ZED

SDK

v0.9.3

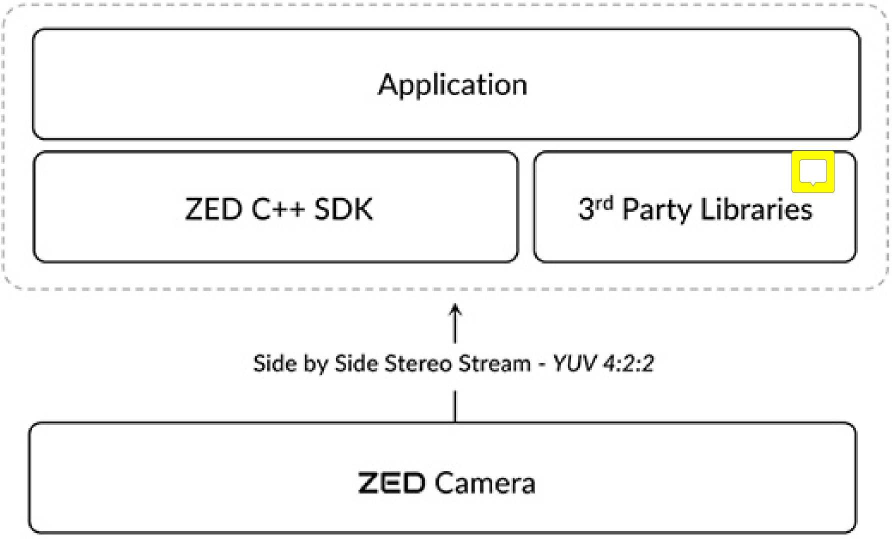
Contents

ZED SDK

Getting Started

SDK Structure

The ZED Stereo Camera is a lightweight depth sensor based on passive stereovision. It outputs a high resolution side-by-side video on USB 3.0 that contains two synchronized left and right video streams. Using the ZED SDK, the graphics processing unit (GPU) from a host machine computes the depth map from the side-by-side video, allowing developers to create depth-enabled applications. Because the ZED SDK massively uses CUDA, a NVIDIA GPU is required to be able to use the SDK.



Content Structure

The SDK is installed by default:

- on **Windows** at C:/Program Files (x86)/ZED SDK
- on **Linux** at /usr/local/zed

The SDK falls into the following directories:

- **bin/** (Windows only) contains the windows DLLs required to use the SDK
- **dependencies/** (Windows only) contains samples and SDK (OpenCV) dependencies
- **doc/** contains documentations, licence and tutorials
  - **API/** contains the API documentation on Web format
  - **license/** contains the licenses of the SDK and its third-parties’
- **include/** contains C/C++ header files
- **lib/** contains the libraries files required to link with the SDK (".lib" on Windows and ".so" on Linux)
- **sample/** contains sample programs using the SDK
- **settings/** (Linux only) (on Windows at AppData\Roaming\Stereolabs) contains ZED camera parameters
- **tools/** contains binary executables included in the SDK that configure and manage the ZED camera

System Requirements

This section describes the minimum system requirements to use the ZED SDK, we recommend to use a high performance hardware to ensure reliable computation time.

Operating Systems the ZED SDK currently supports Windows 7, 8.1 and 10 (64 bits) as well as Linux Ubuntu 14.04 (64 bits) and Linux4Tegra (Jetson OS).

Minimum System Requirements

In order to run the ZED SDK you need to have **at least** the following configuration:

- Dual-core 2,3GHz or faster processor
- 4 GB RAM or more
- NVIDIA GPU with Compute Capabilities $>$ 2.0
- CUDA 7.5
- USB 3.0 port
- Windows 7, Windows 8.1, Windows 10 (64 bits), Ubuntu 14.04 (64 bits), L4T21.3/4 (Jetson)

In order to take full advantage of the ZED SDK you need a modern and powerful NVIDIA GPU. We currently support CUDA 7.5 and all the compute capabilities from 2.0 to 5.2. We will provide a newer version of the SDK as soon as possible when the next version of CUDA is released.

Recommended configuration:

- Dual-core 2,5GHz processor or faster
- 4 GB of RAM
- NVidia GTX 560
- CUDA 7.5
- USB 3.0 port
- Windows 7, Windows 8.1, Windows 10 (64 bits), Ubuntu 14.04 (64 bits), L4T21.3/4 (Jetson)

OpenCV dependencies:

ZED SDK is using some modules of OpenCV. The current version for the ZED SDK is v2.4.9.

Include, libs and dlls are provided as dependencies in the ZED SDK setup for Windows. Under Linux, you need to download OpenCV source code and compile it with default dependencies. A very useful tutorial can be found here

Recording configuration:

Additionally, if you plan to record high resolution footage –i.e 3840x1080 @ 30fps – we recommend having a fast SSD with transfer speeds easily reaching 250 MB/s. If you are recording on your OS disk, we strongly recommend at least 256GB capacity. The videos are recorded with no compression to ensure the best quality in playback. An available USB 3.0 port is mandatory.

Notes:

If you don't have an NVIDIA GPU – i.e: Intel HD Graphics or AMD chip - you can still use the tools such as the ZED Explorer to view and record side by side 3D video but you will not be able to visualize depth or use any of the samples. If you are looking for a portable workstation, the Nvidia GT 740M inside a PC Laptop provides minimal graphics power to run the samples.

Installation

Step 1: Set up your environment

Before you start developing applications with the ZED, make sure the following assets are installed.

1. Latest USB 3.0 drivers.
2. Latest NVIDIA Display Driver.
3. NVIDIA CUDA 7.5 Toolkit.
4. Windows Only: Visual Studio 2013 Redistributable Packages (available on the ZED USB Drive)

Step 2: Run the ZED installer

1. Plug your ZED into an USB 3.0 port and make sure the computer has an internet access.

   - Internet access is required since the installer will fetch your camera calibration file from our online database.
   - If you don't have an internet access at all you can still use the ZED SDK but depth measurement will be less accurate.

2. Launch the installer ZED_SDK_WinSetup_vX.Y.exe for windows or ZED_SDK_Linux_x86_64_vX.Y.-run for Ubuntu which you will find in the ZED USB Drive and follow the instructions.

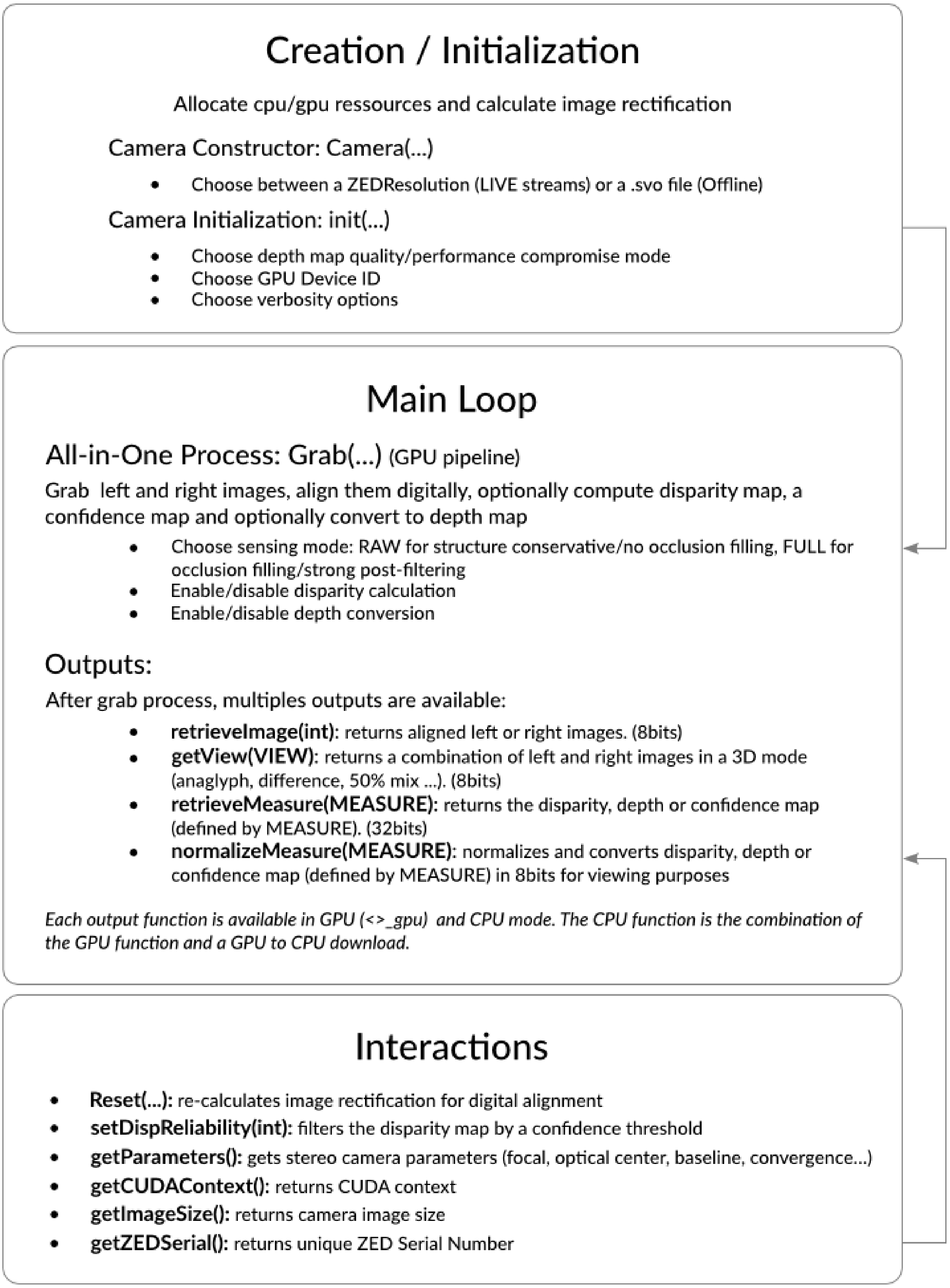
3. Finish the setup procedure, unplug your ZED and restart your computer.

IMPORTANT: Make sure you unplug your ZED before restarting your computer. On some computers, we have noticed that USB 3.0 devices could prevent from booting.

Compilation Settings

We use the cross platform open-source, build system `CMake` tool to generate IDE/compiler specific projects. To compile the samples you will need to install the latest CMake and use either the command line or the GUI to configure and generate the projects. As the installer sets up the environment path for the dependencies, you should not have to change the project's settings.

SDK Overview



Creation / Initialization

Allocate cpu/gpu ressources and calculate image rectification

Camera Constructor: Camera(...)

- Choose between a ZEDResolution (LIVE streams) or a .svo file (Offline)

Camera Initialization: init(...)

- Choose depth map quality/performance compromise mode
- Choose GPU Device ID
- Choose verbosity options

Main Loop

All-in-One Process: Grab(...) (GPU pipeline)

Grab left and right images, align them digitally, optionally compute disparity map, a confidence map and optionally convert to depth map

- Choose sensing mode: RAW for structure conservative/no occlusion filling, FULL for occlusion filling/strong post-filtering
- Enable/disable disparity calculation
- Enable/disable depth conversion

Outputs:

After grab process, multiples outputs are available:

- **retrieveImage(int)**: returns aligned left or right images. (8bits)
- **getView(VIEW)**: returns a combination of left and right images in a 3D mode (anaglyph, difference, 50% mix ...). (8bits)
- **retrieveMeasure(MEASURE)**: returns the disparity, depth or confidence map (defined by MEASURE). (32bits)
- **normalizeMeasure(MEASURE)**: normalizes and converts disparity, depth or confidence map (defined by MEASURE) in 8bits for viewing purposes

Each output function is available in GPU (<>_gpu) and CPU mode. The CPU function is the combination of the GPU function and a GPU to CPU download.

Interactions

- **Reset(...):** re-calculates image rectification for digital alignment
- **setDispReliability(int):** filters the disparity map by a confidence threshold
- **getParameters():** gets stereo camera parameters (focal, optical center, baseline, convergence...)
- **getCUDAContext():** returns CUDA context
- **getImageSize():** returns camera image size
- **getZEDSerial():** returns unique ZED Serial Number

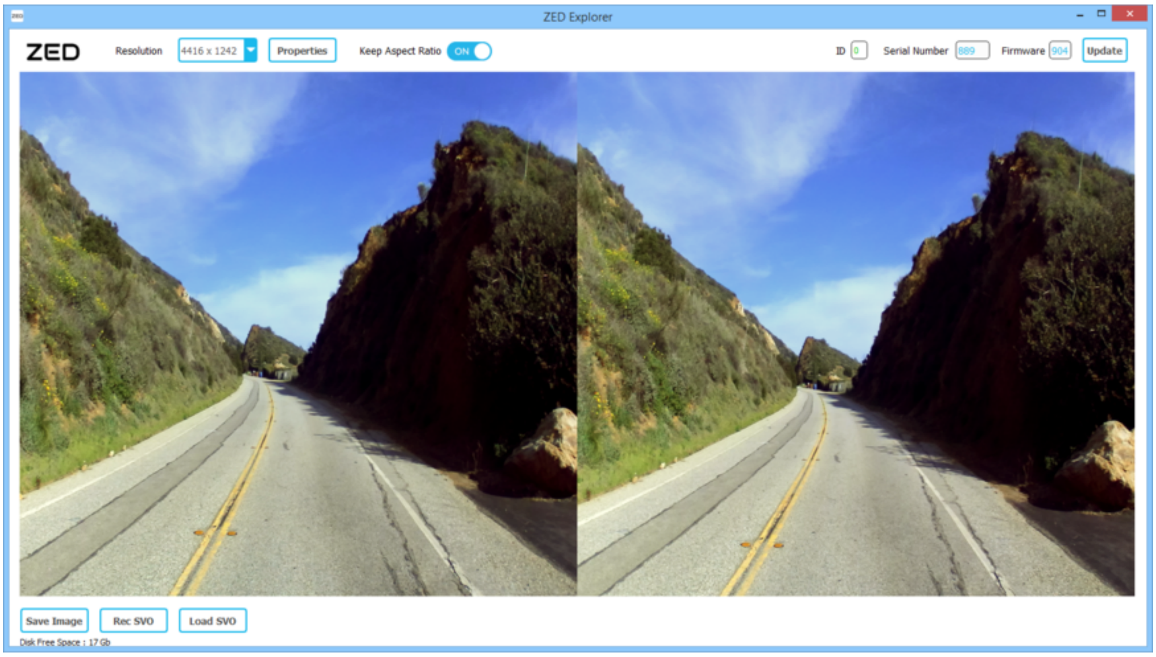
Terminology

Term	Definition
Convergence (CV)	Angle formed by the two sensor planes
SVO File Format	StereoLabs video file format containing additional ZED data in addition to the video files
Reliability Index	Global threshold
Confidence map	Score of confidence for the estimated depth data of each pixel
RAW Sensing Mode	Most accurate version of the depth map. Check Sensing Modes Section
FULL Sensing Mode	Filtered and “natural” mode. Occlusions are filled. Check Sensing Modes Section
Anaglyph View	A red/cyan display mode of the stereo 3D video
SBS (Side by Side) View	A left/right display mode of the stereo 3D video
Overlay View	Left and Right videos overlayed with 50% opacity each
Frayscale Difference View	View showing the difference map calculated between the grayscale left and right images

Using the Tools

ZED Explorer

The ZED Explorer is the main application for ZED video preview, recording ZED SDK compatible files (∗.svo) and manage ZED camera parameters. The application also lets you change video resolution, aspect ratio and camera control parameters. It also allows you to capture full resolution snapshots and video with the ZED.



Resolution 4416 x 1242
Keep Aspect Ratio ON
ID 0
Serial Number 889
Firmware 904
Disk Free Space : 17 Gb

Double click on the 3D image to toggle Full Sceen mode.

Video Modes

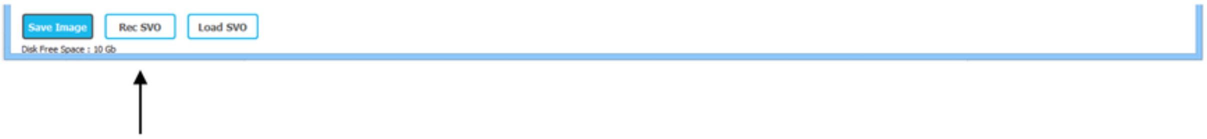
You can switch between four different video modes, each one with different framerates:

Video Mode	Output Resolution (SBS)	Framerate	Field of View
2.2K	4416x1242	15fps	Wide
1080p	3840x1080	15/30fps	Wide
720p	2560x720	15/30/60fps	Extra Wide
VGA	1280x480	15/30/60/100fps	

Please note that the field of view is linked to the video mode.

Recording Video

The ZED Explorer records feeds in a Stereolabs SVO format that is compatible with the ZED SDK functions. With SVO files, you can record stereo feeds from the ZED and compute later the depth map for these feeds using the SDK. You can convert SVO files to Side by Side or 2D+Depth AVI files using the ∗∗SVO Converter ∗∗ Sample provided in the SDK.

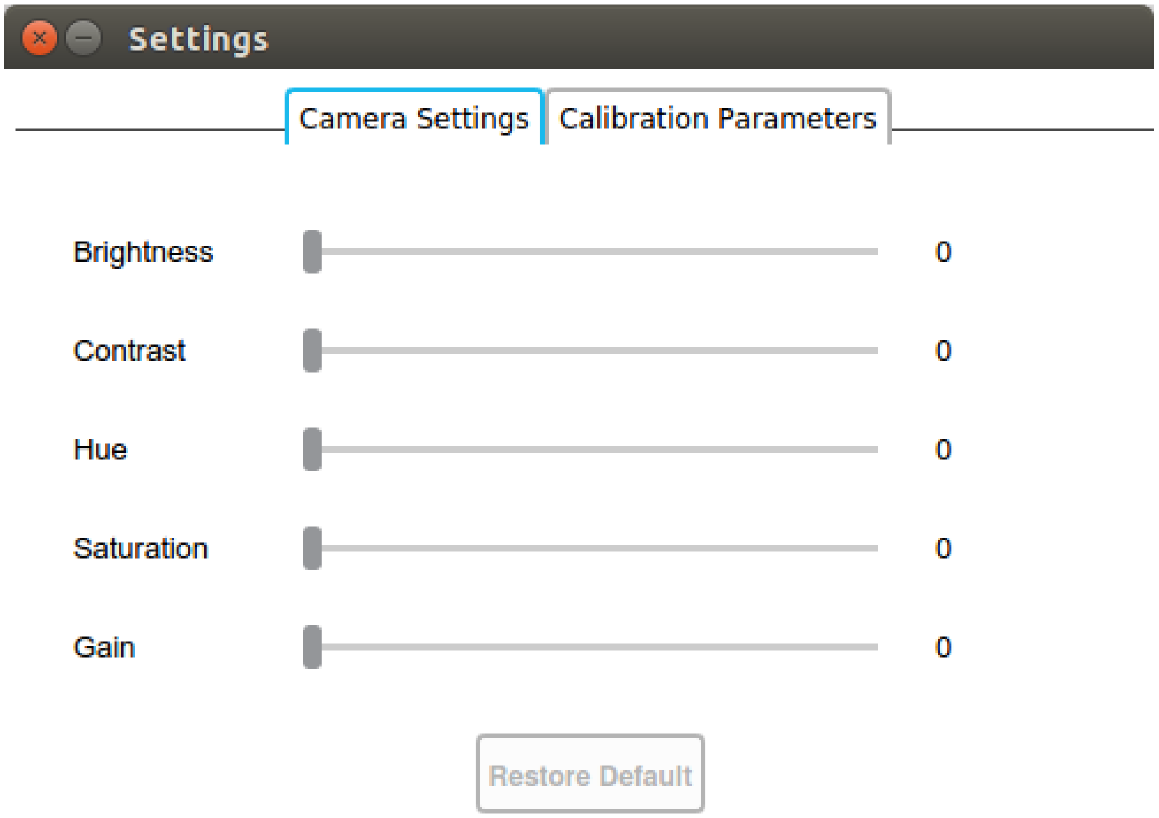


Disk Free Space : 10 Gb

For further information on how to convert SVO files refer to the dedicated **ZED SVO Converter** Sample section.

Managing ZED Camera parameters

A) Managing Sensor Parameters : You can change the different sensor/camera parameters such as brightness, contrast, gain , color temperature as listed in the window. Note that those parameters are kept while the camera is connected. You can also control those parameters through dedicated functions in the ZED SDK.



Brightness 0

Contrast 0

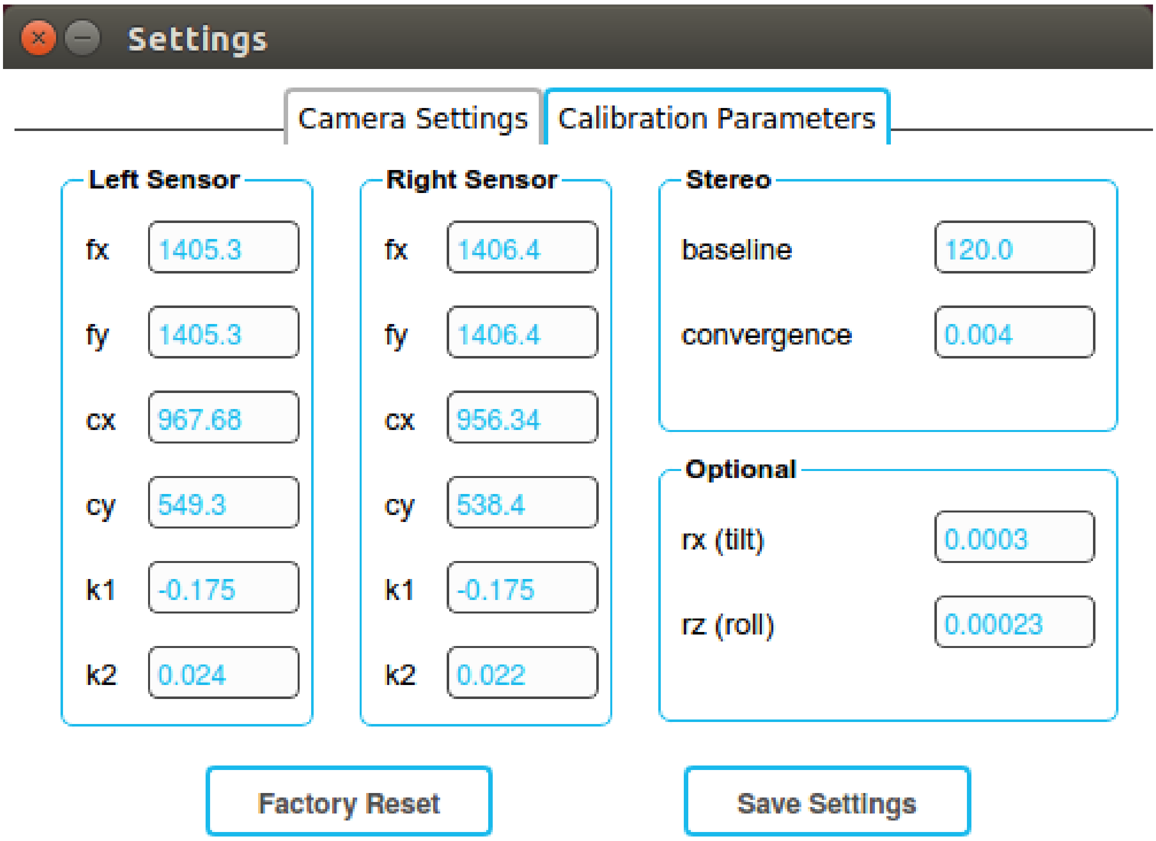
Hue 0

Saturation 0

Gain 0

A) Managing ZED Calibration Parameters : You also have access to ZED Stereo calibration parameters. Those parameters contains the intrinsic parameters of each sensor (Left and Right) and the extrinsic parameters of the stereo camera (baseline, rotations). Changing this parameters will influence the depth estimation, therefore be careful if you want to change them. You can restore calibration parameters from factory by pressing the factory button, or call the ZED Explorer in command line with –dc parameter or download the original file at `http://calib.stereolabs.com`

When changing parameters (factory or on your own), always save configuration ("Save Configuration" button) to apply changes.



Left Sensor

fx	1405.3
fy	1405.3
cx	967.68
cy	549.3
k1	-0.175
k2	0.024

Right Sensor

fx	1406.4
fy	1406.4
cx	956.34
cy	538.4
k1	-0.175
k2	0.022

Stereo

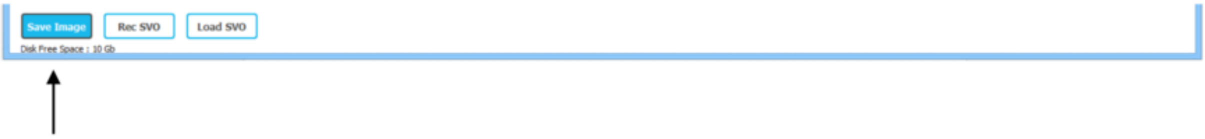
baseline	120.0
convergence	0.004

Optional

rx (tilt)	0.0003
rz (roll)	0.00023

Saving Still Images

You can save a Side by Side raw PNG image by clicking on the **Save Image** button.



Disk Free Space : 10 Gb

ZED Explorer in command line

ZED Explorer can also be used in Command Line to download calibration file, check CUDA compatibility or record svo without GUI.

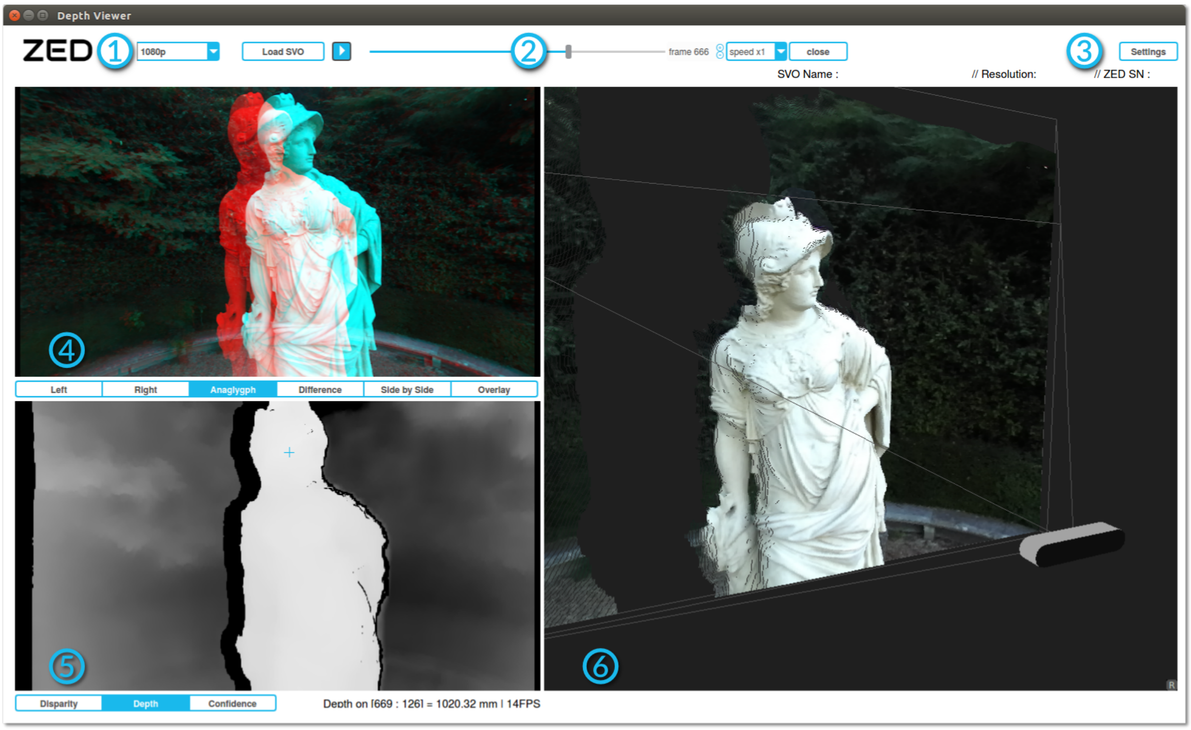
Option	Description	Arguments
--help	Display help message	
--dc <SN>	Download the calibration file for the ZED connected or the ZED with the specified serial number <SN>. if <SN> is not specified, then the first ZED connected and detected will be taken.	(optional) ZEDS S/N
<**name.svo**>	Start a recording of SVO file for the first ZED camera detected, with the specified name	SVO Filename (with desired path)
-r	(Record mode) Resolution of the ZED SVO file	**enum**
-f	(Record mode) Framerate of the desired resolution	**int**
-l	(Record mode) Number of frames to record (auto stop)	**int**
--cc	Launch the CUDA check	

Examples :

```
./ZED\ Explorer Output.svo //Record a SVO file named Output.svo at 1080p (HD1080) 30fps.

./ZED\ Explorer -r HD720 -f 15 -l 200 Output.svo //Record a SVO file named Output.svo at 720p (HD720), 15

./ZED\ Explorer --dc //Download the calibration file for the ZED connected (if connected)

./ZED\ Explorer --dc 1257 //Download the calibration file for the ZED with serial number S/N : 1257.

./ZED\ Explorer --cc // check that CUDA is installed (with the correct toolkit) and that driver is up to d
```

ZED Depth Viewer

The ZED Depth Viewer tool allows you to visualize Depth and Point Cloud computed from the ZED Camera or from an SVO File.



Depth on [669 : 126] = 1020.32 mm | 14FPS

① ZED Camera Resolution (in Live Mode only)

② SVO Control Panel (in SVO Mode only)

③ Settings

④ RGB Frame

⑤ Depth Frame

⑥ 3D Point Cloud Visualizer

ZED Camera Resolution

You can choose the **Resolution** between **2K** / **1080p** / **720p** and **VGA**.

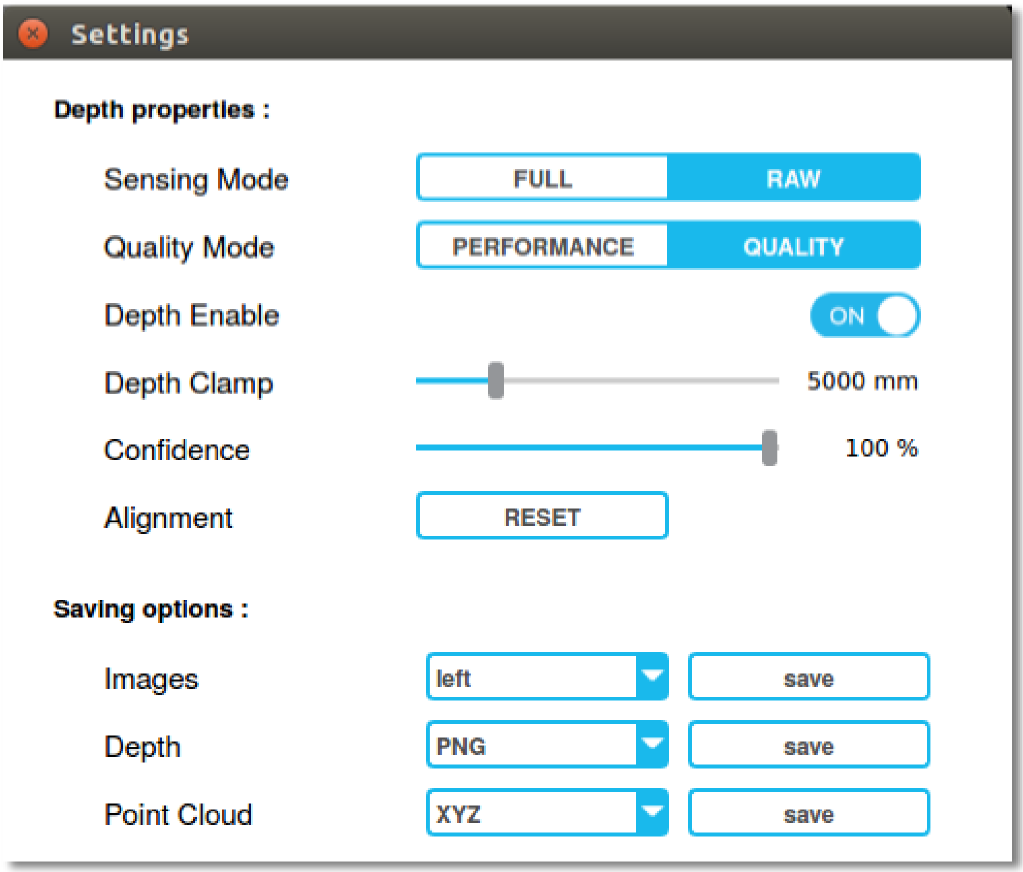
SVO Control Panel

This panel allows you to:

- **Load** an SVO File. You can also directly **Drag & Drop** the file
- **Play/Pause** the video
- Select a frame with the **slider** or with the **+/-** buttons
- Select the video **speed**
- **Close** the file

Settings

This is the Settings panel:



Settings

Depth properties :

Sensing Mode: FULL | RAW

Quality Mode: PERFORMANCE | QUALITY

Depth Enable: ON

Depth Clamp: 5000 mm

Confidence: 100 %

Alignment: RESET

Saving options :

Images: left — save

Depth: PNG — save

Point Cloud: XYZ — save

Several Depth properties are customizable such as Sensing Mode and Quality.

You can also save RGB Images, Depth and Point Cloud in different file formats.

NB: The 3D Point Cloud visualization is **not available** with **FULL** Sensing Mode.

Saving options

Images	Depth	Point Cloud
left	PNG	PCD
right	PGM	PLY
anaglyph	PFM	XYZ
difference		VTK
side by side		
overlay		

RGB Frame

By default, this frame displays the **Anaglyph** view. You can also switch between **Left** View / **Right** View / **Gray Scale Difference** View/ **Side by Side** View and **Overlay** View.

Depth Frame

By default, this frame displays the **Depth map** (with real world depth information). You can also visualize the **Disparity Map** (without real world depth information) or the **Confidence Map** (for confidence of disparity estimation).

3D Visualizer

In this frame you can visualize and move around the **3D Point Cloud**.

Controls

Key	Description
R	**Reset** the view
T	**Top** view
P	**Increase** point size
M	**Decrease** point size

ZED SVO Editor

The ZED SVO Editor tool allows you to cut a part of an SVO file and also merge multiple SVO files into a single one.

Run the program

This program must be run in a command prompt.

Cut an SVO file

```
./ZED\ SVOEditor -cut SVOToCut.svo -s 100 -e 200 Output.svo
```

Merge multiple SVO files

```
./ZED\ SVOEditor -merge SVO1.svo SVO2.svo ... Output.svo
```

Available Options

Option	Description	Arguments
--help	Display help message	
-inf	Print SVO file information	**path** to an SVO file
-cut	Cut an SVO file between **-s** and **-e** frames	**path** to an SVO file
-s	Starting frame If not specified, first frame will be taken	**int**
-e	Ending frame If not specified, last frame will be taken	**int**
-merge	Merge up to 32 SVO files together	**paths** to SVO files last argument is the **path** of the output SVO file

Using the Samples

Each of the samples found in the sample/ directory of the SDK includes binary executable and CMAKE file that can be used to build the sample. If you like, you can use one of these samples as a starting point for building a new application that uses the ZED SDK.

Building the Samples

To build a sample, a very simple way to do it is to use CMAKE with the CMakeLists.txt we provide in command line:

- Create a build directory by calling `mkdir build`
- Enter the build directory by calling `cd build`
- Create the VS project (Windows) or the Makefile (Linux) by calling `cmake ..`
- On Linux, just call make to build the sample
- On Windows, Open the VS project to compile the sample.

You can also checkout our `github page` for more samples.

CPU Samples

ZED SVO Converter

This sample introduces the use of the Stereolabs SVO file format for the ZED.

This sample demonstrates how to grab images and depth with the ZED SDK from an SVO file, display aligned Left and Right images, and compute depth to record it in various formats (image, video...).

An SVO file must be specified in cmd arguments. (See ZED Recorder sample or ZED Explorer to save SVO files)

This sample needs Boost, module *progam_options*. On windows it's advised to:

- build it statically (with 'b2.exe link=static')
- in visual studio project add '{YOUR_BOOST_DIR}/stage/lib' to the linker path and 'libboost_program-_options-XXX.lib' to the dependency
- in visual studio project also add the include dir ('{YOUR_BOOST_DIR}') -> That way boost is no longer a dependency

Available Options

Option	Description	Argument
--help	Display help message	
-f, --filename	SVO filename	**path** to an SVO file
-r, --record	Record a sequence of images Left+Disparity with -z option Left+Right otherwise	

-v, **--video**	Record a video file Left+Disparity with -z option Left+Right otherwise	**filename** WITH ".mp4" EXTENSION
-s, **--nodisplay**	Disable image display	
-z, **--disparity**	Compute disparity	
-q, **--quality**	Disparity Map quality (-z needed)	**1**: PERFORMANCE **2**: QUALITY
-d, **--device**	CUDA device (-z needed)	**int**

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/SVO Converter/src

ZED Grabbing Thread

This sample demonstrates how to grab images and depth map with the ZED SDK using parallel threads.

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/SVO Converter/src

ZED Multi Input (Linux Only)

This sample demonstrates how to use multiple ZEDs into a single program. Each grab is done in a separate thread.

- Tested with 2 ZEDs in HD1080 @ 30fps (ASUS Sabertooth Z87 motherboard and GTX770)
- Tested with 3 ZEDs in HD720 @ 30fps (ASUS Sabertooth Z87 motherboard and GTX770)

A middle-range NVIDIA GPU card is needed with enough memory (2GB is good).

This sample needs OpenCV for image display and basic processing.

WARNING: This sample only works on Linux

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/Multi Input/src

ZED Recorder

This sample demonstrates how to record an SVO file with ZED SDK functions. Optionally, images are displayed using OpenCV.

Available Options

Option	Description	Argument
--help	Display help message	
-d, **--display**	Enable image display (might slow down the recording)	

-f, **--filename**	Output filename	**string**

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/Recorder/src

ZED Save depth

This sample allows you to save depth information provided by the ZED Camera or by an SVO file, in different formats (PNG 16-Bit, PFM, PGM, XYZ, PCD, PLY, VTK).

Keyboard Shortcuts

Main Hotkeys
h: Display help message
p: Save Point Cloud
d: Save Depth
m: Change Point Cloud output format
n: Change Depth output format
q: Quit the program

Available Options

Option	Description	Argument
--help	Display help message	
-o, **--output**	Specify output format	**0**: PNG **1**: PFM
-f, **--filename**	SVO filename	path to an SVO file
-p, **--path**	Output path (can include output filename prefix)	**string**
-s, **--nodisplay**	Disable depth display when reading an SVO file	
-r, **--resolution**	ZED Camera resolution	**0**: HD2K **1**: HD1080 **2**: HD720 **3**: VGA
-q, **--quality**	Disparity Map quality	**1**: PERFORMANCE **2**: QUALITY

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/Save depth/src

ZED SVO Playback

This sample displays an SVO file and allows you to fast-forward or rewind the video. You can also switch between the different views such as Side by Side or Anaglyph.

Keyboard Shortcuts

Main Hotkeys	Display Hotkeys
SPACEBAR: Play/Pause	**0**: Left View
f: fast-forward	**1**: Right View

r: rewind	**2**: Anaglyph View
	3: Gray Scale Difference View
	4: Side by Side View
	5: Overlay View

Source Code

You will find the corresponding source code in [SDK Root]/sample/cpu/SVO Playback/src

ZED with OpenCV

The ZED with OpenCV sample shows how to interface the ZED SDK with the OpenCV library. You will find instructive comments within the sample source code. You can either visualize the Disparity Map (without real world depth information) or the Depth Map (with real world depth information). Press d to toggle between the two.

Depth view: RAW sensing mode

Side by Side view: source 3D image

Call the different views using keys 0-5 on your keyboard.

Press s to view the confidence map.

Toggle between the Sensing Modes with the r – Sensing Mode: RAW - and f keys – Sensing Mode: FULL

You can set the reliability index with the b – Decrease - and n - Increase - keys

You can save a PNG image of the disparity by pressing the v key or an SBS screenshot using the w key.

WARNING: GPU->CPU readback is time consuming. This samples is not designed to operate in real-time.

This sample needs OpenCV.

Main Hotkeys	Display Hotkeys
b: Decrease reliability index by 1	**0**: Left View
n: Increase reliability index by 1	**1**: Right View
r: Sensing Mode: RAW	**2**: Anaglyph View
f: Sensing Mode: FULL	**3**: Gray Scale Difference View
s: Display Confidence Map	**4**: Side by Side View
w: Save Side by Side image in PNG	**5**: Overlay View
v: Save Disparity image in PNG	**d**: Toggle Disparity/Depth View

Source Code

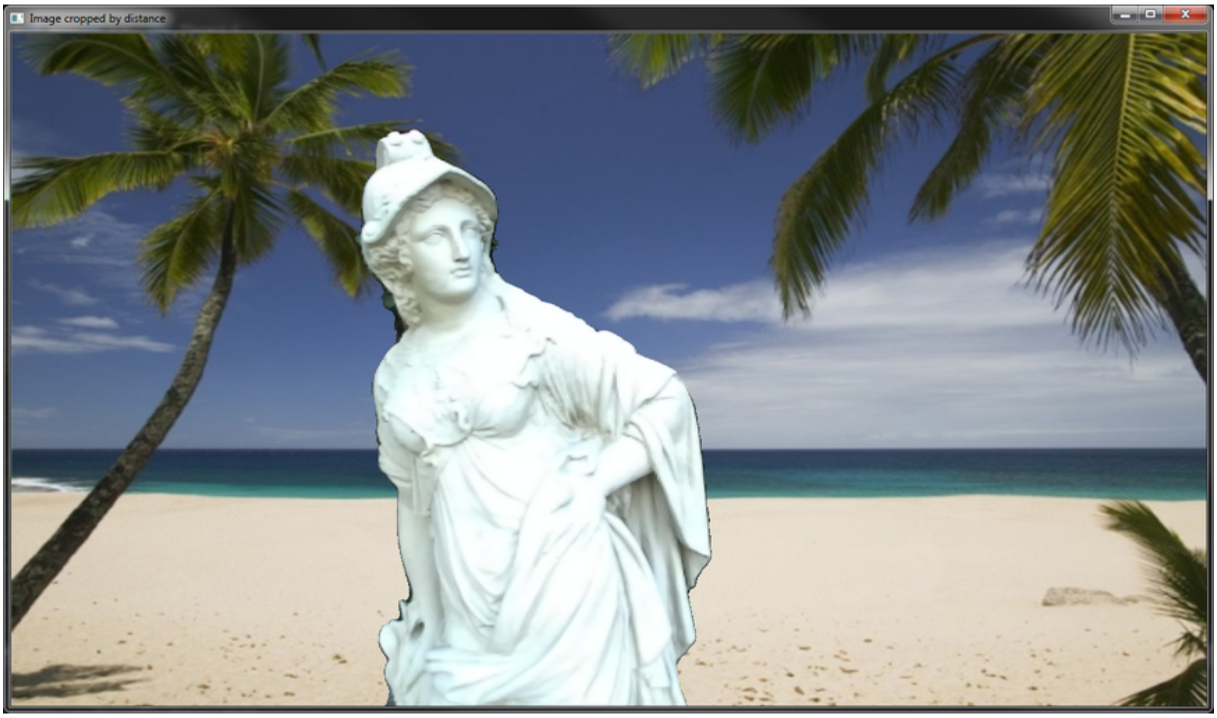
You will find the corresponding source code in [SDK Root]/sample/cpu/with OpenCV/src

GPU Samples

CUDA

ZED Background Substraction

This sample demonstrates how to use the depth to mask the current image with an other.



Some events are linked with keys (using openCV).

This sample needs CUDA and OpenCV.

Keyboard Shortcuts

Main Hotkeys
h: Display help message
p: Increase the distance threshold
m: Decrease the distance threshold

Source Code

You will find the corresponding source code in [SDK Root]/sample/gpu/cuda/background substraction/src

OpenGL

ZED OpenGL 3D Viewer

This sample demonstrates how to grab images and disparity map with the ZED SDK and apply the result in a 3D view "point cloud style" with OpenGL /freeGLUT.

This sample needs freeGLUT (OpenGL Window).

Source Code

You will find the corresponding source code in [SDK Root]/sample/gpu/ogl/openGL 3D Viewer/src

ZED OpenGL GPU Interop

This sample demonstrates the most efficient way to grab and display images and disparity map with the ZED SDK. No GPU->CPU readback is needed to display images, for real-time monitoring.

The GPU buffer is ingested directly into OpenGL texture for avoiding GPU->CPU readback time.

For the Left image, a GLSL shader is used for RGBA–>BGRA transformation , just as an example.

This sample needs freeGLUT (OpenGL Window) and GLEW (for GLSL).

Source Code

You will find the corresponding source code in [SDK Root]/sample/gpu/ogl/openGL gpu interop/src

Release Notes

Release History

ZED SDK 0.9.3 Beta

Platforms

- Compatible with CUDA 7.5 for Windows and Linux.

SDK

- Improved grab() time.
- Reduce Latency when using cpu functions for Jetson.
- Moved Selfcalibration in a non-blocking function. Self-Calibration is done in background and the result can be checked with the status function.
- Improved Init() time

Tools

- Merged ZED Explorer and ZED Settings App into one single tool.
- Added command line options for ZED Explorer for check compatibility and recording

ZED SDK 0.9.2b Beta

SDK

Bug fixes. Patches version.

ZED SDK 0.9.2 Beta

Platforms

- Added Window 10 compatibility.

SDK

- Improved Left/Right image quality with retrieveImage$*$().
- Improved grab time on embedded platform (Jetson TK1).
- Added possibility to disable self-calibration during init().
- Added function that returns results of self-calibration.
- More calibration parameters available.

Tools

- Bug fixes in Depth Viewer tool.
- Added more calibration parameters in ZED Settings App

ZED SDK 0.9.1 Beta

SDK

- Bug fixes for Auto-calibration that made samples crash on Windows 8
- Bug fixes for ply write function that made incoherent color.
- Add more support for slMat2cvMat function.

Tools

- Improved global framerate of Depth Viewer tool.

ZED SDK 0.9.0 Beta

SDK

- Added triangulation function to extract XYZ or XYZRGBA point cloud.
- Added Save functions to save point cloud in multiple formats and depth image in multiple format.
- Added Camera and Current Timestamp extraction function to help synchronization with other devices.
- Added FPS extraction function.
- Bug fixes when using svo files.
- Improved AutoCalibration.

Samples

- Complete re-factory of samples.
- Added Cuda sample for background substraction and image disparity for right image.
- Added ROS Sample to demonstrate how to interact with ROS.
- Added Recording sample to demonstrate how to use recording functions.
- Added SVO Playback sample to demonstrate how to simply use the svo file.
- Simplify existing samples.

Tools

- Added ZED Depth Viewer sample to have a plug and play tool to demonstrate all the functions of the ZED SDK.
- Added Cmd Line mode for ZED Explorer to record svo without Graphics.

ZED SDK 0.8.2 Beta

SDK

- Added flip mode (to work with ZED Camera vertically flipped)
- Added multiple input possibility under Linux system.

Samples

- Added Multiple input sample.

ZED SDK 0.8.1 Beta

SDK

- Major improvement of Disparity estimation in untextured areas and repetitive patterns
- Reduced grab() time by 10%
- Reduced retrieveImage() time to <0.5ms when grab is done with disparity enabled
- Added new function to adjust ZED camera parameters (set / getCameraSettingsValue), including exposure, white balance, brightness, contrast and hue
- Added option to select a different framerate other than the default ones available in the ZED Camera constructor. For example, it is now possible to select VGA mode @ 30 fps
- Added getSVOPosition() to retrieve current SVO position
- Bug fix in ‘Live mode’ on Windows x64
- Added compatibility with NVIDIA GeForce 9 series of graphics cards

Tools

- Minor bug fix in ZED Explorer on Linux

ZED SDK 0.8.0 Beta

Platforms

- Added Linux (Ubuntu 14.04 LTS only) compatibility.

Tools

- Added SVOEditor: command line tool to cut and concatenate SVO files. See -help for more details
- Bug fix in ZED Settings App with "-sn" options
- Minor bug fix in ZED Explorer

SDK

- Improved Disparity estimation
- Bug fix in Disparity/Depth normalization
- Bug fix in NONE grab mode (MODE::NONE)
- Merged MODE::MEDIUM et MODE::QUALITY into a single QUALITY MODE
- Add specific functions for SVO (set position and get number of frames)
- Changed name of "DispReliability" to "ConfidenceThreshold"
- Conformed set/get functions

Known Issues

- Tools or Samples may not launch when run twice on Win8. Reconnecting the ZED will solve this issue.

ZED SDK 0.7.1a Alpha

Notes

- Initial release.

Known Issues

- Specific samples can crash at launch on Win 8. Relaunching the sample solves the problem.

Troubleshooting

This section regroups a non-exhaustive list of events that can occur when installing or using the ZED and its SDK. You can also checkout the `Help Center` for more informations.

FAQ

Why is the ZED Explorer display window black or the video frame rate low?

This problem occurs when the ZED is connected on a USB 2.0 port. Please connect the ZED to a USB 3.0 port and make sure that USB 3.0 drivers are installed.

When I record and play an SVO Video File, the video is skipping frames.

This is due to the speed of your recording/playback media. If you record at high resolution, make sure you are using an SSD drive.

Common Error Messages

"MSVCR120.DLL", "MSVCP110.DLL", "MSVCP120.dll" or "MFC110U.DLL missing"

In order to fix the issue you may install the latest Microsoft Visual C++ 2012 and 2013 Redistributable Packages. You can download these packages from Microsoft website or find them on the ZED USB drive.

"CUDART32_65.DLL missing"

You need to have an NVIDIA graphics card and NVIDIA CUDA installed to run the samples included in the SDK. To fix this issue, download and install CUDA 6.5 from NVIDIA website

"CUDART64_70.DLL / NPPI64_70.DLL / NPPC64_70.DLL missing"

You need to have an NVIDIA graphics card and NVIDIA CUDA installed to run the samples included in the SDK. To fix this issue, download and install CUDA 7.0 from NVIDIA website

"GPU Device Not Found"

Your NVIDIA Display Driver is probably out of date, please update your graphics card display driver with the latest version available on NVIDIA website, or with a driver version compatible with the CUDA toolkit version (6.5/7.0)

"ZED Found / ZED Not Available"

This problem can occur if your computer does not supply enough power through USB to the ZED. Try another USB 3.0 port available.

"Autocalibration Failed"

Make sure the ZED has a clear field of view when you start the sample. It's also recommended to avoid pointing at objects less than 1 m away from the camera, or towards the ground.

Module Index

Modules

Here is a list of all modules:

Namespace Index

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Module Documentation

Public enumeration types

Enumerations

- enum sl::DEPTH_FORMAT { sl::PNG, sl::PFM, sl::PGM, sl::LAST_DEPTH_FORMAT }

  Enumerate the file format for depth saving.

- enum sl::POINT_CLOUD_FORMAT {
  sl::XYZ, sl::PCD, sl::PLY, sl::VTK,
  sl::LAST_CLOUD_FORMAT }

  Enumerate the file format avaliable for saving point clouds (depth triangulation)
- enum sl::zed::MODE { sl::zed::NONE, sl::zed::PERFORMANCE, sl::zed::QUALITY }

  Enumerate for the pre-defined disparity map computation parameters.
- enum sl::zed::VIEW_MODE {
  sl::zed::STEREO_LEFT, sl::zed::STEREO_RIGHT, sl::zed::STEREO_ANAGLYPH, sl::zed::STEREO_DIFF,
  sl::zed::STEREO_SBS, sl::zed::STEREO_OVERLAY, sl::zed::STEREO_LEFT_GREY, sl::zed::STEREO_RIGHT_GREY,
  sl::zed::LAST_VIEW_MODE }

  Enumerate for available output views for monitoring.
- enum sl::zed::SIDE {
  sl::zed::LEFT, sl::zed::RIGHT, sl::zed::LEFT_GREY, sl::zed::RIGHT_GREY,
  sl::zed::LAST_SIDE }

  Enumerate for the Left and Right side of stereo Camera.
- enum sl::zed::SENSING_MODE { sl::zed::FULL, sl::zed::RAW }

  Enumerate for the different types of disparity map computation.
- enum sl::zed::MEASURE {
  sl::zed::DISPARITY, sl::zed::DEPTH, sl::zed::CONFIDENCE, sl::zed::XYZ,
  sl::zed::XYZRGBA, sl::zed::LAST_MEASURE }

  Enumerate for the retrievable measure (each measure should be normalized to be displayed)
- enum sl::zed::ERRCODE {
  sl::zed::SUCCESS, sl::zed::NO_GPU_COMPATIBLE, sl::zed::NOT_ENOUGH_GPUMEM, sl::zed::ZED_NOT_AVAILABLE,
  sl::zed::ZED_RESOLUTION_INVALID, sl::zed::ZED_SETTINGS_FILE_NOT_AVAILABLE, sl::zed::INVALID_SVO_FILE, sl::zed::RECORDER_ERROR,
  sl::zed::LAST_ERRCODE }

  Enumerate for error code during the sl::zed::Camera::init.
- enum sl::zed::ZED_SELF_CALIBRATION_STATUS { sl::zed::SELF_CALIBRATION_NOT_CALLED, sl::zed::SELF_CALIBRATION_RUNNING, sl::zed::SELF_CALIBRATION_FAILED, sl::zed::SELF_CALIBRATION_SUCCESS }

  Status for self calibration. Since v0.9.3, self-calibration is done in background and start in the Init or Reset function.
- enum sl::zed::ZEDResolution_mode {
  sl::zed::HD2K, sl::zed::HD1080, sl::zed::HD720, sl::zed::VGA,
  sl::zed::LAST_RESOLUTION }

  Enumerate for available resolutions for ZED Camera.
- enum sl::zed::ZEDCamera_settings {
  sl::zed::ZED_BRIGHTNESS, sl::zed::ZED_CONTRAST, sl::zed::ZED_HUE, sl::zed::ZED_SATURATION,
  sl::zed::ZED_GAIN, sl::zed::ZED_WHITEBALANCE, sl::zed::LAST_SETTINGS }

  Enumerate the available camera settings for the ZED Camera(whitebalance, contrast, Hue , Saturation ...)

Detailed Description

Enumeration Type Documentation

enum sl::DEPTH_FORMAT

Enumerate the file format for depth saving.

Enumerator

PNG image format, in 16bits, the color encoding the depth

PFM stream of bytes, graphic image file format

PGM grayscale image format

LAST_DEPTH_FORMAT

enum sl::zed::ERRCODE

Enumerate for error code during the sl::zed::Camera::init.

Enumerator

SUCCESS Every step went fine, the camera is ready to use

NO_GPU_COMPATIBLE No GPU found or the CUDA cupability of the device is not sufficient

NOT_ENOUGH_GPUMEM not enough GPU memory for this depth calculation mode please try a "lighter" mode (PERFORMANCE...)

ZED_NOT_AVAILABLE The ZED camera is not plugged or detected

ZED_RESOLUTION_INVALID For Jetson only, resolution not yet supported (USB3.0 bandwidth)>

ZED_SETTINGS_FILE_NOT_AVAILABLE ZED Settings file is not found on the host machine. Use ZED Settings App tool to correct it.

INVALID_SVO_FILE The provided SVO file is not valid

RECORDER_ERROR An recorder related error occured (not enough free storage, invalid file)

LAST_ERRCODE

enum sl::zed::MEASURE

Enumerate for the retrievable measure (each measure should be normalized to be displayed)

Enumerator

DISPARITY Disparity map

DEPTH Depth map

CONFIDENCE Certainty/confidence of the diparity map

XYZ

XYZRGBA

LAST_MEASURE

enum sl::zed::MODE

Enumerate for the pre-defined disparity map computation parameters.

Since v0.8, MEDIUM and QUALITY mode have been combined to provide a single quality mode.

Enumerator

NONE Disparity map not computed, only the rectified images will be available.

PERFORMANCE Fastest mode, also requires less GPU memory, the disparity map is less detailed

QUALITY Better quality mode, the disparity map is more precise

enum sl::POINT_CLOUD_FORMAT

Enumerate the file format avaliable for saving point clouds (depth triangulation)

Enumerator

XYZ 3D format, store the 3D coordinates

PCD Point Cloud Data file, store the 3D coordinates and the color

PLY PoLYgon file format, store the 3D coordinates and the color

VTK Visualization ToolKit file, store the 3D coordinates

LAST_CLOUD_FORMAT

enum sl::zed::SENSING_MODE

Enumerate for the different types of disparity map computation.

Enumerator

FULL Occlusion filling, edge sharpening, advanced post-filtering. Application example : Refocusing, Multi-view generation

RAW Structure conservative, no occlusion filling. Application example : Obstacle detection, 3D reconstructions

enum sl::zed::SIDE

Enumerate for the Left and Right side of stereo Camera.

Enumerator

LEFT

RIGHT

LEFT_GREY

RIGHT_GREY

LAST_SIDE

enum sl::zed::VIEW_MODE

Enumerate for available output views for monitoring.

Enumerator

STEREO_LEFT Left view

STEREO_RIGHT Right view

STEREO_ANAGLYPH Anaglyph (Red & Blue) view

STEREO_DIFF View of the difference between the left image and right image in gray scale

STEREO_SBS Side by Side view (in single image resolution)

STEREO_OVERLAY View of both images in 5O% transparency

STEREO_LEFT_GREY Left view grey scale

STEREO_RIGHT_GREY Right view grey scale

LAST_VIEW_MODE

enum sl::zed::ZED_SELF_CALIBRATION_STATUS

Status for self calibration. Since v0.9.3, self-calibration is done in background and start in the Init or Reset function.

You can follow the current status for the self-calibration any time once ZED object has been construct.

Enumerator

SELF_CALIBRATION_NOT_CALLED Self Calibration has not yet been called (no Init(...) or Reset(...) called

SELF_CALIBRATION_RUNNING Self Calibration is currently running.

SELF_CALIBRATION_FAILED Self Calibration has finished running but did not manage to get coherent values. Old Parameters are taken instead.

SELF_CALIBRATION_SUCCESS Self Calibration has finished running and did manage to get coherent values. New Parameters are taken.

enum sl::zed::ZEDCamera_settings

Enumerate the available camera settings for the ZED Camera(whitebalance, contrast, Hue , Saturation ...)

Each enum defines one of those settings

Enumerator

ZED_BRIGHTNESS Defines the brightness control. Affected value should be between 0 and 8

ZED_CONTRAST Defines the constral control. Affected value should be between 0 and 8

ZED_HUE Defines the hue control. Affected value should be between 0 and 11

ZED_SATURATION Defines the saturation control. Affected value should be between 0 and 8

ZED_GAIN Defines the gain control. Affected value should be between 0 and 8

ZED_WHITEBALANCE Defines the while balance (color temperature) manual control. Affected value should be between 2800 and 6500

LAST_SETTINGS

enum sl::zed::ZEDResolution_mode

Enumerate for available resolutions for ZED Camera.

Enumerator

HD2K 2208$*$1242, supported framerate : 15 fps (unsupported by the Jetson TK1 at the moment)

HD1080 1920$*$1080, supported framerates : 15, 30 fps (unsupported by the Jetson TK1 at the moment)

HD720 1280$*$720, supported framerates : 15, 30, 60 fps, Jetson TK1 : 15 fps

VGA 640$*$480, supported framerates : 15, 30, 60, 100 fps, Jetson TK1 : 15, 30 fps

LAST_RESOLUTION

Namespace Documentation

sl Namespace Reference

Namespaces

- zed

Classes

- struct uchar3Struct
- struct float3Struct
- struct double3Struct

Typedefs

- typedef unsigned char uchar
- typedef struct sl::uchar3Struct uchar3
- typedef struct sl::float3Struct float3
- typedef struct sl::double3Struct double3

Enumerations

- enum DEPTH_FORMAT { PNG, PFM, PGM, LAST_DEPTH_FORMAT }

  Enumerate the file format for depth saving.
- enum POINT_CLOUD_FORMAT {
  XYZ, PCD, PLY, VTK,
  LAST_CLOUD_FORMAT }

  Enumerate the file format avaliable for saving point clouds (depth triangulation)

Functions

- SLSTEREO_EXPORT_DLL bool writeDepthAs (sl::zed::Camera ∗zed, sl::DEPTH_FORMAT format, std::string name, float factor=1.)

  write the current depth map into a file
- SLSTEREO_EXPORT_DLL bool writePointCloudAs (sl::zed::Camera ∗zed, sl::POINT_CLOUD_FORMAT format, std::string name, bool withColor=false, bool keepOccludedPoint=false)

  write the current point cloud into a file

Typedef Documentation

typedef struct sl::double3Struct sl::double3

typedef struct sl::float3Struct sl::float3

typedef unsigned char sl::uchar

typedef struct sl::uchar3Struct sl::uchar3

Function Documentation

SLSTEREO_EXPORT_DLL bool sl::writeDepthAs (sl::zed::Camera ∗ *zed*, sl::DEPTH_FORMAT *format*, std::string *name*, float *factor = 1.*)

write the current depth map into a file

Parameters

DEPTH_FORMAT	: for file format
name	: the name (path) in which the depth will be saved
factor	: only for PNG and PGM, apply a gain to the depth value (default 1.) The maximum value is 65536, so you can set the Camera::setDepthClampValue to 20000 and give a factor to 3, Do not forget to scale (by 1./factor) the pixel value to get the real depth. The occlusions are represented by 0.

Return values

return	false if something wrong happen, else return true

Note

factor : only for PNG and PGM

SLSTEREO_EXPORT_DLL bool sl::writePointCloudAs (sl::zed::Camera ∗ *zed*, sl::POINT_CLOUD_FORMAT *format*, std::string *name*, bool *withColor* = `false`, bool *keepOccludedPoint* = `false`)

write the current point cloud into a file

Parameters

POINT_CLOUD-_FORMAT	: for file format
name	: the name (path) in which the point cloud will be saved
withColor	: indicates if the color must be saved (default false). Not avaliable for XYZ and VTK
keepOccludedPoint	: indicates if the non avaliable data should be saved and set to 0 (default false), if set to true this give a Point Cloud with a size = height ∗ width

Return values

return	false if something wrong happen, else return true

Note

The color is not saved for XYZ and VTK files

sl::zed Namespace Reference

Classes

- class Camera

  The main class to use the ZED camera.
- class Mat

  This class is used to store a buffer (the pointer) of an image and the associated metadata (size, type...).
- struct resolution

  Width and Height of each left and right image.
- struct CamParameters

  Intrinsic parameters of one camera.
- struct StereoParameters

  Intrinsic parameters of each cameras, baseline and convergence (radians)

Enumerations

- enum DATA_TYPE { FLOAT, UCHAR }
- enum MAT_TYPE { CPU, GPU }
- enum MODE { NONE, PERFORMANCE, QUALITY }

  Enumerate for the pre-defined disparity map computation parameters.

- enum VIEW_MODE {
STEREO_LEFT, STEREO_RIGHT, STEREO_ANAGLYPH, STEREO_DIFF,
STEREO_SBS, STEREO_OVERLAY, STEREO_LEFT_GREY, STEREO_RIGHT_GREY,
LAST_VIEW_MODE }

  Enumerate for available output views for monitoring.
- enum SIDE {
LEFT, RIGHT, LEFT_GREY, RIGHT_GREY,
LAST_SIDE }

  Enumerate for the Left and Right side of stereo Camera.
- enum SENSING_MODE { FULL, RAW }

  Enumerate for the different types of disparity map computation.
- enum MEASURE {
DISPARITY, DEPTH, CONFIDENCE, XYZ,
XYZRGBA, LAST_MEASURE }

  Enumerate for the retrievable measure (each measure should be normalized to be displayed)
- enum ERRCODE {
SUCCESS, NO_GPU_COMPATIBLE, NOT_ENOUGH_GPUMEM, ZED_NOT_AVAILABLE,
ZED_RESOLUTION_INVALID, ZED_SETTINGS_FILE_NOT_AVAILABLE, INVALID_SVO_FILE, RECORDER_ERROR,
LAST_ERRCODE }

  Enumerate for error code during the sl::zed::Camera::init.
- enum ZED_SELF_CALIBRATION_STATUS { SELF_CALIBRATION_NOT_CALLED, SELF_CALIBRATION_RUNNING, SELF_CALIBRATION_FAILED, SELF_CALIBRATION_SUCCESS }

  Status for self calibration. Since v0.9.3, self-calibration is done in background and start in the Init or Reset function.
- enum ZEDResolution_mode {
HD2K, HD1080, HD720, VGA,
LAST_RESOLUTION }

  Enumerate for available resolutions for ZED Camera.
- enum ZEDCamera_settings {
ZED_BRIGHTNESS, ZED_CONTRAST, ZED_HUE, ZED_SATURATION,
ZED_GAIN, ZED_WHITEBALANCE, LAST_SETTINGS }

  Enumerate the available camera settings for the ZED Camera(whitebalance, contrast, Hue , Saturation ...)

Functions

- cv::Mat slMat2cvMat (sl::zed::Mat slMat)

  The function cast a sl::zed::Mat into an openCV cv::Mat.
- sl::zed::Mat cvMat2slMat (cv::Mat &cvMat)

  The function cast an openCV cv::Mat into a sl::zed::Mat.
- static std::string errcode2str (ERRCODE err)
- static std::string statuscode2str (ZED_SELF_CALIBRATION_STATUS stat)
- static std::string qualitycode2str (MODE qual)

Variables

- static std::vector< resolution > zedResolution

Enumeration Type Documentation

enum sl::zed::DATA_TYPE

Enumerate defines the type of the stored elements

Enumerator

- ***FLOAT*** float elements (require to cast the pointer)
- ***UCHAR*** unsigned char 8 bits elements

enum sl::zed::MAT_TYPE

Enumerate defines the buffer type

Enumerator

- ***CPU*** the buffer is stored in the memory (CPU memory)
- ***GPU*** the buffer is stored in the CUDA device (Npp pointer, GPU memory)

Function Documentation

sl::zed::Mat sl::zed::cvMat2slMat (cv::Mat & *cvMat*) `[inline]`

The function cast an openCV cv::Mat into a sl::zed::Mat.

Return values

Output	image as a sl::zed::Mat

References sl::zed::Mat::setUp().

static std::string sl::zed::errcode2str (ERRCODE *err*) `[inline]`, `[static]`

References INVALID_SVO_FILE, NO_GPU_COMPATIBLE, NOT_ENOUGH_GPUMEM, RECORDER_ERROR, SUCCESS, ZED_NOT_AVAILABLE, ZED_RESOLUTION_INVALID, and ZED_SETTINGS_FILE_NOT_AVAILABLE.

static std::string sl::zed::qualitycode2str (MODE *qual*) `[inline]`, `[static]`

References NONE, PERFORMANCE, and QUALITY.

cv::Mat sl::zed::slMat2cvMat (sl::zed::Mat *slMat*) `[inline]`

The function cast a sl::zed::Mat into an openCV cv::Mat.

Return values

Output	image as a cv::Mat (opencv format)

References sl::zed::Mat::channels, sl::zed::Mat::data, sl::zed::Mat::data_type, sl::zed::Mat::getDataSize(), sl::zed::Mat::height, and sl::zed::Mat::width.

static std::string sl::zed::statuscode2str (ZED_SELF_CALIBRATION_STATUS *stat*) `[inline]`, `[static]`

References SELF_CALIBRATION_FAILED, SELF_CALIBRATION_NOT_CALLED, SELF_CALIBRATION_RUNNING, and SELF_CALIBRATION_SUCCESS.

Variable Documentation

std::vector<resolution> sl::zed::zedResolution `[static]`

Initial value:

```
= [] {
            std::vector<resolution> v;
            v.push_back(resolution(2208, 1242));
            v.push_back(resolution(1920, 1080));
            v.push_back(resolution(1280, 720));
            v.push_back(resolution(640, 480));
            return v;
        }()
```

Available mode for the ZED camera sensors

Class Documentation

sl::zed::Camera Class Reference

The main class to use the ZED camera.

Collaboration diagram for sl::zed::Camera:

Public Member Functions

- Camera (ZEDResolution_mode zedRes_mode=ZEDResolution_mode::HD720, float fps=0.0, int zed_linux_id=0)

  Camera constructor. The ZEDResolution_mode sets the sensor resolution and defines the size of the output images, including the measures (disparity map, confidence map..).
- Camera (std::string zed_record_filename)

  Camera constructor, from svo file previously recorded. The size of the output is defined by the recorded images.
- ∼Camera ()
- ERRCODE init (MODE quality, int device=-1, bool verbose=false, bool vflip=false, bool disable_self_calib=false)

  The init function must be called after the instantiation. The function checks if the ZED camera is plugged and opens it, if the graphics card is compatible, allocates the memory and launches the automatic calibration.
- bool grab (SENSING_MODE dm_type=SENSING_MODE::RAW, bool computeMeasure=1, bool computeDisparity=1, bool computeXYZ=0)

  The function grabs a new image, rectifies it and computes the disparity map and optionally the depth map. The grabbing function is typically called in the main loop.
- Mat getView (VIEW_MODE view)

  Gets a CPU image to display. Several modes available SidebySide, anaglyph... (for more see sl::zed::VIEW_MODE)
- Mat getView_gpu (VIEW_MODE view)

  Gets a GPU image to display. Several modes available SidebySide, anaglyph... (for more see sl::zed::VIEW_MODE)
- Mat retrieveImage (SIDE side)

  Downloads the rectified image from the device and returns the CPU buffer. The retrieve function should be called after the function Camera::grab.
- Mat retrieveMeasure (MEASURE measure)

  Downloads the measure (disparity, depth or confidence of disparity) from the device and returns the CPU buffer. The retrieve function should be called after the function Camera::grab.
- Mat retrieveImage_gpu (SIDE side)

  Gets the rectified image GPU buffer. The retrieve function should be called after the function Camera::grab.
- Mat retrieveMeasure_gpu (MEASURE measure)

  Gets the measure (disparity, depth or certainty of disparity) GPU buffer. The retrieve function should be called after the function Camera::grab.
- Mat normalizeMeasure (MEASURE measure, float min=0, float max=0)

  Performs a GPU normalization of the measure value and download the result as a CPU image.

- Mat normalizeMeasure_gpu (MEASURE measure, float min=0, float max=0)

  GPU Normalization of the measure value and get the result as a GPU image.
- StereoParameters ∗ getParameters ()

  Gets the ZED parameters.
- void setConfidenceThreshold (int ThresholdIdx)

  Sets a filtering value for the disparity map (and by extension the depth map). The function should be called before Camera::grab to be taken into account.
- int getConfidenceThreshold ()

  Gets the current confidence threshold value for the disparity map (and by extension the depth map).
- CUcontext getCUDAContext ()

  Gets the CUDA context used for all the computation.
- resolution getImageSize ()

  Gets the image size.
- unsigned int getZEDSerial ()

  Gets the ZED Serial Number.
- unsigned int getZEDFirmware ()

  Gets the ZED Current Firmware version.
- bool setSVOPosition (int frame)

  Sets the position of the SVO file to a desired frame.
- int getSVOPosition ()

  Get the current position of the SVO file.
- int getSVONumberOfFrames ()

  Get the number of frames in the SVO file.
- void setDepthClampValue (int distanceMax)

  Set the maximum distance of depth/disparity estimation (all values after this limit will be reported as TOO_FAR value)
- int getDepthClampValue ()

  Get the current maximum distance of depth/disparity estimation.
- void setCameraSettingsValue (ZEDCamera_settings mode, int value, bool usedefault=false)

  Set the value to the corresponding Camera Settings mode (Gain, brightness, hue, white balance....)
- int getCameraSettingsValue (ZEDCamera_settings mode)

  Get the current value to the corresponding Camera Settings mode (Gain, brightness, hue, white balance....)
- float getCurrentFPS ()

  Get the current fps of the camera.
- long long getCameraTimestamp ()

  Get the timestamp at the time the frame has been extracted from USB stream. (should be called after a grab())
- long long getCurrentTimestamp ()

  Get the current timestamp at the time the function is called. Can be compared to the camera timestamp for synchronization.
- void setBaselineRatio (float ratio)

  Convert depth values from mm to desired unit, depending on ratio value.
- ZED_SELF_CALIBRATION_STATUS getSelfCalibrationStatus ()

  Get the current status of the self-calibration.
- sl::double3 getSelfCalibrationRotation ()

  Get the estimated rotation from the self-calibration.
- bool resetSelfCalibration ()

  The reset function can be called at any time AFTER the Camera::init function has been called. It will reset and calculate again correction for misalignment, convergence and color mismatch. It can be called after changing camera parameters without needing to restart your executable.
- float getClosestDepthValue ()

  Get the closest measurable distance by the camera, according to the camera and the depth map parameters.
- bool setFPS (int desiredFPS)

Set a new frame rate for the camera, or the closest avaliable frame rate.

- void setFlip (bool flip)

  Useful if you use the camera upside down, this will flip the images so you can get the images in a normal way.
- ERRCODE initRecording (std::string filepath, bool avi_file=false, bool side_by_side=true)

  Initializes the recorder.
- bool record ()

  Records one camera frame. Typically called in a loop, without calling grab function.
- sl::zed::Mat getCurrentRawRecordedFrame ()

  Get the current side by side YUV 4:2:2 frame, CPU buffer.
- bool stopRecording ()

  Stops the recording and closes the file.
- void displayRecorded ()

  Displays the current recorded frame.

Static Public Member Functions

- static std::string getSDKVersion ()

  The function return the version of the currently installed ZED SDK.
- static int isZEDconnected ()

  The function checks if ZED Cameras are connected, can be call before instantiate a Camera object.
- static int sticktoCPUCore (int cpu_core)

  The function stick the calling thread to a specific CPU core. This function is only available for Jetson TK1 and TX1.

Detailed Description

The main class to use the ZED camera.

Constructor & Destructor Documentation

sl::zed::Camera::Camera (ZEDResolution_mode *zedRes_mode* = `ZEDResolution_mode::HD720`, float *fps* = `0.0`, int *zed_linux_id* = `0`)

Camera constructor. The ZEDResolution_mode sets the sensor resolution and defines the size of the output images, including the measures (disparity map, confidence map..).

All computation is done on a CUDA capable device (That means that every CPU computation will need a memory retrieve of the images, which takes some time). If the performance is the main focus, all external computation should run on GPU. The retrieve∗_gpu gives directly access to the gpu buffer.

Parameters

zedRes_mode	: the chosen ZED resolution
fps	: a requested fps for this resolution. set as 0.0 will choose the default fps for this resolution (see User guide)
zed_linux_id	: ONLY for LINUX : if multiple ZEDs are connected, it will choose the first zed listed (if zed_linux_id=0), the second listed (if zed_linux_id=1), ... Each ZED will create its own memory (CPU and GPU), therefore the number of ZED available will depend on the configuration of your computer. Currently not available for Windows

sl::zed::Camera::Camera (std::string *zed_record_filename*)

Camera constructor, from svo file previously recorded. The size of the output is defined by the recorded images.

Parameters

zed_record_-filename	: path with filename to the recorded svo file

sl::zed::Camera::∼Camera ( )

Member Function Documentation

void sl::zed::Camera::displayRecorded ( )

Displays the current recorded frame.

Warning

> Might reduce the recording framerate

int sl::zed::Camera::getCameraSettingsValue (ZEDCamera_settings *mode*)

Get the current value to the corresponding Camera Settings mode (Gain, brightness, hue, white balance....)

Parameters

ZEDCamera_-settings	: enum for the control mode

Return values

Current	value for the corresponding control (-1 if something wrong happened).

long long sl::zed::Camera::getCameraTimestamp ( )

Get the timestamp at the time the frame has been extracted from USB stream. (should be called after a grab())

Return values

Current	Timestamp in ns. Return -1 if working with svo files.

float sl::zed::Camera::getClosestDepthValue ( )

Get the closest measurable distance by the camera, according to the camera and the depth map parameters.

Return values

float	: the distance in mm (or baseline unit if changed)

int sl::zed::Camera::getConfidenceThreshold ( )

Gets the current confidence threshold value for the disparity map (and by extension the depth map).

Return values

	current filtering value between 0 and 100.

CUcontext sl::zed::Camera::getCUDAContext ( )

Gets the CUDA context used for all the computation.

Return values

CUDA_context	: the CUcontext

float sl::zed::Camera::getCurrentFPS ( )

Get the current fps of the camera.

Return values

Current	FPS. Return 0 if working with SVO files or -1.f if something goes wrong.

sl::zed::Mat sl::zed::Camera::getCurrentRawRecordedFrame ( )

Get the current side by side YUV 4:2:2 frame, CPU buffer.

Warning

> The buffer must be duplicated if record() is called, it will be overwritten otherwise. The buffer must not be freed. Undefined behavior if called outside recorder mode.

long long sl::zed::Camera::getCurrentTimestamp ( )

Get the current timestamp at the time the function is called. Can be compared to the camera timestamp for synchronization.

Return values

Current	Timestamp in ns. Return -1 if working with svo files.

int sl::zed::Camera::getDepthClampValue ( )

Get the current maximum distance of depth/disparity estimation.

Return values

	current maximum distance in mm (or baseline unit)

resolution sl::zed::Camera::getImageSize ( )

Gets the image size.

Return values

resolution	: the image size

StereoParameters∗ sl::zed::Camera::getParameters ( )

Gets the ZED parameters.

Return values

StereoParameters	pointer containing the intrinsic parameters of each camera and the baseline (mm) and convergence (radian) of the ZED.

static std::string sl::zed::Camera::getSDKVersion ( ) `[static]`

The function return the version of the currently installed ZED SDK.

Return values

ZED	SDK version as a string with the following format : MAJOR.MINOR.PATCH

sl::double3 sl::zed::Camera::getSelfCalibrationRotation ( )

Get the estimated rotation from the self-calibration.

Return values

double3	(structure of 3 double values) of the Rotation (one component for each axis) <struct>.d1 will return the Rx component ("Tilt" or "Pitch") <struct>.d2 will return the Ry component ("Convergence") <struct>.d3 will return the Rz component ("Roll") All values are in radians.

ZED_SELF_CALIBRATION_STATUS sl::zed::Camera::getSelfCalibrationStatus ( )

Get the current status of the self-calibration.

Return values

ZED_SELF_CALIBRATION_STATUS	: The status code gives information about the self calibration status. For more details see sl::zed::ZED_SELF_CALIBRATION_STATUS.

int sl::zed::Camera::getSVONumberOfFrames ( )

Get the number of frames in the SVO file.

Return values

SVO	Style Only : the total number of frames in the SVO file (-1 if the SDK is not reading a SVO)

int sl::zed::Camera::getSVOPosition ( )

Get the current position of the SVO file.

Return values

SVO	Style Only : the current position in the SVO file as int (-1 if the SDK is not reading a SVO)

Mat sl::zed::Camera::getView (VIEW_MODE *view*)

Gets a CPU image to display. Several modes available SidebySide, anaglyph... (for more see sl::zed::VIEW_MODE)

Parameters

view	: the wanted mode

Return values

View	: the return value is a CPU sl::zed::Mat.

Warning

> The buffer must be duplicated if another CPU retrieve has to be done, it will be overwritten otherwise. The buffer must not be freed.

Mat sl::zed::Camera::getView_gpu (VIEW_MODE *view*)

Gets a GPU image to display. Several modes available SidebySide, anaglyph... (for more see sl::zed::VIEW-_MODE)

Parameters

view	: the wanted mode

Return values

View	: the return value is a GPU sl::zed::Mat.

Warning

> The buffer must be duplicated if another GPU retrieve has to be done, it will be overwritten otherwise. The buffer must not be freed.

unsigned int sl::zed::Camera::getZEDFirmware ( )

Gets the ZED Current Firmware version.

Return values

Returns	the ZED Firmware version (as uint), 0 if the ZED is not connected.

unsigned int sl::zed::Camera::getZEDSerial ( )

Gets the ZED Serial Number.

Return values

Returns	the ZED Serial Number (as uint) (Live or SVO).

bool sl::zed::Camera::grab (SENSING_MODE *dm_type* = `SENSING_MODE::RAW`, bool *computeMeasure* = `1`, bool *computeDisparity* = `1`, bool *computeXYZ* = `0`)

The function grabs a new image, rectifies it and computes the disparity map and optionally the depth map. The grabbing function is typically called in the main loop.

Parameters

dm_type	: defines the type of disparity map, more info : sl::zed::SENSING_MODE definition
computeMeasure	: defines if the depth map should be computed. If false, only the disparity map is computed.
computeDisparity	: defines if the disparity map should be computed. If false, the depth map can't be computed and only the camera images can be retrieved.

Return values

error	: the function returns false if no problem was encountered, true otherwise.

Precondition

> Camera::init function must have been called (once for the lifetime of the Camera object) before calling Camera::grab.

ERRCODE sl::zed::Camera::init (MODE *quality*, int *device* = `-1`, bool *verbose* = `false`, bool *vflip* = `false`, bool *disable_self_calib* = `false`)

The init function must be called after the instantiation. The function checks if the ZED camera is plugged and opens it, if the graphics card is compatible, allocates the memory and launches the automatic calibration.

Parameters

quality	: defines the quality of the disparity map, affects the level of details and also the computation time. The MODE::QUALITY requires a quite powerful graphics card and more gpu memory, if the frame rate is too low try a lower quality
device	: defines the graphics card on which the computation will be done. The default value -1 search the powerful usable GPU. The computation should be launched on the fastest card.
verbose	: if set to true, it will output some information about the current status of initialization
vflip	: if set to true, it will vertically flipped frames coming from the ZED. (for "Hang out" mode)
disable_self_-calib	: if set to true, it will disable self-calibration and take the optional calibration parameters without optimizing them. It is adviced to leave it as false, so that calibration parameters can be optimized.

Return values

ERRCODE	: The error code gives information about the internal process, if SUCCESS is returned, the camera is ready to use. Every other code indicates an error and the program should be stopped. For more details see sl::zed::ERRCODE.

ERRCODE sl::zed::Camera::initRecording (std::string *filepath*, bool *avi_file* = `false`, bool *side_by_side* = `true`)

Initializes the recorder.

Parameters

filepath	: path with absolute or relative filename of the recorded file
avi_file	: record in avi (not compatible with the ZED SDK as an input) rather than svo
side_by_side	: record one avi file with side by side images, left and right otherwise

Note

The disparity/depth map is neither recorded nor computed

Warning

Camera::init mustn't be called.

static int sl::zed::Camera::isZEDconnected ( ) `[static]`

The function checks if ZED Cameras are connected, can be call before instantiate a Camera object.

Return values

the	number of connected ZED

Warning

On Windows only one ZED is accessible so this function will return even if multiple ZED are connected.

Mat sl::zed::Camera::normalizeMeasure (MEASURE *measure*, float *min* = *0*, float *max* = *0*)

Performs a GPU normalization of the measure value and download the result as a CPU image.

Parameters

MEASURE	: defines the requested output (see sl::zed::MEASURE)
min	: defines the lower bound of the normalization, default : automatically found
max	: defines the upper bound of the normalization, default : automatically found

Return values

normalized_measure	: the CPU buffer.

Warning

> The buffer must be duplicated if another CPU retrieve has to be done, it'll be overwritten otherwise. The buffer must not be freed.

Mat sl::zed::Camera::normalizeMeasure_gpu (MEASURE *measure*, float *min* = *0*, float *max* = *0*)

GPU Normalization of the measure value and get the result as a GPU image.

Parameters

MEASURE	: defines the requested output (see sl::zed::MEASURE)
min	: defines the lower bound of the normalization, default : automatically found
max	: defines the upper bound of the normalization, default : automatically found

Return values

normalized_measure	: the GPU buffer.

Warning

> The buffer must be duplicated if another GPU retrieve has to be done, it'll be overwritten otherwise. The buffer must not be freed.

bool sl::zed::Camera::record ( )

Records one camera frame. Typically called in a loop, without calling grab function.

Warning

> Camera::grab mustn't be called.

bool sl::zed::Camera::resetSelfCalibration ( )

The reset function can be called at any time AFTER the Camera::init function has been called. It will reset and calculate again correction for misalignment, convergence and color mismatch. It can be called after changing camera parameters without needing to restart your executable.

Return values

ERRCODE	: error boolean value : the function returns false if no problem was encountered, true otherwise. if no problem was encountered, the camera will use new parameters. Otherwise, it will be the old ones

Mat sl::zed::Camera::retrieveImage (SIDE *side*)

Downloads the rectified image from the device and returns the CPU buffer. The retrieve function should be called after the function Camera::grab.

Parameters

SIDE	: defines the image side wanted (see sl::zed::SIDE)

Return values

Image	: the return value is a CPU sl::zed::Mat. The size is given by the input parameters of the constructor Camera.

Warning

> The buffer must be duplicated if another CPU retrieve has to be done, it will be overwritten otherwise. The buffer must not be freed.

Mat sl::zed::Camera::retrieveImage_gpu (SIDE *side*)

Gets the rectified image GPU buffer. The retrieve function should be called after the function Camera::grab.

Parameters

SIDE	: defines the image side wanted (see sl::zed::SIDE)

Return values

Image	: the return value is a GPU sl::zed::Mat. The size is given by the input parameters of the constructor Camera.

Warning

> The buffer must not be freed.

Mat sl::zed::Camera::retrieveMeasure (MEASURE *measure*)

Downloads the measure (disparity, depth or confidence of disparity) from the device and returns the CPU buffer. The retrieve function should be called after the function Camera::grab.

Parameters

MEASURE	: defines the type wanted, such as disparity map, depth map or the confidence (see sl::zed::MEASURE)

Return values

Measure	: the return value is a CPU sl::zed::Mat. For Depth measure, values are given in mm. For Confidence map, a value close to 1 means a good precision, a value close to 100 means less precision.

Warning

> The buffer must be duplicated if another CPU retrieve has to be done, it will be overwritten otherwise. The buffer must not be freed.

Mat sl::zed::Camera::retrieveMeasure_gpu (MEASURE *measure*)

Gets the measure (disparity, depth or certainty of disparity) GPU buffer. The retrieve function should be called after the function Camera::grab.

Parameters

MEASURE	: defines the type wanted, such as disparity map, depth map or the disparity map certainty (see sl::zed::MEASURE)

Return values

Measure	: the return value is a GPU sl::zed::Mat. The size is given by the input parameters of the constructor Camera. For Depth measure, values are given in mm.

Warning

The buffer must not be freed.

void sl::zed::Camera::setBaselineRatio (float *ratio*)

Convert depth values from mm to desired unit, depending on ratio value.

Parameters

ratio	: value used to convert mm to the desired unit. Example 1 : ratio=0.001 will convert mm to meters Example 2 : ratio=0.0393701 will convert mm to inches. : The depth unit is linked to the baseline unit in the configuration file. By default, this value is in mm. In the above example, we assume that baseline unit is still in mm.

void sl::zed::Camera::setCameraSettingsValue (ZEDCamera_settings *mode*, int *value*, bool *usedefault* = `false`)

Set the value to the corresponding Camera Settings mode (Gain, brightness, hue, white balance....)

Parameters

ZEDCamera_-settings	: enum for the control mode
value	: value to set for the corresponding control
usedefault	: will set default (or automatic) value if set to true (value (int) will not be taken into account)

void sl::zed::Camera::setConfidenceThreshold (int *ThresholdIdx*)

Sets a filtering value for the disparity map (and by extension the depth map). The function should be called before Camera::grab to be taken into account.

Parameters

ThresholdIdx	: a value in [1,100]. A lower value means more confidence and precision (but less density), an upper value reduces the filtering (more density, less certainty). Other value means no filtering.

void sl::zed::Camera::setDepthClampValue (int *distanceMax*)

Set the maximum distance of depth/disparity estimation (all values after this limit will be reported as TO-O_FAR value)

Parameters

distanceMax	: maximum distance in mm (or baseline unit)

void sl::zed::Camera::setFlip (bool *flip*)

Useful if you use the camera upside down, this will flip the images so you can get the images in a normal way.

Parameters

flip	: apply or not the vertical flip

bool sl::zed::Camera::setFPS (int *desiredFPS*)

Set a new frame rate for the camera, or the closest avaliable frame rate.

Parameters

desiredFPS	: the new desired frame rate

Return values

bool	: return an error if something wrong happen

bool sl::zed::Camera::setSVOPosition (int *frame*)

Sets the position of the SVO file to a desired frame.

Parameters

frame	: the number of the desired frame to be decoded.

Return values

true	if succes, false if failed (i.e. if the ZED is currently used in live and not playing a SVO file)

static int sl::zed::Camera::sticktoCPUCore (int *cpu_core*) `[static]`

The function stick the calling thread to a specific CPU core. This function is only available for Jetson TK1 and TX1.

Parameters

cpu_core	: int that defines the core the thread must be run on. could be between 0 and 3. (cpu0,cpu1,cpu2,cpu3).

Return values

0	is stick is OK, otherwise status error.

Warning

> Function only available for Jetson. On other platform, result will be always 0 and no operations are performed.

bool sl::zed::Camera::stopRecording ( )

Stops the recording and closes the file.

sl::zed::CamParameters Struct Reference

Intrinsic parameters of one camera.

Collaboration diagram for sl::zed::CamParameters:

Public Member Functions

- void SetUp (float fx_, float fy_, float cx_, float cy_)

Public Attributes

- float fx
- float fy
- float cx
- float cy
- double disto [5]

Detailed Description

Intrinsic parameters of one camera.

Member Function Documentation

void sl::zed::CamParameters::SetUp (float *fx_*, float *fy_*, float *cx_*, float *cy_*) `[inline]`

References cx, cy, fx, and fy.

Member Data Documentation

float sl::zed::CamParameters::cx

Optical center x

Referenced by SetUp().

float sl::zed::CamParameters::cy

Optical center y

Referenced by SetUp().

double sl::zed::CamParameters::disto[5]

Distortion factor : k1, k2, k3, r1, r2

float sl::zed::CamParameters::fx

Focal x

Referenced by SetUp().

float sl::zed::CamParameters::fy

Focal y

Referenced by SetUp().

sl::double3Struct Struct Reference

Collaboration diagram for sl::double3Struct:

Public Member Functions

- double3Struct (float d1_=0, float d2_=0, float d3_=0)

Public Attributes

- double d1
- double d2
- double d3

Constructor & Destructor Documentation

sl::double3Struct::double3Struct (float *d1_* = *0*, float *d2_* = *0*, float *d3_* = *0*) `[inline]`

References d1, d2, and d3.

Member Data Documentation

double sl::double3Struct::d1

Referenced by double3Struct().

double sl::double3Struct::d2

Referenced by double3Struct().

double sl::double3Struct::d3

Referenced by double3Struct().

sl::float3Struct Struct Reference

Collaboration diagram for sl::float3Struct:

Public Member Functions

- float3Struct (float f1_=0, float f2_=0, float f3_=0)

Public Attributes

- float f1
- float f2
- float f3

Constructor & Destructor Documentation

sl::float3Struct::float3Struct (float *f1_ = 0*, float *f2_ = 0*, float *f3_ = 0*) `[inline]`

References f1, f2, and f3.

Member Data Documentation

float sl::float3Struct::f1

Referenced by float3Struct().

float sl::float3Struct::f2

Referenced by float3Struct().

float sl::float3Struct::f3

Referenced by float3Struct().

sl::zed::Mat Class Reference

This class is used to store a buffer (the pointer) of an image and the associated metadata (size, type...).

Collaboration diagram for sl::zed::Mat:

Public Member Functions

- Mat ()

  Mat default constructor.
- Mat (int width, int height, int channels, DATA_TYPE data_type, sl::uchar ∗data, MAT_TYPE mat_type=MAT_TYPE::CPU)

  Mat constructor.
- sl::uchar3 getValue (int x, int y) const

  Gets the value of the image at the given coordinate.
- void setUp (int width, int height, int step, sl::uchar ∗data, int channels, DATA_TYPE data_type, MAT_TYPE mat_type=CPU)

  Sets up the buffer metadata.
- void setUp (int width, int height, int channels, DATA_TYPE data_type, MAT_TYPE mat_type=CPU)

  Sets up the buffer metadata.
- void allocate_cpu (int width, int height, int channels, DATA_TYPE data_type)

  Allocates the memory.
- void allocate_cpu ()

  Allocates the memory with the metadata previously set.

- void deallocate ()

  Deallocates the memory.
- int getWidthByte () const

  Gets the width of the image in bytes, useful for the npp function (take into account the number of channels)
- int getDataSize () const

  Gets the size of the buffer type.
- void info ()

  Prints in the standard output information about the Mat.

Public Attributes

- int width
- int height
- int step
- sl::uchar ∗ data
- int channels
- DATA_TYPE data_type
- MAT_TYPE type

Detailed Description

This class is used to store a buffer (the pointer) of an image and the associated metadata (size, type...).

Constructor & Destructor Documentation

sl::zed::Mat::Mat ( )

Mat default constructor.

sl::zed::Mat::Mat (int *width*, int *height*, int *channels*, DATA_TYPE *data_type*, sl::uchar ∗ *data*, MAT_TYPE *mat_type* = `MAT_TYPE::CPU`)

Mat constructor.

Parameters

width	: width of the image in pixels
height	: height of the image in pixels
channels	: number of channels in the image
data_type	: type of stored element (see sl::zed::DATA_TYPE). Can be float or uchar.
data	: the buffer
mat_type	: defines where the buffer is stored (CPU or GPU memory)

Warning

> The buffer memory is NEITHER automatically allocated NOR freed and requires an explicit call of the corresponding function (Exception : for some of the sl::zed::Camera functions, the sl::zed::Mat may already be pre-allocated and must not be freed. For more information, refer to the corresponding function documentation)

Member Function Documentation

void sl::zed::Mat::allocate_cpu (int *width*, int *height*, int *channels*, DATA_TYPE *data_type*)

Allocates the memory.

Parameters

width	: width of the image in pixels
height	: height of the image in pixels
channels	: number of channels in the image
data_type	: type of element stored

Warning

> The Mat is required to be a CPU Mat

void sl::zed::Mat::allocate_cpu ( )

Allocates the memory with the metadata previously set.

Precondition

> All the metadata must have been filled before calling allocate to prevent an undefined behavior

Warning

> The Mat is required to be a CPU Mat

void sl::zed::Mat::deallocate ( )

Deallocates the memory.

int sl::zed::Mat::getDataSize ( ) const

Gets the size of the buffer type.

Note

> sizeof() is called on Mat::data using the real type

Return values

size	in bytes

Referenced by sl::zed::slMat2cvMat().

sl::uchar3 sl::zed::Mat::getValue (int *x*, int *y*) const

Gets the value of the image at the given coordinate.

Parameters

x	coordinate
y	coordinate

Return values

pixel_value	If the Mat has less than 3 channels a uchar3 is still produced with the value 0 for the missing channels. If the Mat has more than 3 channels the value correspond to the first 3 channels and the others are ignored

int sl::zed::Mat::getWidthByte ( ) const

Gets the width of the image in bytes, useful for the npp function (take into account the number of channels)

Return values

width_byte	: the width in bytes

void sl::zed::Mat::info ( )

Prints in the standard output information about the Mat.

void sl::zed::Mat::setUp (int *width*, int *height*, int *step*, sl::uchar ∗ *data*, int *channels*, DATA_TYPE *data_type*, MAT_TYPE *mat_type* = CPU)

Sets up the buffer metadata.

Parameters

width	: width of the image in pixels
height	: height of the image in pixels
step	: the step in bytes (essential if the data are aligned)
data	: the buffer
channels	: number of channels in the image
data_type	: type of element stored
mat_type	: define where the buffer is stored (CPU or GPU memory)

Referenced by sl::zed::cvMat2slMat().

void sl::zed::Mat::setUp (int *width*, int *height*, int *channels*, DATA_TYPE *data_type*, MAT_TYPE *mat_type* = CPU)

Sets up the buffer metadata.

Parameters

width	: width of the image in pixels
height	: height of the image in pixels
channels	: number of channels in the image
data_type	: type of element stored
mat_type	: defines where the buffer is stored (CPU or GPU memory)

Note

> if the mat_type is CPU, the step is automatically set

Member Data Documentation

int sl::zed::Mat::channels

Referenced by sl::zed::slMat2cvMat().

sl::uchar∗ sl::zed::Mat::data

The pointer to the data, can also be casted to store float value

Referenced by sl::zed::slMat2cvMat().

DATA_TYPE sl::zed::Mat::data_type

Referenced by sl::zed::slMat2cvMat().

int sl::zed::Mat::height

Referenced by sl::zed::slMat2cvMat().

int sl::zed::Mat::step

MAT_TYPE sl::zed::Mat::type

int sl::zed::Mat::width

Referenced by sl::zed::slMat2cvMat().

sl::zed::resolution Struct Reference

Width and Height of each left and right image.

Collaboration diagram for sl::zed::resolution:

Public Member Functions

- resolution (int w_, int h_)

Public Attributes

- int width
- int height

Detailed Description

Width and Height of each left and right image.

Constructor & Destructor Documentation

sl::zed::resolution::resolution (int *w_*, int *h_*) `[inline]`

References height, and width.

Member Data Documentation

int sl::zed::resolution::height

Height of single image in pixels

Referenced by resolution().

int sl::zed::resolution::width

Width of single image in pixels

Referenced by resolution().

sl::zed::StereoParameters Struct Reference

Intrinsic parameters of each cameras, baseline and convergence (radians)

Collaboration diagram for sl::zed::StereoParameters:

Public Attributes

- float baseline
- float Ty
- float Tz
- float convergence
- float Rx
- float Rz
- CamParameters LeftCam
- CamParameters RightCam

Detailed Description

Intrinsic parameters of each cameras, baseline and convergence (radians)

Member Data Documentation

float sl::zed::StereoParameters::baseline

Distance between the 2 cameras in mm

float sl::zed::StereoParameters::convergence

Convergence between the 2 cameras in radians

CamParameters sl::zed::StereoParameters::LeftCam

Intrinsic parameters of the left camera

CamParameters sl::zed::StereoParameters::RightCam

Intrinsic parameters of the right camera

float sl::zed::StereoParameters::Rx

Rotation around X axis ("tilt"), will be affined by self-calibration - optional

float sl::zed::StereoParameters::Rz

Rotation around Z axis("roll"), will be affined by self-calibration - optional

float sl::zed::StereoParameters::Ty

float sl::zed::StereoParameters::Tz

sl::uchar3Struct Struct Reference

Collaboration diagram for sl::uchar3Struct:

Public Member Functions

- uchar3Struct (uchar c1_=0, uchar c2_=0, uchar c3_=0)
- void setValue (uchar c1_, uchar c2_=0, uchar c3_=0)

Public Attributes

- uchar c1
- uchar c2
- uchar c3

Constructor & Destructor Documentation

sl::uchar3Struct::uchar3Struct (uchar *c1_* = *0*, uchar *c2_* = *0*, uchar *c3_* = *0*) `[inline]`

References c1, c2, and c3.

Member Function Documentation

void sl::uchar3Struct::setValue (uchar *c1_*, uchar *c2_* = *0*, uchar *c3_* = *0*) `[inline]`

References c1, c2, and c3.

Member Data Documentation

uchar sl::uchar3Struct::c1

Referenced by setValue(), and uchar3Struct().

uchar sl::uchar3Struct::c2

Referenced by setValue(), and uchar3Struct().

uchar sl::uchar3Struct::c3

Referenced by setValue(), and uchar3Struct().

Index