filled with depth information computed from a downscaled version of the same image.
- *DepthStaticScene*: This performs an averaging over 8 camera images for reducing noise which improves stereo matching, but increases the latency. This parameter only affects Full and High quality. It is ignored for Medium and Low quality. This is suitable for static scenes!
- *DepthSmooth*: Enables advanced smoothing of disparity values.
- *DepthFill*: Disparity tolerance for filling holes. Larger values can decrease the number of holes, but the filled values can have larger errors.
- *DepthSeg*: Small connected disparity areas of smaller size will be set to invalid. This is useful for removing erroneous disparities. Larger values may remove real objects, too.
- *DepthMinConf*: Minimum confidence as value between 0.5 and 1. Measurements with lower values will be removed. The confidence is the probability that the measured disparity is not an outlier, i.e. false measurement.
- *DepthMinDepth*: Minimum working distance in meters. Smaller measurements will be removed. Larger values implicitly reduce the disparity range and the computation time.
- *DepthMaxDepth*: Maximum working distance in meters. Larger measurements will be removed.
- *DepthMaxDepthErr*: Maximum depth error in meters. Measurements with larger errors will be removed.
- *FocalLengthFactor*: The focal length scaled to an image width of 1 pixel. To get the focal length in pixels for a certain image, this value must be multiplied by the width of the received image. See also the standard parameter *Scan3dFocalLength*.
- *Baseline*: This parameter is deprecated. The standard parameter *Scan3dBaseline* should be used instead.

5.1.5 Proprietary Chunk Parameters

The producer offers some proprietary parameter that are delivered with every image when chunk data is enabled.

- *ChunkRcOut1Reduction*: Reports the brighness reduction that the auto exposure in the mode *AdaptiveOut1* chose.
- *ChunkRcBrightness*: Current brightness of the images.
- *ChunkRcAutoExposureAdapting*: True, if the auto-exposure is currently adapting the exposure time. An application may wait until this flag becomes false to avoid images with under- or over-exposure.
- *ChunkRcReducedDepthRange*: True, if the depth range is smaller than defined with *DepthMinDepth* and *DepthMaxDepth*, due to memory limitations, defined by RC_SGM_MAXMEM.
- *ChunkRcMinDepth*, *ChunkRcMaxDepth*: Current depth range, which can be smaller then specified, due to memory limitations.
- *ChunkRcTest*: True, if the producer provides test images instead of real camera images.

5.2 Calibration Interface (rc_viscore only)

The rc_calib tool offers an easy human interface for calibrating cameras. However, all of the functionality can also be used programmatically through the GenICam nodemap.

5.2.1 Camera Calibration

The parameter *RcCalibrationAvailable* is true, if the camera is calibrated. However, an existing calibration does not necessaritly mean that the calibration is (still) accurate. For testing the calibration or calibrating the camera, (annotated) calibration images must be streamed. The calibration mode is turned on as long as the annotated calibration images are streamed. Streaming of calibration images is only possible after setting *AcquisitionMultiPartMode* to *SingleComponent*. Additionally, chunk data should be enabled to receive valuable information about the calibration process. It is also advisable to perform calibration with either auto exposure or manual exposure, but never in HDR mode. All of this is done by:

- *AcquisitionMultiPartMode=SingleComponent*
- *ComponentSelector=Calibration*
- *ComponentEnable=True*
- *ComponentSelector=Intensity*
- *ComponentEnable=False* (because the intensity component is on by default)
- *ChunkModeActive=True*
- *ExposureAuto=Continuous*

The producer supports adjusting the focus, verifying an existing calibration or calibrating a camera. For verifying or performing calibration, the size of the calibration grid must be specified by setting the parameters *RcCalibrationGridWidth* and *RcCalibrationGridHeight* to the grid size in meter. The parameter *RcCalibrationState* controls what should be done.

- *Focus*: This state is no by default and can always be set. This shows the focus helper bars in the annotated calibration images. The bars show the amount of blur of the calibration grid. A user would place the calibration grid in the middle of the working distance and change the focus of the lens such that the bars reach a minimum.
- *Verify*: This state is only possible if the camera is calibrated and if the calibration grid size is given. It is used to verify the calibration by holding a calibration grid in front of the camera such that it is fully visible in both camera images. This will be acknowledged with a thick green frame around the grid. The current reprojection error is given with each image in the parameter *ChunkRcCalibrationError*. The error is higher when the grid is held closer to the camera. An error of below 0.2-0.3 pixel indicates that the calibration is good.
- *Full*: This state is only possible if the calibration grid size is given. The user is guided by the annotated calibration images through different poses in which the grid must be held in front of the left, right or both cameras. The next pose in which the grid should be held is given as annotation in the calibration image and also in the parameter *ChunkRcCalibrationNextPose*. In an automatic calibration setup with a robot, this value could be used to detect that the pose was accepted and to move the grid into the next pose.
- *Stereo*: This state is only possible if the camera is calibrated and if the calibration grid size is given. Only the relation of the cameras to each other is calibrated by holding the grid close to the camera so that it is still fully visible in both camera images and then further away.
- *Calculate*: This state is only possible from the states *Full* or *Stereo*, if all required images have been collected. This is indicated by *ChunkRcCalibrationDataCollectedFlag=True* and *ChunkRcCalibrationNextPose=Done*. After setting this state, the parameter *ChunkRcCalibrationError* will contain the calibration error. A value of 0.2 pixels indicates a good calibration.
- *Save*: Setting this state is only possible if the current state is *Calculate*. It stores the calibration persistently on the camera and then automatically changes the state.

The annotated calibration images can be horizontally mirrored by setting *RcCalibrationFlip=True*. Some users find it more easy to work with the mirrored feedback.

For setting up calibration with a robot, the following procedure is recommended:

- Ensure that the camera can always be mounted in exactly the same way.
- Mount the calibration grid very stiff on the robot. It must not move or rotate when the robot moves.
- Use the rc_calib tool for teaching the poses and set it to *Monocalibration* (which sets *RcCalibrationState=Full*).
- Disable *Auto Accept* to avoid that the program accepts the grid immediately.
- Move the robot to the requested pose such that the marked corners of the grid are in the middle of the green rectangles. The size of the green rectangles can be increased with *RcCalibrationSensitiveAreaScale* if more tolerance is required, although this is not recommended as it relaxes the pose contraint and can lead to less accurate calibrations.
- Store the robot pose together with the name of the pose in the robot program. See "Next pose" in the status bar of rc_calib. This is the value of the parameter *ChunkRcCalibrationNextPose*.
- Enable *Auto Accept* to accept the current grid and go to next pose. Then disable *Auto Accept* again.
- Proceed with moving the grid to the next pose until all poses are stored.

For performing calibration with a robot:

- Set *RcCalibrationAutoAccept=True*
- Set *RcCalibrationState=Full*
- Move the robot into the pose that is suggested by *ChunkRcCalibrationNextPose*. The grid will be detected automatically and the next pose suggested.
- Continue until *ChunkRcCalibrationNextPose* is *Done*
- Set *RcCalibrationState=Calculate* and wait until *ChunkRcCalibrationError* of the next images is larger than 0.
- Check if the error is ok. A value below 0.2 pixel is typically considered to be ok.
- Set *RcCalibrationState=Save*, poll *RcCalibrationState* and wait as long as it returns *Save*.
- Calibration is now safely stored and the connection to the camera can be closed.

5.3 Computing a Point Cloud

The producer provides depth data encoded in a disparity image. It can be retrieved by setting *ComponentSelector=Disparity* and *ComponentEnable=True*. The disparity image contains 16 bit unsigned integer values. These values must be multiplied by the scale value given in the GenICam feature *Scan3dCoordinateScale* to get the disparity values d in pixels. To compute the 3D object coordinates from the disparity values, the focal length and the baseline as well as the principle point are required. These parameters are provided as GenICam features *Scan3dFocalLength*, *Scan3dBaseline*, *Scan3dPrincipalPointU* and *Scan3dPrincipalPointV*. The focal length and principal point depend on the image resolution of the selected component.

Therefore, it is preferable to enable chunk data with the parameter *ChunkModeActive* and to use the chunk parameters *ChunkScan3dCoordinateScale*, *ChunkScan3dFocalLength*, *ChunkScan3dBaseline*, *ChunkScan3dPrincipalPointU* and *ChunkScan3dPrincipalPointV* that are delivered with every image, because their values already fit to the image resolution of the corresponding image.

Knowing these values, the pixel coordinates and the disparities can be transformed into 3D object coordinates in the camera coordinate frame. Assuming that $d16_{ik}$ is the 16 bit disparity value at image column i and image row k of a disparity image, the float disparity in pixels d_{ik} is given by

$$d_{ik} = d16_{ik} \cdot \mathrm{Scan3dCoordinateScale}$$

The 3D reconstruction in meters can be written with the GenICam parameters as:

$$P_x = (i + 0.5 - \mathrm{Scan3dPrincipalPointU}) \, \frac{\mathrm{Scan3dBaseline}}{d_{ik}},$$

$$P_y = (k + 0.5 - \mathrm{Scan3dPrincipalPointV}) \, \frac{\mathrm{Scan3dBaseline}}{d_{ik}},$$

$$P_z = \mathrm{Scan3dFocalLength} \frac{\mathrm{Scan3dBaseline}}{d_{ik}}.$$

The confidence image contains 8 bit unsigned integer values. These values have to be divided by 255 to get the confidence as value between 0 an 1.

The error image contains 8 bit unsigned integer values. The error $e8_{ik}$ must be multiplied by the scale value given in the GenICam feature *Scan3dCoordinateScale* to get the disparity-error values d_{eps} in pixels. The depth error z_{eps} in meters can be computed with GenICam parameters as

$$z_{eps} = \frac{e8_{ik} \cdot \mathrm{Scan3dCoordinateScale} \cdot \mathrm{Scan3dFocalLength} \cdot \mathrm{Scan3dBaseline}}{(d_{ik})^2}.$$

For texturing the point cloud, the image of the left camera can be used (i.e. component *Intensity*), which is perfectly registered to the disparity image. In general, the disparity image is smaller than the camera image, depending on the parameter *DepthQuality*. Depth quality *High* means that the width and height of the disparity image is two time smaller than camera image. With *Medium*, it is factor four and with *Low* it is factor six. It is easiest to just downscale the camera image by this integer factor and use the intensity or color of the corresponding pixel for the reconstructed 3D point.

6 Using SGM® Producer in Applications

The producer is a software library that implements the GenTL 1.6 interface. A GenICam compatible application is required for using the producer.

6.1 Halcon

Halcon fully supports the rc_visard, rc_viscore and the off-board producer. Halcon can only find the producer if the directory of the producer is specified in the environment variable GENICAM_GENTL64_PATH (see installation above). The GenICamTL package that is provided by MVTec must be installed additionally to Halcon.

For using the SGM® Producer in hdevelop, GenICamTL should be specified as the first parameter in the open_framegrabber() call. The device is identified either by the user-defined name or the device ID.

A good starting point to work with Halcon is the rc_visard example program (i.e. gigevision2_roboception_rcvisard_objectmodel3d.hdev) that is delivered with the Halcon GenICamTL package. The following modifications are needed for using it with the SGM® Producer:

- Specify 'GenICamTL' instead of 'GigEVision2' for the open_framegrabber command (the environment variable GENICAM_GENTL64_PATH must be set as described above).
- Remove or uncomment the line that is setting the parameter 'GevStreamDeliverIncompleteBlocks'. This parameter is not available in the SGM® Producer as it never publishes incomplete buffers.
- For color cameras, set the pixel format to 'RGB8'. The SGM® Producer does not support the pixel format 'YCbCr411_8'.

6.2 C++ API

C++ programmers can use the rc_genicam_api convenience layer from Roboception that can be downloaded from https://github.com/roboception/rc_genicam_api. The package offers a C++ interface to GenICam and the producer. It includes a standard GigE Vision producer for communication with the rc_visard. The standard producer can be replaced by the SGM® Producer by setting the environment variable GENICAM_GENTL64_PATH to the directory of the SGM® Producer (see installation above).

The package contains tools for getting and setting parameters and receiving images. The tools also serve as examples for demonstrating the use of the API.

6.3 OpenCV

A tutorial for getting started with the rc_visard and OpenCV is provided at https://tutorials.roboception.de/rc_visard_general/opencv_example.html. The OpenCV example is based on the rc_genicam_api (see C++ API above). The SGM® Producer is used by setting the environment variable GENICAM_GENTL64_PATH to the directory of the SGM® Producer (see installation above).

6.4 ROS

ROS drivers are available for ROS 1 and ROS 2 on the ROS build farm. If ROS is already installed, the driver can be installed with:

```
sudo apt install ros-${ROS_DISTRO}-rc-genicam-driver
```

The SGM® Producer is used by setting the environment variable GENICAM_GENTL64_PATH to the directory of the SGM® Producer as discussed in the installation section above.

More information about the drivers is given in the readmes of the ROS 1 and ROS 2 drivers, i.e. https://github.com/roboception/rc_genicam_driver_ros and https://github.com/roboception/rc_genicam_driver_ros2.

7 Troubleshooting

Note: The SGM® Producer will never forward disparity, confidence or error images that are processed on-board the rc_visard to avoid confusion. A normal GigE Vision producer should be used for enabling on-board processing.

If the application does not find the producer:

- Check and correct the GENICAM_GENTL64_PATH variable. Restart the application after changing the variable.

If an rc_visard device cannot be discovered:

- Check with the rcdiscover tool https://github.com/roboception/rcdiscover that the device is listed and marked as 'reachable'. You may also verify that the Web GUI can be reached by double-clicking on the device in the rcdiscover window.
- Under Windows, the firewall or an anti-virus package may block the communication to network devices. Try to temporarily disable the firewall or anti-virus package to see if this is the cause.

If an rc_visard is discovered, but cannot be opened:

- Ensure that the firmware version of the rc_visard at least 21.04.0. This can be checked on the system page of the rc_visard Web GUI. If the firmware version is too low, an update can be downloaded from https://roboception.com/en/download/ and uploaded via the Web GUI of the rc_visard.

If an rc_viscore device cannot be discovered or cannot be opened:

Note: the rc_check tool is not included in the Basler branded version of the SGM® Producer.

- Check with the included rc_check tool (see tools for the rc_viscore above) that the device is listed and marked as 'ok'. Follow the hints that are provided by this tool.
- If rc_check cannot find the rc_viscore or any individual cameras of it, then the IP configuration might be wrong. By default, the cameras try to find a DHCP server. If they cannot find one, e.g. because the cameras are connected directly to a computer as recommended, they assign themselves a link local address. In this case, the host port must be configured accordingly, which is the default under Windows, but must be done manually under Linux. Alternatively, a temporary IP address can be manually assigned with the rcdiscover tool https://github.com/roboception/rcdiscover. After starting the tool, uncheck "Only rc_...devices" to see the individual cameras. The entry "Reachable" indicates if the IP configuration is correct. After setting a temporary IP address and making the cameras reachable, the IP configuration can be made permanent with the command line tool gc_config from the rc_genicam_api package https://github.com/roboception/rc_genicam_api.

If the camera trigger does not work as expected:

- The rc_viscore offers software and hardware triggering (see *TriggerMode* and *TriggerSource* features), which grab a single stereo image. Some modes like *ExposureAuto=HDR*, *DepthDoubleShot=1* or *DepthStaticScene=1* require more than one image for operation, thus multiple trigger are required for producing a result. Furthermore, the auto exposure cannot adapt without a continuous stream of images, thus if *DepthExposureAdaptTimeout* should be set to 0, which is the default, otherwise stereo matching may just wait for the next images. For using these modes, it is recommended to let the camera grab continuously and trigger acquisition of depth images instead, i.e. set *TriggerMode=Off*, *DepthAcquisitionMode=SingleFrameOut1* and call *DepthAcquisitionTrigger* for triggering a depth image.
- Enable printing or storing log information and check for messages about incomplete buffers. See next issue in this case.

If rc_visard or rc_viscore images cannot be received or if received buffers are incomplete:

- Try to connect the device directly (i.e. without switch) to the computer. In case of using a switch with an rc_viscore, the switch should be connected with at least 2.5 Gbit to the host computer. If the switch only has a bandwith of 1 Gbit to the host computer, the throughput of the rc_viscore

must be limited by setting the GenICam parameters *DeviceLinkThroughputLimitMode=On* and *DeviceLinkThroughputLimit=110000000*. This limits the throughput to just under 1 Gbit.

- In case of an rc_visard, check on the system page of the Web GUI that the link speed is 1000 Mbit/s. Replace the network cable or check the network port settings on your host computer if the link speed is lower than 1000 Mbit/s.
- For avoiding incomplete buffers, the MTU should be set to 9000 if possible. Under Windows, this is often called jumbo frames. Increasing the receive buffer may also help. More information about network setup and optimization is provided in a tutorial at https://tutorials.roboception.de/rc_visard_general/network_setup.html.
- Under Windows, the firewall or an anti-virus package may block the communication to the device. Try to temporarily disable the firewall or anti-virus package to see if this is the cause.

If camera images can be received, but disparity, confidence and error images are not delivered:

- The most likely cause for never receiving disparity images is insufficient GPU memory. Try to reduce the depth quality with the GenICam parameter 'DepthQuality'. Additionally or alternatively, the disparity range can be reduced by the GenICam parameters 'DepthMinDepth' and 'DepthMaxDepth'. It is also possible to specify the maximum amount of memory with the environment variable RC_SGM_MAXMEM (see above). In this case, the producer increases the minimum depth as necessary in order to not exceed the given amount of memory.

If disparity, confidence and/or error images are delivered but the frequency is lower than expected:

- Check the display name of the producer (i.e. GenICam system) in your application. You may also install rc_genicam_api from https://github.com/roboception/rc_genicam_api and call 'gc_info -l' in the command line. It shows the display name in the first few lines of the output. Ensure that the display name reports your GPU. If not, then stereo matching is performed on the CPU, which is much slower than running it on the GPU. If an Nvidia GPU is installed on the computer, try updating the graphics card driver.
- Reducing the depth quality with the GenICam parameter 'DepthQuality' and/or reducing the disparity range with the GenICam parameters 'DepthMinDepth' and 'DepthMaxDepth' will increase the framerate.

Why are parameters reset when reconnecting to a camera?

- The SGM® Producer provides a virtual device that runs on the host computer and vanishes when a connection is closed. In case of rc_visard, some parameters (e.g. for controlling exposure, gain, etc) are loaded from and stored on the camera, while others (e.g. for depth image computation) are only kept in memory of the host computer. In case of an rc_viscore, all parameters start with default values when the connection is opened.

Still having problems?

- Create a log file by setting the environment variable RC_SGM_PRODUCER_LEVEL to 5 and RC_SGM_PRODUCER_LOG to the full path and name of the logfile to be generated. The log file is a normal text file that can be checked for errors or sent to Roboception support.
- Contact Roboception support via support@roboception.com. The support team may ask you for a log file (see above).

8 Appendix: Example Nodemap of an rc_viscore

```
Category: DeviceControl (RO)
  Enumeration: DeviceType (RO) [Transmitter Receiver Transceiver Peripheral]: Transmitter
  Enumeration: DeviceScanType (RO) [Areascan Linescan Areascan3D Linescan3D]: Areascan3D
  String: DeviceVendorName (RO): Roboception GmbH
  String: DeviceModelName (RO): rc_viscore 210m-16-12M-H1
  String: DeviceManufacturerInfo (RO): Roboception GmbH
  String: DeviceVersion (RO): 23.04.0
  String: DeviceFirmwareVersion (RO): 23.04.0
  String: DeviceSerialNumber (RO): GX045119
  String: DeviceID (RO): GX045119
  String: DeviceUserID (RW): GX045119
  Integer: DeviceSFNCVersionMajor (RO) [-9223372036854775808, 9223372036854775807]: 2
  Integer: DeviceSFNCVersionMinor (RO) [-9223372036854775808, 9223372036854775807]: 5
  Integer: DeviceSFNCVersionSubMinor (RO) [-9223372036854775808, 9223372036854775807]: 0
  String: DeviceManifestPrimaryURL (RO): local:rc_sgm_producer_v1.zip;10000000;5dee
  String: DeviceManifestSecondaryURL (RO):
  Enumeration: DeviceTLType (RO) [GigEVision CameraLink CameraLinkHS CoaXPress USB3Vision
    Custom]: GigEVision
  Integer: DeviceTLVersionMajor (RO) [0, 65535]: 2
  Integer: DeviceTLVersionMinor (RO) [0, 65535]: 1
  Integer: DeviceTLVersionSubMinor (RO) [-9223372036854775808, 9223372036854775807]: 0
  Integer: DeviceLinkSelector (RO) [-9223372036854775808, 9223372036854775807]: 0
  Integer: DeviceLinkSpeed (RO) [0, 4294967295]: 250000000 Bps
  Enumeration: DeviceLinkThroughputLimitMode (RW) [Off On]: Off
  Integer: DeviceLinkThroughputLimit (NA)
  Integer: DeviceLinkConnectionCount (RO) [0, 4294967295]: 1
  Integer: DeviceStreamChannelCount (RO) [0, 4294967295]: 1
  Integer: DeviceStreamChannelSelector (RW) [0, 0]: 0
  Enumeration: DeviceStreamChannelType (RO) [Transmitter Receiver]: Transmitter
  Enumeration: DeviceStreamChannelEndianness (RO) [Big Little]: Little
  Integer: DeviceEventChannelCount (RO) [0, 4294967295]: 0
  Enumeration: DeviceCharacterSet (RO) [UTF8 ASCII]: UTF8
  Enumeration: DeviceRegistersEndianness (RO) [Big Little]: Big
  Command: TimestampLatch (WO)
  Integer: TimestampLatchValue (RO) [0, 9223372036854775807]: 0 ns
  Boolean: RcSystemReady (RO): 1
  Integer: RcError (RO) [0, 4294967295]: 0
Category: ImageFormatControl (RO)
  Integer: WidthMax (RO) [0, 4294967295]: 4112
  Integer: HeightMax (RO) [0, 4294967295]: 6016
  Integer: Width (RO) [4, 4112]: 4112
  Integer: Height (RO) [4, 6016]: 3008
  Enumeration: PixelFormat (RW) [Mono8]: Mono8
  Enumeration: ComponentSelector (RW) [Intensity IntensityCombined Disparity Confidence Error
    RawCombined Calibration]: Intensity
  Boolean: ComponentEnable (RW): 1
  Integer: ComponentIDValue (RO) [-9223372036854775808, 9223372036854775807]: 1
  Integer: DecimationHorizontal (RO) [-9223372036854775808, 9223372036854775807]: 1
  Integer: DecimationVertical (RO) [-9223372036854775808, 9223372036854775807]: 1
Category: AcquisitionControl (RO)
  Enumeration: AcquisitionMode (RW) [SingleFrame Continuous MultiFrame]: Continuous
  Command: AcquisitionStart (RW)
  Command: AcquisitionStop (RW)
  Integer: AcquisitionFrameCount (RW) [1, 4294967295]: 0
  Float: AcquisitionFrameRate (RW) [1, 9]: 9 Hz
  Boolean: AcquisitionFrameRateEnable (RO): 1
  Float: ExposureTime (RW) [20, 2e+07]: 20000 us
  Enumeration: ExposureAuto (RW) [Off Continuous AdaptiveOut1 Out1High HDR]: Continuous
  Float: ExposureTimeAutoMax (RW) [20, 2e+07]: 111111 us
  Integer: ExposureRegionWidth (RW) [0, 4112]: 0
```

```
  Integer: ExposureRegionHeight (RW) [0, 3008]: 0
  Integer: ExposureRegionOffsetX (RW) [0, 4112]: 0
  Integer: ExposureRegionOffsetY (RW) [0, 3008]: 0
  Enumeration: ExposureMode (RW) [Timed]: Timed
  Enumeration: TriggerSelector (RW) [FrameStart]: FrameStart
  Enumeration: TriggerSource (RW) [Software In1]: In1
  Enumeration: TriggerActivation (RW) [RisingEdge FallingEdge AnyEdge]: RisingEdge
  Float: TriggerDelay (RW) [0, 4.29497e+09]: 0 us
  Enumeration: TriggerMode (RW) [Off On]: Off
  Command: TriggerSoftware (RW)
  Enumeration: AcquisitionAlternateFilter (RW) [Off OnlyHigh OnlyLow]: Off
  Enumeration: AcquisitionMultiPartMode (RW) [SingleComponent SynchronizedComponents]:
    SingleComponent
  Float: RcExposureAutoAverageMax (RW) [0, 1]: 0.75
  Float: RcExposureAutoAverageMin (RW) [0, 1]: 0.25
Category: AnalogControl (RO)
  Enumeration: GainSelector (RO) [All]: All
  Float: Gain (RW) [0, 48]: 0 dB
  Float: Gamma (RW) [0.01, 10]: 0.5
Category: DigitalIOControl (RO)
  Enumeration: LineSelector (RW) [Out1 Out2 Out3 In1 In2 In3 In4]: Out1
  Enumeration: LineMode (RO) [Output]: Output
  Float: RcLineRatio (RW) [0, 1]: 1
  Boolean: LineInverter (RW): 0
  Boolean: LineStatus (RO): 0
  Integer: LineStatusAll (RO) [0x0, 0xff]: 0x0
  Enumeration: LineSource (RW) [Low High ExposureActive ExposureAlternateActive]: Low
Category: RcCalibration (RO)
  Float: RcCalibrationGridWidth (RW) [0, 5]: 0 m
  Float: RcCalibrationGridHeight (RW) [0, 5]: 0 m
  Float: RcCalibrationSensitiveAreaScale (RW) [0.5, 10]: 1
  Enumeration: RcCalibrationState (RW) [Focus]: Focus
  Boolean: RcCalibrationAvailable (RO): 1
Category: FileAccessControl (RO)
  Enumeration: FileSelector (RW) [Calibration]: Calibration
  Enumeration: FileOperationSelector (RW) [Open Close Read Write Delete]: Open
  Enumeration: FileOpenMode (RW) [Read Write ReadWrite]: Read
  Command: FileOperationExecute (RW)
  Enumeration: FileOperationStatus (RO) [Success Failure Invalid]: Success
  Integer: FileOperationResult (RO) [0, 4294967295]: 0 B
  Integer: FileSize (RO) [0, 4294967295]: 0 B
  Integer: FileAccessOffset (RW) [0, 4294967295]: 0 B
  Integer: FileAccessLength (RW) [0, 4294967295]: 0 B
  Register: FileAccessBuffer (RW)
Category: Scan3dControl (RO)
  Enumeration: Scan3dDistanceUnit (RO) [Pixel]: Pixel
  Enumeration: Scan3dOutputMode (RO) [DisparityC]: DisparityC
  Float: Scan3dCoordinateScale (RO) [-1.79769e+308, 1.79769e+308]: 0.0625
  Float: Scan3dCoordinateOffset (RO) [-1.79769e+308, 1.79769e+308]: 0
  Boolean: Scan3dInvalidDataFlag (RO): 1
  Float: Scan3dInvalidDataValue (RO) [-1.79769e+308, 1.79769e+308]: 0
  Float: Scan3dFocalLength (RO) [-1.79769e+308, 1.79769e+308]: 4658.59 Pixel
  Float: Scan3dBaseline (RO) [-3.40282e+38, 3.40282e+38]: 0.209542 m
  Float: Scan3dPrincipalPointU (RO) [-1.79769e+308, 1.79769e+308]: 2056 Pixel
  Float: Scan3dPrincipalPointV (RO) [-1.79769e+308, 1.79769e+308]: 1504 Pixel
  Float: FocalLengthFactor (RO) [-3.40282e+38, 3.40282e+38]: 1.13293
  Float: Baseline (RO) [-3.40282e+38, 3.40282e+38]: 0.209542 m
Category: DepthControl (RO)
  Enumeration: DepthQuality (RW) [Low Medium High Full]: High
  Integer: DepthFill (RW) [0, 4]: 3 pixel
  Integer: DepthSeg (RW) [0, 4000]: 200 pixel
  Float: DepthMinConf (RW) [0, 1]: 0.5
  Float: DepthMinDepth (RW) [0.1, 100]: 0.1 m
```

```
  Float: DepthMaxDepth (RW) [0.1, 100]: 100 m
  Float: DepthMaxDepthErr (RW) [0.01, 100]: 100 m
  Enumeration: DepthAcquisitionMode (RW) [SingleFrame SingleFrameOut1 Continuous]: Continuous
  Command: DepthAcquisitionTrigger (RW)
  Boolean: DepthSmooth (RW): 1
  Boolean: DepthStaticScene (RW): 0
  Boolean: DepthDoubleShot (RW): 0
  Float: DepthExposureAdaptTimeout (RW) [0, 2]: 0 s
Category: ChunkDataControl (RO)
  Boolean: ChunkModeActive (RW): 0
  Enumeration: ChunkSelector (RW) [Image OffsetX OffsetY Width Height PixelFormat Timestamp
    LineStatusAll FrameID ComponentID ComponentIDValue Scan3dDistanceUnit Scan3dOutputMode
    Scan3dCoordinateScale Scan3dCoordinateOffset Scan3dInvalidDataFlag Scan3dInvalidDataValue
    Scan3dFocalLength Scan3dBaseline Scan3dPrincipalPointU Scan3dPrincipalPointV Components
    PartIndex DecimationHorizontal DecimationVertical LineSource]: Image
  Boolean: ChunkEnable (RO): 0
  Integer: ChunkFrameID (NA)
  Integer: ChunkTimestamp (NA)
  Integer: ChunkLineStatusAll (NA)
  Float: ChunkExposureTime (NA)
  Float: ChunkGain (NA)
  Integer: ChunkWidth (NA)
  Integer: ChunkHeight (NA)
  Integer: ChunkOffsetX (NA)
  Integer: ChunkOffsetY (NA)
  Enumeration: ChunkPixelFormat (NA) [Mono8 Mono16 RGB8 Confidence8 Coord3D_C16 Error8]:
  Enumeration: ChunkComponentID (NA) [Intensity IntensityCombined Disparity Confidence Error]:
  Integer: ChunkComponentIDValue (NA)
  Enumeration: ChunkComponentSelector (NA) []:
  Enumeration: ChunkScan3dDistanceUnit (NA) [Pixel]:
  Enumeration: ChunkScan3dOutputMode (NA) [DisparityC]:
  Enumeration: ChunkScan3dCoordinateSelector (RW) [CoordinateC]: CoordinateC
  Float: ChunkScan3dCoordinateScale (NA)
  Float: ChunkScan3dCoordinateOffset (NA)
  Boolean: ChunkScan3dInvalidDataFlag (NA)
  Float: ChunkScan3dInvalidDataValue (NA)
  Float: ChunkScan3dFocalLength (NA)
  Float: ChunkScan3dBaseline (NA)
  Float: ChunkScan3dPrincipalPointU (NA)
  Float: ChunkScan3dPrincipalPointV (NA)
  Integer: ChunkComponents (NA)
  Integer: ChunkPartIndex (NA)
  Integer: ChunkDecimationHorizontal (NA)
  Integer: ChunkDecimationVertical (NA)
  Enumeration: ChunkLineSelector (RW) [Out1 Out2 Out3 In1 In2 In3 In4]: Out1
  Enumeration: ChunkLineSource (NA) [Low High ExposureActive ExposureAlternateActive]:
  Boolean: ChunkLineStatus (NA)
  Float: ChunkRcLineRatio (NA)
  Float: ChunkRcNoise (NA)
  Float: ChunkRcOut1Reduction (NA)
  Boolean: ChunkRcTest (NA)
  Float: ChunkRcBrightness (NA)
  Boolean: ChunkRcAutoExposureAdapting (NA)
  Boolean: ChunkRcReducedDepthRange (NA)
  Float: ChunkRcMinDepth (NA)
  Float: ChunkRcMaxDepth (NA)
  Boolean: ChunkRcCalibrationDataCollectedFlag (NA)
  Float: ChunkRcCalibrationError (NA)
Category: TestControl (RO)
  Enumeration: TestPayloadFormatMode (RW) [Off MultiPart]: Off
Category: TransportLayerControl (RO)
  Integer: TLParamsLocked (RW) [-9223372036854775808, 9223372036854775807]: 0
  Integer: PayloadSize (RO) [0, 4294967295]: 173165060 B
```

```
Category: PtpControl (RO)
  Boolean: PtpEnable (RW): 0
  Command: PtpDataSetLatch (WO)
  Enumeration: PtpStatus (RO) [Initializing Faulty Disabled Listening PreMaster Master
    Passive Uncalibrated Slave]: Disabled
  Integer: PtpOffsetFromMaster (RO) [-2147483648, 2147483647]: 0 ns
Category: GigEVision (RO)
  Integer: GevStreamChannelSelector (RW) [0, 0]: 0
  Integer: GevSCPSPacketSize (RO) [0, 4294967295]: 7500
  Integer: GevSCPD (RW) [0, 1000]: 0 ns
  Integer: GevInterfaceSelector (RW) [0, 1]: 0
  Integer: GevMACAddress (RO) [0:0:0:0:0:0, ff:ff:ff:ff:ff:ff]: 0:c:8d:60:b0:3f
  Boolean: GevCurrentIPConfigurationDHCP (RW): 1
  Boolean: GevCurrentIPConfigurationPersistentIP (RW): 0
  Boolean: GevCurrentIPConfigurationLLA (RO): 1
  Integer: GevCurrentIPAddress (RO) [0.0.0.0, 255.255.255.255]: 172.23.42.4
  Integer: GevCurrentSubnetMask (RO) [0.0.0.0, 255.255.255.255]: 255.255.255.240
  Integer: GevCurrentDefaultGateway (RO) [0.0.0.0, 255.255.255.255]: 172.23.42.1
  Integer: GevPersistentIPAddress (RW) [0.0.0.0, 255.255.255.255]: 0.0.0.0
  Integer: GevPersistentSubnetMask (RW) [0.0.0.0, 255.255.255.255]: 0.0.0.0
  Integer: GevPersistentDefaultGateway (RW) [0.0.0.0, 255.255.255.255]: 0.0.0.0
```