

Reference Manual

Generated by Doxygen 1.7.4

Thu Oct 22 2015 01:40:59

Contents

1	DSA Konqueror and libframemanager	1
1.1	Introduction	1
1.2	License	2
1.3	Installation	2
1.3.1	Dependencies	2
2	Bug List	5
3	Class Index	7
3.1	Class Hierarchy	7
4	Class Index	9
4.1	Class List	9
5	File Index	11
5.1	File List	11
6	Class Documentation	13
6.1	Calibration Class Reference	13
6.1.1	Detailed Description	13
6.1.2	Constructor & Destructor Documentation	13
6.1.2.1	Calibration	13
6.1.3	Member Function Documentation	13
6.1.3.1	getTemperatureNoise	14
6.1.3.2	readTemperatureNoise	14
6.2	Camera Class Reference	14
6.2.1	Detailed Description	15

6.2.2	Constructor & Destructor Documentation	15
6.2.2.1	Camera	15
6.2.2.2	Camera	15
6.2.2.3	Camera	15
6.2.3	Member Function Documentation	15
6.2.3.1	getPosition	16
6.2.3.2	getView	16
6.2.3.3	move	16
6.2.3.4	moveX	16
6.2.3.5	moveY	16
6.2.3.6	moveZ	17
6.2.3.7	rotateX	17
6.2.3.8	rotateY	17
6.2.3.9	setPosition	17
6.2.3.10	setup	18
6.2.3.11	setView	18
6.3	Chebyshev Class Reference	18
6.3.1	Detailed Description	19
6.3.2	Constructor & Destructor Documentation	19
6.3.2.1	Chebyshev	19
6.3.3	Member Function Documentation	19
6.3.3.1	computeInvariants	19
6.3.3.2	computeReconstruction	19
6.3.3.3	initialize	20
6.4	Colormap Class Reference	20
6.4.1	Detailed Description	21
6.4.2	Member Function Documentation	21
6.4.2.1	createColorTable	21
6.4.2.2	getColorFromColormap	21
6.4.2.3	getColorFromTable	21
6.4.2.4	hsl2rgb	22
6.4.2.5	interpolate	22
6.4.2.6	limitColorRange	22
6.4.2.7	rgb2hsl	23

6.5	Controller Class Reference	23
6.5.1	Detailed Description	24
6.5.2	Constructor & Destructor Documentation	24
6.5.2.1	Controller	24
6.5.3	Member Function Documentation	24
6.5.3.1	connectDSA	24
6.5.3.2	connectSDH	25
6.5.3.3	disconnectDSA	25
6.5.3.4	disconnectSDH	25
6.5.3.5	getPreshape	25
6.5.3.6	grasp	25
6.5.3.7	graspReactive	26
6.5.3.8	isAvailableDSA	26
6.5.3.9	isAvailableSDH	26
6.5.3.10	isConnectedDSA	26
6.5.3.11	isConnectedSDH	27
6.5.3.12	loadProfile	27
6.6	Miniball::CoordAccessor< Pit_, Cit_ > Struct Template Reference . . .	27
6.7	Miniball::CoordAccessor< Pit_, Cit_* > Struct Template Reference . .	28
6.8	cSDHOptions Class Reference	28
6.8.1	Detailed Description	29
6.8.2	Constructor & Destructor Documentation	29
6.8.2.1	cSDHOptions	29
6.8.3	Member Function Documentation	30
6.8.3.1	OpenCommunication	30
6.8.3.2	Parse	30
6.9	Device Class Reference	31
6.9.1	Detailed Description	31
6.10	Ext Class Reference	31
6.10.1	Detailed Description	32
6.10.2	Member Function Documentation	33
6.10.2.1	addComport	33
6.10.2.2	Close	33
6.10.2.3	getComportDriver	33

6.10.2.4	IdentifyDevices	33
6.10.2.5	listComports	34
6.10.2.6	Open	34
6.10.2.7	probeSerial8250Comports	34
6.11	ExtException Class Reference	34
6.11.1	Detailed Description	35
6.12	FeatureExtraction Class Reference	35
6.12.1	Detailed Description	35
6.12.2	Constructor & Destructor Documentation	35
6.12.2.1	FeatureExtraction	35
6.12.3	Member Function Documentation	36
6.12.3.1	computeCentroid	36
6.12.3.2	computeMiniball	36
6.12.3.3	computeMiniball	36
6.12.3.4	computeMiniballCentroid	37
6.12.3.5	computeMiniballCentroid	37
6.12.3.6	computeMiniballPoints	37
6.12.3.7	computeMiniballPoints	38
6.12.3.8	computeMoments	38
6.12.3.9	computeStandardDeviation	38
6.13	FeatureExtractionWrapper Class Reference	39
6.13.1	Detailed Description	39
6.14	ForwardKinematics Class Reference	39
6.14.1	Detailed Description	40
6.14.2	Constructor & Destructor Documentation	40
6.14.2.1	ForwardKinematics	40
6.14.3	Member Function Documentation	40
6.14.3.1	computeTransformationMatrixPointOnSensorPlaneXYZ	40
6.14.3.2	computeTransformationMatrixTaxelXYZ	40
6.14.3.3	GetPointOnSensorPlaneXYZ	41
6.14.3.4	GetTaxelXYZ	41
6.14.3.5	setAngles	41
6.15	FrameGrabberDSA Class Reference	42
6.15.1	Detailed Description	42

6.15.2	Constructor & Destructor Documentation	42
6.15.2.1	FrameGrabberDSA	42
6.15.2.2	FrameGrabberDSA	42
6.15.2.3	~FrameGrabberDSA	43
6.15.3	Member Function Documentation	43
6.15.3.1	enableRecording	43
6.15.3.2	execute	43
6.15.3.3	finish	43
6.15.3.4	getFrameNumber	43
6.15.3.5	isCapturing	44
6.15.3.6	isRecording	44
6.15.3.7	isRunning	44
6.15.3.8	pause	44
6.15.3.9	resume	44
6.15.3.10	setFrameManager	44
6.15.3.11	setFramerate	45
6.15.3.12	start	45
6.15.3.13	suspendRecording	45
6.16	FrameGrabberSDH Class Reference	46
6.16.1	Detailed Description	46
6.16.2	Constructor & Destructor Documentation	46
6.16.2.1	FrameGrabberSDH	46
6.16.2.2	FrameGrabberSDH	47
6.16.2.3	~FrameGrabberSDH	47
6.16.3	Member Function Documentation	47
6.16.3.1	enableRecording	47
6.16.3.2	execute	47
6.16.3.3	finish	47
6.16.3.4	getFrameNumber	47
6.16.3.5	isCapturing	48
6.16.3.6	isRecording	48
6.16.3.7	isRunning	48
6.16.3.8	pause	48
6.16.3.9	pauseBlocking	48

6.16.3.10 resume	48
6.16.3.11 setFrameManager	49
6.16.3.12 setFrameRateJointAngles	49
6.16.3.13 setFrameRateTemperature	49
6.16.3.14 setJointAngle	49
6.16.3.15 setTemperature	49
6.16.3.16 start	50
6.16.3.17 suspendRecording	50
6.17 FrameManager Class Reference	50
6.17.1 Detailed Description	53
6.17.2 Constructor & Destructor Documentation	53
6.17.2.1 FrameManager	53
6.17.3 Member Function Documentation	53
6.17.3.1 addTSFrame	53
6.17.3.2 allocateTSFrame	53
6.17.3.3 computeSlip	53
6.17.3.4 computeSlipLive	54
6.17.3.5 convertCellIndex	54
6.17.3.6 convertCellIndex	54
6.17.3.7 createJointAngleMapping	54
6.17.3.8 createSelectedCellsIdx	55
6.17.3.9 createTemperatureMapping	55
6.17.3.10 cropToFrames	55
6.17.3.11 decodeTSFrame	55
6.17.3.12 deleteTSFrame	56
6.17.3.13 disableSlipDetection	56
6.17.3.14 enableSlipDetection	56
6.17.3.15 encodeTSFrame	56
6.17.3.16 float2hex	57
6.17.3.17 getCorrespondingJointAngle	57
6.17.3.18 getCorrespondingTemperature	57
6.17.3.19 getCurrentFilteredFrame	57
6.17.3.20 getCurrentFrame	58
6.17.3.21 getCurrentFrameID	58

6.17.3.22	getCurrentJointAngleFrame	58
6.17.3.23	getDynamicMask	58
6.17.3.24	getFilteredFrame	58
6.17.3.25	getFilteredTexel	59
6.17.3.26	getFilteredTexel	59
6.17.3.27	getFrame	59
6.17.3.28	getFrameCountJointAngles	60
6.17.3.29	getFrameCountTemperature	60
6.17.3.30	getFrameCountTS	60
6.17.3.31	getFrameProcessor	60
6.17.3.32	getJointAngleFrame	60
6.17.3.33	getJointAngleFrameAvailable	61
6.17.3.34	getMatrixInfo	61
6.17.3.35	getNumCells	61
6.17.3.36	getNumMatrices	61
6.17.3.37	getNumSelectedCells	61
6.17.3.38	getProfileName	61
6.17.3.39	getSelection	62
6.17.3.40	getSensorInfo	62
6.17.3.41	getSlipDetectionState	62
6.17.3.42	getSlipDetectionState	62
6.17.3.43	getSlipDetector	62
6.17.3.44	getSlipLive	63
6.17.3.45	getSlipLiveBinary	63
6.17.3.46	getStaticMask	63
6.17.3.47	getTemperatureFrame	63
6.17.3.48	getTexel	64
6.17.3.49	getTexel	64
6.17.3.50	getTSFrameAvailable	64
6.17.3.51	hex2float	64
6.17.3.52	isConnectedDSA	65
6.17.3.53	isConnectedSDH	65
6.17.3.54	isSelected	65
6.17.3.55	loadFrames	65

6.17.3.56 loadProfile	66
6.17.3.57 print_tree	66
6.17.3.58 print_tree	66
6.17.3.59 printJointAngleFrame	66
6.17.3.60 printTemperatureFrame	67
6.17.3.61 printTSMatrices	67
6.17.3.62 printTSMatrix	67
6.17.3.63 queryDSAInfo	67
6.17.3.64 requestJointAngleFrame	68
6.17.3.65 requestTemperatureFrame	68
6.17.3.66 resetOffline	68
6.17.3.67 resetOnline	68
6.17.3.68 selectCell	68
6.17.3.69 setCurrentFrameID	69
6.17.3.70 setDSA	69
6.17.3.71 setDynamicMask	69
6.17.3.72 setFilterBilateral	69
6.17.3.73 setFilterGaussian	70
6.17.3.74 setFilterMedian	70
6.17.3.75 setFilterMedian3D	70
6.17.3.76 setFilterMorphological	71
6.17.3.77 setFilterNone	71
6.17.3.78 setFrameGrabberDSA	71
6.17.3.79 setFrameGrabberSDH	71
6.17.3.80 setJointAngleFrameAvailable	72
6.17.3.81 setLiveFrame	72
6.17.3.82 setSDH	72
6.17.3.83 setSensitivity	72
6.17.3.84 setSlipReferenceFrame	73
6.17.3.85 setSlipReferenceFrameLive	73
6.17.3.86 setSlipThresholdConsecutive	73
6.17.3.87 setSlipThresholdReference	73
6.17.3.88 setThreshold	74
6.17.3.89 setTSFrameAvailable	74

6.17.3.90	slipResultConsumer	74
6.17.3.91	slipResultProducer	75
6.17.3.92	storeFrames	75
6.17.3.93	storeProfile	75
6.17.3.94	storeProfileSelection	75
6.18	FrameManagerWrapper Class Reference	76
6.18.1	Detailed Description	77
6.19	FrameProcessor Class Reference	77
6.19.1	Detailed Description	78
6.19.2	Constructor & Destructor Documentation	78
6.19.2.1	FrameProcessor	78
6.19.3	Member Function Documentation	78
6.19.3.1	applyFilter	78
6.19.3.2	calcCharacteristics	78
6.19.3.3	getAverage	79
6.19.3.4	getFilterType	79
6.19.3.5	getMatrixAverage	79
6.19.3.6	getMatrixMax	79
6.19.3.7	getMatrixMin	80
6.19.3.8	getMatrixNumActiveCells	80
6.19.3.9	getMax	80
6.19.3.10	getMin	80
6.19.3.11	getNumActiveCells	81
6.19.3.12	setFilterBilateral	81
6.19.3.13	setFilterGaussian	81
6.19.3.14	setFilterMedian	82
6.19.3.15	setFilterMedian3D	82
6.19.3.16	setFilterNone	82
6.19.3.17	setFilterOpening	82
6.19.3.18	setFrameManager	83
6.20	guiChart Class Reference	83
6.20.1	Detailed Description	83
6.20.2	Member Function Documentation	84
6.20.2.1	initDataset	84

6.20.2.2	updateDataset	84
6.21	guiController Class Reference	84
6.21.1	Detailed Description	87
6.21.2	Member Function Documentation	87
6.21.2.1	on_button_threshold_clicked	87
6.22	guiGraph Class Reference	87
6.22.1	Detailed Description	88
6.22.2	Member Function Documentation	88
6.22.2.1	on_button_press_event	88
6.22.2.2	on_button_release_event	88
6.22.2.3	on_motion_notify_event	88
6.22.2.4	setMarkerPosition	88
6.23	guiMain Class Reference	89
6.23.1	Detailed Description	91
6.23.2	Member Function Documentation	91
6.23.2.1	embedPython	91
6.23.2.2	on_menu_load_profile	91
6.23.2.3	on_menu_new_profile	91
6.23.2.4	resetGUIOffline	91
6.23.2.5	updateGUIOffline	91
6.24	guiRenderer Class Reference	91
6.24.1	Detailed Description	92
6.25	guiRenderer2D Class Reference	93
6.25.1	Detailed Description	93
6.26	guiRenderer3D Class Reference	94
6.26.1	Detailed Description	94
6.26.2	Member Function Documentation	95
6.26.2.1	on_button_press_event	95
6.26.2.2	on_button_release_event	95
6.26.2.3	on_configure_event	95
6.26.2.4	on_expose_event	95
6.26.2.5	on_motion_notify_event	95
6.26.2.6	on_realize	95
6.27	guiScreenshot Class Reference	95

6.27.1 Detailed Description	96
6.28 guiSeekBar Class Reference	96
6.28.1 Detailed Description	97
6.28.2 Member Function Documentation	97
6.28.2.1 on_signal_timeout	97
6.28.2.2 setSliderPosition	97
6.29 guiSlipDetection Class Reference	97
6.29.1 Detailed Description	98
6.30 guiSlipDetectionMultiPlot Class Reference	98
6.30.1 Detailed Description	99
6.31 guiTools Class Reference	99
6.31.1 Detailed Description	100
6.32 guiTreeView Class Reference	100
6.32.1 Detailed Description	100
6.33 HSL Class Reference	100
6.33.1 Detailed Description	101
6.33.2 Constructor & Destructor Documentation	101
6.33.2.1 HSL	101
6.34 JointAngleFrame Struct Reference	101
6.34.1 Detailed Description	102
6.35 matrixInfo Struct Reference	102
6.35.1 Detailed Description	102
6.36 Miniball::Miniball< CoordAccessor > Class Template Reference	103
6.37 NiceScale Class Reference	103
6.37.1 Detailed Description	104
6.38 NumPyArrayData< T > Class Template Reference	104
6.39 Orientation Class Reference	104
6.39.1 Detailed Description	105
6.40 OrientationTrajectory Class Reference	105
6.40.1 Detailed Description	105
6.41 RGB Class Reference	105
6.41.1 Detailed Description	106
6.41.2 Constructor & Destructor Documentation	106
6.41.2.1 RGB	106

6.42	Rotation Class Reference	106
6.42.1	Detailed Description	107
6.42.2	Constructor & Destructor Documentation	107
6.42.2.1	Rotation	107
6.42.3	Member Function Documentation	107
6.42.3.1	computeRotation	107
6.42.3.2	rotationFromMoments	107
6.42.3.3	setReferenceFrame	107
6.43	RowData Class Reference	108
6.44	Ext::sControllerInfo Struct Reference	108
6.44.1	Detailed Description	109
6.45	sensorInfo Struct Reference	109
6.45.1	Detailed Description	109
6.46	SlipDetector Class Reference	109
6.46.1	Detailed Description	110
6.46.2	Constructor & Destructor Documentation	110
6.46.2.1	SlipDetector	110
6.46.3	Member Function Documentation	110
6.46.3.1	computeSlip	110
6.46.3.2	reset	111
6.46.3.3	setReferenceFrame	111
6.46.3.4	setReferenceFrameRotation	111
6.46.3.5	setReferenceFrameRotation	111
6.46.3.6	setReferenceFrameTranslation	112
6.46.3.7	setReferenceFrameTranslation	112
6.47	slipResult Struct Reference	112
6.47.1	Detailed Description	113
6.48	SlipVectorLive Class Reference	113
6.48.1	Detailed Description	114
6.49	SlipVectorTrajectory Class Reference	114
6.49.1	Detailed Description	114
6.50	TemperatureFrame Struct Reference	114
6.50.1	Detailed Description	115
6.51	TemperatureNoise Struct Reference	115

6.51.1 Detailed Description	115
6.52 TimeSeriesDataset Struct Reference	115
6.53 TimestampComparator< Frame > Struct Template Reference	116
6.53.1 Detailed Description	116
6.54 Translation Class Reference	116
6.54.1 Detailed Description	117
6.54.2 Constructor & Destructor Documentation	117
6.54.2.1 Translation	117
6.54.3 Member Function Documentation	117
6.54.3.1 computeSlip	117
6.54.3.2 computeSlipReference	117
6.54.3.3 init	118
6.54.3.4 setReferenceFrame	118
6.55 TSFrame Struct Reference	118
6.55.1 Detailed Description	119
7 File Documentation	121
7.1 sdhoptions.cpp File Reference	121
7.1.1 Detailed Description	121
7.1.2 Copyright	122
7.1.3 Define Documentation	122
7.1.3.1 XSTRINGIFY	122
7.2 sdhoptions.h File Reference	122
7.2.1 Detailed Description	123
7.2.2 General file information	123
7.2.3 Copyright	123
7.2.4 Define Documentation	123
7.2.4.1 SDHUSAGE_DEFAULT	123
7.3 utils.h File Reference	123
7.3.1 Detailed Description	124

Chapter 1

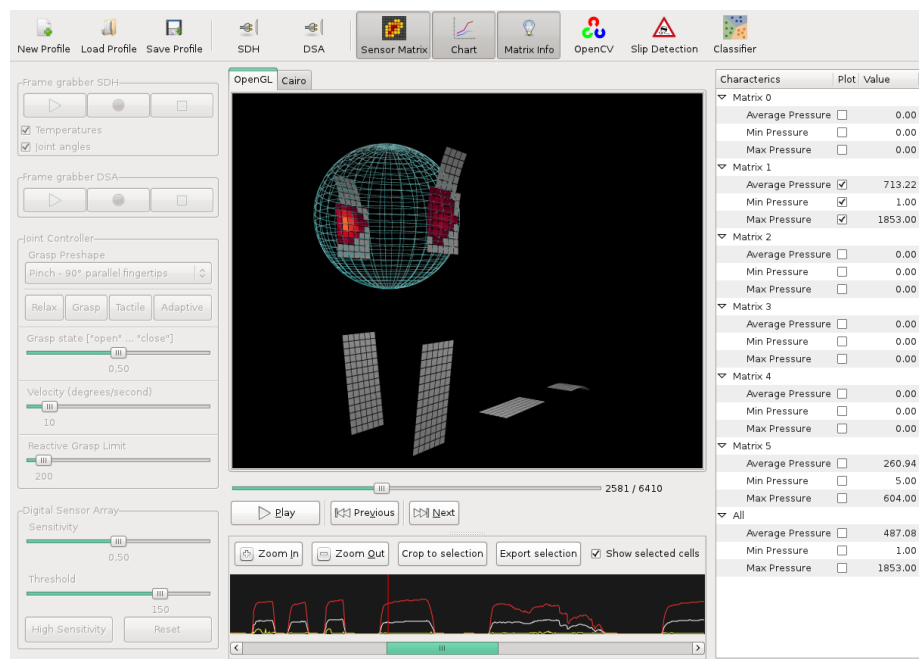
DSA Konqueror and libframemanager

Author

Peter Kiechle

Date

2015



Graphical user interface

1.1 Introduction

This software package was created during the course of my master thesis "Evaluation of Tactile Sensors" at the Intelligent and Interactive Systems group of the Institute of

Computer Science at the University of Innsbruck. My supervision was Professor Justus Piater, Ph.D.

Libframemanager consists of a frame manager and two separate frame grabbers. One grabber requests and processes the temperatures and axis angles received from the SDH-2 while the other captures the tactile sensor frames from the DSA controller. DSA Konqueror, not to be confused with DSA Explorer by Weiss Robotics, is a graphical user interface or GUI to control the SDH-2, i.e. perform grasps and display the tactile sensor readings. It simplifies the capturing and recording of pressure profiles and offers real-time visualization as well as offline processing. In addition, there are Python bindings to the most frequently used functionalities. See master thesis for details.

1.2 License

DSA-Konqueror and libframemanager Copyright (C) 2015 Peter Kiechle, peter@kiechle-pfronten.de

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>

1.3 Installation

```
cd build
cmake ..
make VERBOSE=1
```

1.3.1 Dependencies

See "dependencies" folder

C++:

- SDH Library (patched version, see dependencies/dsa_patch)
- OpenCV
- Eigen 3
- Boost
- Boost.NumPy (extension for Boost.Python that adds NumPy support, see dependencies/Boost.NumPy)

- Gtkmm 2.4 (GUI)
- GtkGLExtmmb (OpenGL extension)

Python 2.7:

- numpy
- scipy
- sklearn
- cairo
- PIL

Chapter 2

Bug List

Member `SDHUSAGE_DEFAULT` When compiled with VCC then the macros `WITH_ESD_CAN` / `WITH_PEAK_CAN` used above are not available since these are defined in the VCC project settings of the SDHLibrary VCC-Project. Therefore the value of `SDHUSAGE_DEFAULT` is incorrect and thus the `cSDHOptions` will display an incomplete usage string when called with `-h/--help`.

Workaround: use the online help contained in the doxygen documentation: [Online help of demonstration programs](#)

Chapter 3

Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Calibration	13
Camera	14
Chebyshev	18
Colormap	20
Controller	23
Miniball::CoordAccessor< Pit_, Cit_ >	27
Miniball::CoordAccessor< Pit_, Cit_ * >	28
cSDHOptions	28
Device	31
Ext	31
ExtException	34
FeatureExtraction	35
FeatureExtractionWrapper	39
ForwardKinematics	39
FrameGrabberDSA	42
FrameGrabberSDH	46
FrameManager	50
FrameManagerWrapper	76
FrameProcessor	77
guiChart	83
guiController	84
guiGraph	87
guiMain	89
guiRenderer	91
guiRenderer2D	93
guiRenderer3D	94
guiScreenshot	95
guiSeekBar	96
guiSlipDetection	97

guiSlipDetectionMultiPlot	98
guiTools	99
guiTreeView	100
HSL	100
JointAngleFrame	101
matrixInfo	102
Miniball::Miniball< CoordAccessor >	103
NiceScale	103
NumPyArrayData< T >	104
Orientation	104
OrientationTrajectory	105
RGB	105
Rotation	106
RowData	108
Ext::sControllerInfo	108
sensorInfo	109
SlipDetector	109
slipResult	112
SlipVectorLive	113
SlipVectorTrajectory	114
TemperatureFrame	114
TemperatureNoise	115
TimeSeriesDataset	115
TimestampComparator< Frame >	116
Translation	116
TSFrame	118

Chapter 4

Class Index

4.1 Class List

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

Calibration (Reads an XML file containing the calibration parameters for the High-Sensitivity Mode)	13
Camera (Simple camera class similar to the ones used in first-person shooter video games. Based on the OpenGL coordinate system and glu-LookAt())	14
Chebyshev (Computes discrete Chebyshev polynomials and thereon based image moments (translation- and rotation invariant))	18
Colormap (Manages colors and colormaps)	20
Controller (Implements the controller of the GUI's model-view-controller (MVC) pattern)	23
Miniball::CoordAccessor< Pit_, Cit_ >	27
Miniball::CoordAccessor< Pit_, Cit_* >	28
cSDHOptions (Class for command line option parsing holding option parsing results)	28
Device (Simple structure to manage the device, its type and serial number) . .	31
Ext (Extension to the SDH-Library to automatically query all available com-ports for connected DSA / SDH devices)	31
ExtException (Derived exception class for low-level DSA related exceptions) .	34
FeatureExtraction (Unifies the computation of features, in detail: The standard deviation, Chebyshev moments and minimal bounding sphere) . . .	35
FeatureExtractionWrapper (Defines Boost.Python wrappers around the FeatureExtraction class. See Python examples for usage)	39
ForwardKinematics (Implements forward kinematics for the tactile sensors) . .	39
FrameGrabberDSA (A frame grabber/recorder class for tactile sensor based on boost thread)	42
FrameGrabberSDH (A frame grabber/recorder class for joint angles and temperatures based on boost thread)	46
FrameManager (The heart of this project)	50

FrameManagerWrapper (Defines Boost.Python wrappers around the Frame- Manage class. See Python examples for usage)	76
FrameProcessor (Manages temporal, spatial and spatio-temporal filtering) . .	77
guiChart (The chart containing the the zoom, crop and export buttons as well as the graph. Manages the dataset before it is displayed in the graph)	83
guiController (GUI to control the SDH-2)	84
guiGraph (The graph)	87
guiMain (The main window. Inherits from Gtk::Window)	89
guiRenderer (Base class for guiRenderer2D and guiRenderer3D)	91
guiRenderer2D (Renders visualization of tactile sensor profiles with the help of Cairo, a vector graphics library)	93
guiRenderer3D (Renders visualization of tactile sensor profiles in OpenGL) .	94
guiScreenshot (Take screen shot GUI)	95
guiSeekBar (The seekbar)	96
guiSlipDetection (The Slip-detection control GUI)	97
guiSlipDetectionMultiPlot (Combines individual widgets)	98
guiTools (Dialog for OpenCV filters)	99
guiTreeView (Tree view of matrix characteristics. Follows the MVC pattern) .	100
HSL (Simple HSL color management class. Hue, Saturation and Luminance are doubles in the range [0.0, 1.0])	100
JointAngleFrame (Joint angles in accordance with SDHLibrary)	101
matrixInfo (Individual sensor matrices (see dsa.h))	102
Miniball::Miniball< CoordAccessor >	103
NiceScale (Pretty axis tick labels. Graphics Gems, Volume 1 by Andrew S. Glassner)	103
NumPyArrayData< T >	104
Orientation (Visualizes current orientation)	104
OrientationTrajectory (Visualizes rotation trajectory)	105
RGB (Simple RGB color management class. Red Green Blue are floats in the range [0.0, 1.0])	105
Rotation (Implements the rotational slip detection based on the principal axis method)	106
RowData	108
Ext::sControllerInfo (A data structure describing the controller info about the remote DSACON32m controller)	108
sensorInfo (Tactile sensor Controller info (see dsa.h))	109
SlipDetector (Combined Slip-Detection class (Translation + Rotation))	109
slipResult (Final return type struct: Boost tuples are limited to 10 elements, so this is why..)	112
SlipVectorLive (Visualizes current slip vector)	113
SlipVectorTrajectory (Visualizes slip vector trajectory)	114
TemperatureFrame (Temperatures 0-6: close to axes motors, Temperature 7: FPGA, Temperature 8: Printed circuit board)	114
TemperatureNoise (A data structure containing the linear regression parame- ters as well as the RMS Error of the prediction band)	115
TimeSeriesDataset	115
TimestampComparator< Frame > (Timestamp comparator functor)	116
Translation (Implements the translational slip detection based on tracking the convolution matrix's center of gravity)	116
TSFrame (Tactile sensor frame)	118

Chapter 5

File Index

5.1 File List

Here is a list of all documented files with brief descriptions:

calibration.h	??
camera.h	??
chebyshevMoments.h	??
colormap.h	??
controller.h	??
extension.h	??
featureExtraction.h	??
forwardKinematics.h	??
framegrabberDSA.h	??
framegrabberSDH.h	??
framemanager.h	??
frameprocessor.h	??
guiChart.h	??
guiController.h	??
guiGraph.h	??
guiMain.h	??
guiRenderer.h	??
guiRenderer2D.h	??
guiRenderer3D.h	??
guiScreenshot.h	??
guiSeekBar.h	??
guiSlipDetection.h	??
guiSlipDetectionMultiPlot.h	??
guiTools.h	??
guiTreeView.h	??
Miniball.hpp	??
NumPyArrayData.h	??
sdhoptions.cpp (Implementation of a class to parse common SDH related command line options)	121

sdhoptions.h (Implementation of a class to parse common SDH related com- mand line options)	122
slipdetection.h	??
utils.h	123

Chapter 6

Class Documentation

6.1 Calibration Class Reference

Reads an XML file containing the calibration parameters for the High-Sensitivity Mode.

```
#include <calibration.h>
```

Public Member Functions

- [Calibration](#) ()
- void [readTemperatureNoise](#) (const std::string &filename)
- [TemperatureNoise](#) & [getTemperatureNoise](#) (uint matrixID)

6.1.1 Detailed Description

Reads an XML file containing the calibration parameters for the High-Sensitivity Mode.

Note

Regression parameters were determined with "noise-temperature_calibration.py"
Values could be tweaked manually if wear and tear changes the sensor's behavior

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Calibration::Calibration ()

Constructor that immediately imports the file "calibration_temperature_noise.xml"

6.1.3 Member Function Documentation

6.1.3.1 TemperatureNoise & Calibration::getTemperatureNoise (uint *matrixID*)

Returns the calibrated slope, intercept and RMSE of the specified sensor matrix

Parameters

<i>matrixID</i>	Specified sensor matrix
-----------------	-------------------------

Returns

The linear regression parameters

6.1.3.2 void Calibration::readTemperatureNoise (const std::string & *filename*)

Reads calibration parameter file

Parameters

<i>filename</i>	The XML file containing the calibrated parameters
-----------------	---

Returns

void

The documentation for this class was generated from the following files:

- calibration.h
- calibration.cpp

6.2 Camera Class Reference

Simple camera class similar to the ones used in first-person shooter video games. Based on the OpenGL coordinate system and gluLookAt().

```
#include <camera.h>
```

Public Member Functions

- [Camera](#) (double x, double y, double z)
- [Camera](#) (Eigen::Vector3d pos)
- [Camera](#) (Eigen::Vector3d pos, Eigen::Vector3d view)
- void [setPosition](#) (Eigen::Vector3d pos)
- void [setView](#) (Eigen::Vector3d view)
- Eigen::Vector3d [getPosition](#) ()
- Eigen::Vector3d [getView](#) ()
- void [rotateX](#) (double angle)
- void [rotateY](#) (double angle)

- void [moveX](#) (double distance)
- void [moveY](#) (double distance)
- void [moveZ](#) (double distance)
- void [move](#) (Eigen::Vector3d &direction)
- void [setup](#) ()

6.2.1 Detailed Description

Simple camera class similar to the ones used in first-person shooter video games. Based on the OpenGL coordinate system and `gluLookAt()`.

Note

There is a problem when `upVector` is colinear to `viewVector`. To prevent the so called *gimbal lock*, use a camera class based on Quaternions instead.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 Camera::Camera (double *x*, double *y*, double *z*)

Delegating constructor

Parameters

<i>x,y,z</i>	Coordinates of the initial position in 3D space.
--------------	--

6.2.2.2 Camera::Camera (Eigen::Vector3d *pos*)

Delegating constructor

Parameters

<i>pos</i>	Coordinate vector of the initial position in 3D space.
------------	--

6.2.2.3 Camera::Camera (Eigen::Vector3d *pos*, Eigen::Vector3d *view*)

Delegating constructor

Parameters

<i>pos</i>	Coordinate vector of the initial position in 3D space.
<i>view</i>	Initial view vector.

6.2.3 Member Function Documentation

6.2.3.1 Eigen::Vector3d Camera::getPosition () [inline]

Returns the current position vector.

Returns

The position vector.

6.2.3.2 Eigen::Vector3d Camera::getView () [inline]

Returns the current view vector.

Returns

The view vector.

6.2.3.3 void Camera::move (Eigen::Vector3d & direction)

Computes the new position vector based on the given direction vector.

Parameters

<i>direction</i>	Distance in xyz-direction (OpenGL coordinate system)
------------------	--

Returns

void

6.2.3.4 void Camera::moveX (double distance)

Moves sideways and computes the new position vector.

Parameters

<i>distance</i>	Distance in x-direction (OpenGL coordinate system)
-----------------	--

Returns

void

6.2.3.5 void Camera::moveY (double distance)

Moves forward/backward and computes the new position vector.

Parameters

<i>distance</i>	Distance in y-direction (OpenGL coordinate system)
-----------------	--

Returns

void

6.2.3.6 void Camera::moveZ (double *distance*)

Moves upward/downward and computes the new position vector.

Parameters

<i>distance</i>	Distance in z-direction (OpenGL coordinate system)
-----------------	--

Returns

void

6.2.3.7 void Camera::rotateX (double *angle*)

Computes the new pitch, i.e. the rotation around the rightVector (look up/down).

Note

: We won't compute the new up vector since this may lead to a gimbal lock.

Parameters

<i>angle</i>	The rotation angle.
--------------	---------------------

Returns

void

6.2.3.8 void Camera::rotateY (double *angle*)

Computes the new Yaw, i.e. the rotation around the upVector (look left/right).

Parameters

<i>angle</i>	The rotation angle.
--------------	---------------------

Returns

void

6.2.3.9 void Camera::setPosition (Eigen::Vector3d *pos*)

(Re)sets the current position vector.

Parameters

<i>pos</i>	Coordinate vector of the position in 3D space.
------------	--

Returns

void

6.2.3.10 void Camera::setup ()

Final step: executes gluLookAt().

Returns

void

6.2.3.11 void Camera::setView (Eigen::Vector3d view)

(Re)sets the current view vector.

Parameters

<i>view</i>	View vector.
-------------	--------------

Returns

void

The documentation for this class was generated from the following files:

- camera.h
- camera.cpp

6.3 Chebyshev Class Reference

Computes discrete [Chebyshev](#) polynomials and thereon based image moments (translation- and rotation invariant).

```
#include <chebyshevMoments.h>
```

Public Member Functions

- [Chebyshev](#) ()
- void [initialize](#) (cv::Mat &frame, int pmax_rot)
- void [computeInvariants](#) (cv::Mat &frame, int pmax, array_type &T_pq_doubleprime)
- void [computeReconstruction](#) (cv::Mat &frame)

6.3.1 Detailed Description

Computes discrete [Chebyshev](#) polynomials and thereon based image moments (translation- and rotation invariant).

Note

See Chapter 6.6, [Chebyshev](#) moments of my thesis for used formulas and further details. Based on publications by Mukundan et al. as well as "Moments and Moment Invariants in Pattern Recognition", Jan Flusser, Barbara Zitova and Tomas Suk. The look-up table approach is following: "Symmetric image recognition by Tchebichef moment invariants", Hui Zhang, Xiubing Dai, Pei Sun, Hongqing Zhu, and Huazhong Shu.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 `Chebyshev::Chebyshev () [inline]`

Constructor

6.3.3 Member Function Documentation

6.3.3.1 `void Chebyshev::computeInvariants (cv::Mat & frame, int pmax, array_type & T_pq_doubleprime)`

Computes discrete translation and rotation invariant [Chebyshev](#) moments

Note

In order to compute rotation invariant moments of order p, "normal" moments of order $2*(pmax_rot-1)+1$ are needed.

Parameters

in	<i>frame</i>	Reference to the tactile image.
in	<i>pmax</i>	Maximum invariant moment order.
out	<i>T_pq_doubleprime</i>	Reference to final rotation and translation invariant Chebyshev moments.

Returns

void

6.3.3.2 `void Chebyshev::computeReconstruction (cv::Mat & frame)`

Performs a reconstruction from moments assuming [computeInvariants\(\)](#) has been executed before.

Parameters

out	frame	Reference to the reconstructed image.
-----	-------	---------------------------------------

Returns

void

6.3.3.3 void Chebyshev::initialize (cv::Mat & frame, int pmax_rot)

Initializes look-up tables that stay the same as long as the image size and moment order does not change.

Note

In order to compute rotation invariant moments of order p, "normal" moments of order $2*(pmax_rot-1)+1$ are needed.

Parameters

in	frame	Reference to the tactile image.
in	pmax_rot	Maximum rotation invariant moment order.

Returns

void

The documentation for this class was generated from the following files:

- chebyshevMoments.h
- chebyshevMoments.cpp

6.4 Colormap Class Reference

Manages colors and colormaps.

```
#include <colormap.h>
```

Public Member Functions

- float [interpolate](#) (float value, float f0, float f1, float x0, float x1)
Linear interpolation.
- [RGB hsl2rgb](#) (HSL &hsl)
Conversion from [HSL](#) to [RGB](#) color space.
- [HSL rgb2hsl](#) (RGB &rgb)
Conversion from [RGB](#) to [HSL](#) color space.
- float [limitColorRange](#) (float value, float low, float high)

- [RGB](#) `getColorFromColormap` (vector< [RGB](#) > &colormap, float value, InterpMethod interpolationMethod=HSL_INTERPOLATION)
- void `createColorTable` (ColorGradient colorGradient, int nColors)
- [RGB](#) & `getColorFromTable` (int position)

6.4.1 Detailed Description

Manages colors and colormaps.

6.4.2 Member Function Documentation

6.4.2.1 void Colormap::createColorTable (ColorGradient *colorGradient*, int *nColors*)

Creates a predefined colormap.

Parameters

<i>colorGradient</i>	Name (enum) of the actual colormap.
<i>nColors</i>	Number of Colors.

Returns

void

6.4.2.2 [RGB](#) Colormap::getColorFromColormap (vector< [RGB](#) > & *colormap*, float *value*, InterpMethod *interpolationMethod* = HSL_INTERPOLATION)

Interpolates color between neighboring colors of a colormap.

Parameters

<i>colormap</i>	The colormap, a vector of colors.
<i>value</i>	The desired color in the range [0.0, 1.0].
<i>interpolationMethod</i>	The interpolation method, HSL_INTERPOLATION or RGB_INTERPOLATION.

Returns

The resulting color in [RGB](#).

6.4.2.3 [RGB](#) & Colormap::getColorFromTable (int *position*)

Returns entry of the colormap at specified position.

Parameters

<i>position</i>	Position of the color in the colormap.
-----------------	--

Returns

A reference to the requested [RGB](#) struct.

6.4.2.4 RGB Colormap::hsl2rgb (HSL & *hsl*)

Conversion from [HSL](#) to [RGB](#) color space.

Range: h[0..1], s[0..1], l[0..1], r[0..1], g[0..1], b[0..1]

Parameters

<i>hsl</i>	Reference to HSL struct
------------	---

Returns

Resulting [RGB](#) struct.

6.4.2.5 float Colormap::interpolate (float *value*, float *f0*, float *f1*, float *x0*, float *x1*)

Linear interpolation.

| f1 |-----* | . | | ? |-----X | | . | f0 |---* | | . |_____|_____|_____|_____. x0
value x1

Note

x0 has to be smaller than *x1*

Parameters

<i>value</i>	Value between <i>x0</i> and <i>x1</i>
<i>f0,f1</i>	Function values at supporting points.
<i>x0,x1</i>	Supporting points.

Returns

Interpolated function value.

6.4.2.6 float Colormap::limitColorRange (float *value*, float *low*, float *high*)

Clamps value to [low..high].

Parameters

<i>value</i>	The value to be clamped.
<i>low</i>	The lower limit.
<i>high</i>	The upper limit.

Returns

The clamped value.

6.4.2.7 HSL Colormap::rgb2hsl (RGB & rgb)

Conversion from [RGB](#) to [HSL](#) color space.

Range: h[0..1], s[0..1], l[0..1], r[0..1], g[0..1], b[0..1]

Parameters

<i>rgb</i>	Reference to RGB struct.
------------	--

Returns

Resulting [HSL](#) struct.

The documentation for this class was generated from the following files:

- colormap.h
- colormap.cpp

6.5 Controller Class Reference

Implements the controller of the GUI's model–view–controller (MVC) pattern.

```
#include <controller.h>
```

Public Member Functions

- [Controller](#) (int argc, char *argv[])
- bool [isAvailableSDH](#) ()
- bool [isAvailableDSA](#) ()
- void [connectSDH](#) ()
- void [connectDSA](#) ()
- void [disconnectSDH](#) ()
- void [disconnectDSA](#) ()
- bool [isConnectedSDH](#) ()
- bool [isConnectedDSA](#) ()
- void [loadProfile](#) (const std::string &filename)
- cDSA * [getDSA](#) ()
- *Getter/Setter.*
- cSDH * [getSDH](#) ()
- [FrameManager](#) * [getFrameManager](#) ()
- [FrameGrabberDSA](#) * [getFrameGrabberDSA](#) ()
- [FrameGrabberSDH](#) * [getFrameGrabberSDH](#) ()

- [guiRenderer](#) * **getRenderer** ()
- void **setRenderer** ([guiRenderer](#) *r)
- boost::filesystem::path **getProfilePath** ()
- std::string **getProfilePathName** ()
- std::string **getProfileDirectory** ()
- std::string **getProfileName** ()
- std::string **getProfileBaseName** ()
- std::string **getProfileExtension** ()
- [Calibration](#) & **getCalibration** ()
- vector< double > **getPreshape** (int graspID, double closeRatio)
- void [grasp](#) (int graspID, double closeRatio, double velocity)
- boost::tuple< bool, float > [graspReactive](#) (int graspID, double velocity, double limit)

6.5.1 Detailed Description

Implements the controller of the GUI's model–view–controller (MVC) pattern.

Note

In case the GUI is not of interest for your project, this class may still serve as an example application.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 Controller::Controller (int argc, char * argv[])

Constructor automatically initializes a whole lot of different subsystems

Note

You will have to change the hard-coded serial numbers according to your SDH-2

6.5.3 Member Function Documentation

6.5.3.1 void Controller::connectDSA ()

Creates an instance of the DSA controller and tries to open the communication.

Returns

void

6.5.3.2 void Controller::connectSDH ()

Creates an instance of the SHD-2 and tries to open the communication.

Returns

void

6.5.3.3 void Controller::disconnectDSA ()

Disconnects the DSA controller and deletes the instance.

Returns

void

6.5.3.4 void Controller::disconnectSDH ()

Disconnects the SDH-2 and deletes the instance.

Returns

void

6.5.3.5 vector< double > Controller::getPreshape (int *graspID*, double *closeRatio*)

Gets joint angles of preshaped grasp

Parameters

<i>graspID</i>	The graspID, see source for description.
<i>closeRatio</i>	Open/Close ratio in the range [0.0, 1.0].

Returns

void

6.5.3.6 void Controller::grasp (int *graspID*, double *closeRatio*, double *velocity*)

Reimplementation of the SDHLibrary's grasping routine.

This function is non-blocking and continuously queries the axis states itself. Thus, joint angles can be recorded while grasping. In contrast, the original function executes the grasp and only returns when the grasp is completed.

Parameters

<i>graspID</i>	The graspID, see source for description.
----------------	--

<i>closeRatio</i>	Open/Close ratio in the range [0.0, 1.0].
<i>velocity</i>	The grasping speed.

Returns

void

6.5.3.7 `boost::tuple< bool, float > Controller::graspReactive (int graspID, double velocity, double limit)`

Reactive grasping routine similar to [grasp\(\)](#) that also takes the tactile sensors into account.

This function is non-blocking and continuously queries the axis states itself. Thus, joint angles can be recorded while grasping. In contrast, the original function executes the grasp and only returns when the grasp is completed.

Parameters

<i>graspID</i>	The graspID, see source for description.
<i>closeRatio</i>	Open/Close ratio in the range [0.0, 1.0].
<i>velocity</i>	The grasping speed.
<i>velocity</i>	Maximum sensor value limit.

Returns

Tuple of the state of the grasp and maximum sensor value.

6.5.3.8 `bool Controller::isAvailableDSA ()`

Reports if a the DSA controller was found.

Returns

The state.

6.5.3.9 `bool Controller::isAvailableSDH ()`

Reports if the SDH-2 was found.

Returns

The state.

6.5.3.10 `bool Controller::isConnectedDSA ()`

Reports if a the DSA controller is connected and ready to go.

Returns

The state.

6.5.3.11 bool Controller::isConnectedSDH ()

Reports if a the SDH-2 is connected and ready to go.

Returns

The state.

6.5.3.12 void Controller::loadProfile (const std::string & *filename*)

Opens the specified *.dsa file and loads it's contents.

Parameters

<i>filename</i>	The filename of the *.dsa profile
-----------------	-----------------------------------

Returns

void

The documentation for this class was generated from the following files:

- controller.h
- controller.cpp

6.6 Miniball::CoordAccessor< Pit_, Cit_ > Struct Template Reference

Public Types

- typedef Pit_ **Pit**
- typedef Cit_ **Cit**

Public Member Functions

- Cit **operator()** (Pit it) const

```
template<typename Pit_, typename Cit_> struct Miniball::CoordAccessor< Pit_, Cit_ >
```

The documentation for this struct was generated from the following file:

- Miniball.hpp

6.7 Miniball::CoordAccessor< Pit_, Cit_ * > Struct Template Reference

Public Types

- typedef Pit_ **Pit**
- typedef Cit_ * **Cit**

Public Member Functions

- Cit **operator()** (Pit it) const

template<typename Pit_, typename Cit_> struct Miniball::CoordAccessor< Pit_, Cit_ * >

The documentation for this struct was generated from the following file:

- Miniball.hpp

6.8 cSDHOptions Class Reference

class for command line option parsing holding option parsing results

```
#include <sdhoptions.h>
```

Public Member Functions

- [cSDHOptions](#) (char const *option_selection=SDHUSAGE_DEFAULT)
- [~cSDHOptions](#) ()
destructor, clean up
- int [Parse](#) (int argc, char **argv, char const *helptext, char const *programe, char const *version, char const *libname, char const *librelease)
- void [OpenCommunication](#) (NS_SDH cSDH &hand)

Public Attributes

- std::string **usage**
- int **debug_level**
- std::ostream * **debuglog**
- int **sdhport**
- char **sdh_rs_device** [MAX_DEV_LENGTH]
- double **timeout**
- unsigned long **rs232_baudrate**
- bool **use_can_esd**

- int **net**
- bool **use_can_peak**
- char **sdh_canpeak_device** [MAX_DEV_LENGTH]
- unsigned long **can_baudrate**
- unsigned int **id_read**
- unsigned int **id_write**
- bool **use_radians**
- bool **use_fahrenheit**
- double **period**
- int **dsaport**
- char **dsa_rs_device** [MAX_DEV_LENGTH]
- bool **do_RLE**
- int **framerate**
- bool **fullframe**
- bool **sensorinfo**
- bool **controllerinfo**
- int **matrixinfo** [6]
- double **sensitivity** [6]
- unsigned int **threshold** [6]
- bool **reset_to_default**
- bool **persistent**
- bool **showsasettings**
- bool **use_tcp**
- std::string **tcp_adr**
- int **tcp_port**

Static Public Attributes

- static int const **MAX_DEV_LENGTH** = 32

6.8.1 Detailed Description

class for command line option parsing holding option parsing results

6.8.2 Constructor & Destructor Documentation

6.8.2.1 **cSDHOptions::cSDHOptions** (char const * *option_selection* = SDHUSAGE_DEFAULT)

constructor: init members to their default values

Parameters

<i>option_selection</i>	<p>- string that names the options to include in helptext for online help. With a text including one of the following keywords the corresponding helptext is added to the usage helptext</p> <ul style="list-style-type: none"> • "general" see sdhusage_general • "sdhcom_serial" see sdhusage_sdhcom_serial • "sdhcom_common" see sdhusage_sdhcom_common • "sdhcom_esdcan" see sdhusage_sdhcom_esdcan • "sdhcom_peakcan" see sdhusage_sdhcom_peakcan • "sdhcom_cancommon" see sdhusage_sdhcom_cancommon • "sdhcom_tcp" see sdhusage_sdhcom_tcp • "sdhother" see sdhusage_sdhother • "dsacom" see sdhusage_dsacom • "dsaother" see sdhusage_dsaother
-------------------------	---

6.8.3 Member Function Documentation

6.8.3.1 void cSDHOptions::OpenCommunication (NS_SDH cSDH & *hand*)

convenience function to open the communication of the given *hand* object according to the parsed parameters.

Parameters

<i>hand</i>	- reference to a cSDH object to open
-------------	--------------------------------------

6.8.3.2 int cSDHOptions::Parse (int *argc*, char ** *argv*, char const * *helptext*, char const * *progrname*, char const * *version*, char const * *libname*, char const * *librelease*)

parse the command line parameters *argc*, *argv* into members. *helptext*, *progrname*, *version*, *libname* and *librelease* are used when printing online help. start parsing at option with index *p_option_index parse all options if parse_all is true, else only one option is parsed

Returns

the optind index of the first non option argument in argv

The documentation for this class was generated from the following files:

- [sdhoptions.h](#)
- [sdhoptions.cpp](#)

6.9 Device Class Reference

Simple structure to manage the device, its type and serial number.

```
#include <extension.h>
```

Public Member Functions

- **Device** (string name, DeviceType type, UInt32 serial)
- **Device** (string name, DeviceType type, UInt32 serial, string format_string)

Public Attributes

- string **device_name**
- DeviceType **device_type**
- UInt32 **serial_no**
- string **device_format_string**

6.9.1 Detailed Description

Simple structure to manage the device, its type and serial number.

The documentation for this class was generated from the following file:

- extension.h

6.10 Ext Class Reference

Extension to the SDH-Library to automatically query all available comports for connected DSA / SDH devices.

```
#include <extension.h>
```

Classes

- struct **sControllerInfo**
A data structure describing the controller info about the remote DSACON32m controller.
- struct **sResponse**
Data structure for storing responses from the remote DSACON32m controller.

Public Types

- enum [eDSAErrorCode](#) {
E_SUCCESS, E_NOT_AVAILABLE, E_NO_SENSOR, E_NOT_INITIALIZED,
E_ALREADY_RUNNING, E_FEATURE_NOT_SUPPORTED, E_INCONSISTENT_DATA, E_TIMEOUT,
E_READ_ERROR, E_WRITE_ERROR, E_INSUFFICIENT_RESOURCES, E_CHECKSUM_ERROR,
E_CMD_NOT_ENOUGH_PARAMS, E_CMD_UNKNOWN, E_CMD_FORMAT_ERROR, E_ACCESS_DENIED,
E_ALREADY_OPEN, E_CMD_FAILED, E_CMD_ABORTED, E_INVALID_HANDLE,
E_DEVICE_NOT_FOUND, E_DEVICE_NOT_OPENED, E_IO_ERROR, E_INVALID_PARAMETER,
E_INDEX_OUT_OF_BOUNDS, E_CMD_PENDING, E_OVERRUN, E_RANGE_ERROR }

Error codes returned by the remote DSACON32m tactile sensor controller.

Public Member Functions

- struct [Ext::sControllerInfo](#) **SDH__attribute__** ((__packed__))
- [Ext](#) (int debug_level, std::list< [Device](#) * > &_deviceList)
Constructor.
- [~Ext](#) ()
Destructor: clean up and delete dynamically allocated memory.
- void [Open](#) () throw (ExtException*)
- void [Close](#) () throw (ExtException*)
- string [getComportDriver](#) (const string &tty)
- void [addComport](#) (list< string > &comList, list< string > &comList8250, const string &dir)
- void [probeSerial8250Comports](#) (list< string > &comList, list< string > comList8250)
- list< string > [listComports](#) ()
- void [IdentifyDevices](#) ()

Friends

- std::ostream & **operator**<< (std::ostream &stream, Ext::sResponse const &response)

6.10.1 Detailed Description

Extension to the SDH-Library to automatically query all available comports for connected DSA / SDH devices.

Note

Code is partially taken from the SDH-Library

6.10.2 Member Function Documentation

6.10.2.1 `void Ext::addComport (list< string > & comList, list< string > & comList8250, const string & dir)`

Register the available device Credits go to Søren Holm: <http://stackoverflow.com/questions/2530096/how>

Parameters

<i>comList</i>	The final list of devices.
<i>comList8250</i>	A separate list of serial8250-devices
<i>dir</i>	The device directory

Returns

void

6.10.2.2 `void Ext::Close () throw (ExtException*)`

Set the framerate of the remote DSACON32m controller to 0 and close connection to it.

Returns

void

6.10.2.3 `string Ext::getComportDriver (const string & tty)`

Enumeration of available com ports Credits go to Søren Holm: <http://stackoverflow.com/questions/2530096/how>

Parameters

<i>tty</i>	The tty-path
------------	--------------

Returns

The driver name

6.10.2.4 `void Ext::IdentifyDevices ()`

Auto-detect available tty devices based on response and serial number

Returns

void

6.10.2.5 `list< string > Ext::listComports ()`

List available comports Credits go to Søren Holm: <http://stackoverflow.com/questions/2530096>

Returns

The device list

6.10.2.6 `void Ext::Open () throw (ExtException*)`

(Re-)open connection to DSACon32m controller, this is called by the constructor automatically, but is still useful to call after a call to [Close\(\)](#)

Returns

void

6.10.2.7 `void Ext::probeSerial8250Comports (list< string > & comList, list< string > comList8250)`

Serial8250-devices must be probe to check for validity Credits go to Søren Holm: <http://stackoverflow.com>

Parameters

<i>comList</i>	The final list of devices.
<i>comList8250</i>	A separate list of serial8250-devices

Returns

void

The documentation for this class was generated from the following files:

- extension.h
- extension.cpp

6.11 ExtException Class Reference

Derived exception class for low-level DSA related exceptions.

```
#include <extension.h>
```

Public Member Functions

- **ExtException** (cMsg const &_msg)

6.11.1 Detailed Description

Derived exception class for low-level DSA related exceptions.

The documentation for this class was generated from the following file:

- extension.h

6.12 FeatureExtraction Class Reference

Unifies the computation of features, in detail: The standard deviation, [Chebyshev](#) moments and minimal bounding sphere.

```
#include <featureExtraction.h>
```

Public Member Functions

- [FeatureExtraction](#) ([FrameManager](#) &fm)
- `std::vector< double > computeCentroid` (int frameID, int matrixID)
- `array_type computeMoments` (int frameID, int matrixID, int pmax)
- `double computeStandardDeviation` (int frameID, int matrixID)
- `std::vector< double > computeMiniball` (int frameID, double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double phi6)
- `std::vector< double > computeMiniball` (int frameID, std::vector< double > &angles)
- `std::vector< double > computeMiniballCentroid` (int frameID, double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double phi6)
- `std::vector< double > computeMiniballCentroid` (int frameID, std::vector< double > &angles)
- `std::vector< double > computeMiniballPoints` (std::vector< std::vector< double > > &taxels, double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double phi6)
- `std::vector< double > computeMiniballPoints` (std::vector< std::vector< double > > &taxels, std::vector< double > &angles)

6.12.1 Detailed Description

Unifies the computation of features, in detail: The standard deviation, [Chebyshev](#) moments and minimal bounding sphere.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 FeatureExtraction::FeatureExtraction ([FrameManager](#) & *fm*)

Constructor

6.12.3 Member Function Documentation

6.12.3.1 `std::vector< double > FeatureExtraction::computeCentroid (int frameID, int matrixID)`

Computes center of mass (in texel coordinates)

Parameters

<i>frameID</i>	The frameID.
<i>matrixID</i>	The matrixID.

Returns

The centroid.

6.12.3.2 `std::vector< double > FeatureExtraction::computeMiniball (int frameID, double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double phi6)`

Compute the minimal bounding sphere of active cells. See overloaded variant.

Parameters

<i>frameID</i>	The frameID.
<i>phi0,phi1,phi2</i>	Joint angles [phi0 .. phi6].

Returns

Center and radius of miniball.

6.12.3.3 `std::vector< double > FeatureExtraction::computeMiniball (int frameID, std::vector< double > & angles)`

Compute the minimal bounding sphere of active cells, overloaded variant.

Parameters

<i>frameID</i>	The frameID.
<i>angles</i>	Joint angles [phi0 .. phi6].

Returns

Center and radius of miniball.

6.12.3.4 `std::vector< double > FeatureExtraction::computeMiniballCentroid (int frameID,
std::vector< double > & angles)`

Compute the minimal bounding sphere based on the per matrix centroid of active cells, overloaded variant.

Parameters

<i>frameID</i>	The frameID.
<i>angles</i>	Joint angles [phi0 .. phi6].

Returns

Center and radius of miniball.

6.12.3.5 `std::vector< double > FeatureExtraction::computeMiniballCentroid (int frameID,
double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double
phi6)`

Compute the minimal bounding sphere based on the per matrix centroid of active cells. See overloaded variant.

Parameters

<i>frameID</i>	The frameID.
<i>phi0,phi1,phi2</i>	Joint angles [phi0 .. phi6].

Returns

Center and radius of miniball.

6.12.3.6 `std::vector< double > FeatureExtraction::computeMiniballPoints (std::vector<
std::vector< double > > & taxels, std::vector< double > & angles)`

Compute the minimal bounding sphere based on the specified taxels, overloaded variant.

Parameters

<i>taxels</i>	Vector of specified taxels per matrix .
<i>angles</i>	Joint angles [phi0 .. phi6].

Returns

Center and radius of miniball.

6.12.3.7 `std::vector< double > FeatureExtraction::computeMiniballPoints (std::vector< std::vector< double > > & taxels, double phi0, double phi1, double phi2, double phi3, double phi4, double phi5, double phi6)`

Compute the minimal bounding sphere based on the specified taxels. See overloaded variant.

Parameters

<i>taxels</i>	Vector of specified taxels per matrix .
<i>angles</i>	Joint angles [<i>phi0</i> .. <i>phi6</i>].

Returns

Center and radius of miniball.

6.12.3.8 `array_type FeatureExtraction::computeMoments (int frameID, int matrixID, int pmax)`

Computes [Chebyshev](#) moments

Parameters

<i>frameID</i>	The frameID.
<i>matrixID</i>	The matrixID.
<i>pmax</i>	Maximum moment order.

Returns

The [Chebyshev](#) moments.

6.12.3.9 `double FeatureExtraction::computeStandardDeviation (int frameID, int matrixID)`

Computes standard deviation of tactile sensor frames (intensity values, not 2D image moments). Only active cells are considered

Parameters

<i>frameID</i>	The frameID.
<i>matrixID</i>	The matrixID.

Returns

The standard deviation.

The documentation for this class was generated from the following files:

- featureExtraction.h
- featureExtraction.cpp

6.13 FeatureExtractionWrapper Class Reference

Defines Boost.Python wrappers around the [FeatureExtraction](#) class. See Python examples for usage.

Public Member Functions

- **FeatureExtractionWrapper** ([FrameManagerWrapper](#) &fmw)
- **np::ndarray computeCentroid** (int frameID, int matrixID)
- **double computeStandardDeviation** (int frameID, int matrixID)
- **bp::list computeStandardDeviationList** (int frameID)
- **np::ndarray computeChebyshevMoments** (int frameID, int matrixID, int pmax)
- **bp::list computeChebyshevMomentsList** (int frameID, int pmax)
- **np::ndarray computeMinimalBoundingSphere** (int frameID, np::ndarray &phi_ndarray)
- **np::ndarray computeMinimalBoundingSphereCentroid** (int frameID, np::ndarray &phi_ndarray)
- **np::ndarray computeMinimalBoundingSpherePoints** (np::ndarray &taxel_ndarray, np::ndarray &phi_ndarray)

6.13.1 Detailed Description

Defines Boost.Python wrappers around the [FeatureExtraction](#) class. See Python examples for usage.

The documentation for this class was generated from the following file:

- framemanager_python.cpp

6.14 ForwardKinematics Class Reference

Implements forward kinematics for the tactile sensors.

```
#include <forwardKinematics.h>
```

Public Member Functions

- [ForwardKinematics](#) ()
- void [setAngles](#) (std::vector< double > &all_angles)
- cv::Matx44d [computeTransformationMatrixTaxelXYZ](#) (int m, int y)
- cv::Matx44d [computeTransformationMatrixPointOnSensorPlaneXYZ](#) (int m, double y)
- std::vector< double > [GetTaxelXYZ](#) (int m, int x, int y)
- std::vector< double > [GetPointOnSensorPlaneXYZ](#) (int m, double x, double y)

6.14.1 Detailed Description

Implements forward kinematics for the tactile sensors.

The transformations are based on the classic Denavit-Hartenberg convention. The coordinate center is at gripper's base, i.e. in the middle of the equilateral triangle. See Figure 4.4: Isometric projection of my thesis for details. See `featureExtraction.cpp` for example usage.

Note

Written for readability. Let's rely on the look-up table approach and the compiler's optimization.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 `ForwardKinematics::ForwardKinematics ()`

Constructor automatically creates static transformation matrices

6.14.3 Member Function Documentation

6.14.3.1 `cv::Matx44d ForwardKinematics::computeTransformationMatrixPointOnSensorPlaneXYZ (int m, double y)`

Computes transformation matrix of position(`_y`) on matrix `m`

Parameters

<i>m</i>	Matrix index
<i>y</i>	y-coordinate on sensor matrix (only needed for distal matrix)

Returns

Transformation matrix relative to hand's origin

6.14.3.2 `cv::Matx44d ForwardKinematics::computeTransformationMatrixTaxelXYZ (int m, int y)`

Computes transformation matrix of `taxel(_y)` on matrix `m`

Parameters

<i>m</i>	Matrix index
<i>y</i>	y-coordinate on sensor matrix (only needed for distal matrix)

Returns

Transformation matrix relative to hand's origin

6.14.3.3 `std::vector< double > ForwardKinematics::GetPointOnSensorPlaneXYZ (int m, double x, double y)`

Computes Cartesian coordinates (in mm) of position(*x*,*y*) on matrix *m*

(0.0, 0.0) refers to the center of the top-left taxel(0,0) Meaningful values that actual lie on the sensor plane: Proximal: ([-1.7, 18.7], [-1.7, 45.9]) Distal: ([-1.7, 18.7], [-1.7, 42.5])

Parameters

<i>m</i>	- Matrix index.
<i>x</i>	- x-coordinate on sensor matrix.
<i>y</i>	- y-coordinate on sensor matrix.

Returns

Position [*x*,*y*,*z*] relative to hand's origin.

6.14.3.4 `std::vector< double > ForwardKinematics::GetTaxelXYZ (int m, int x, int y)`

Computes Cartesian coordinates (in mm) of taxel(*x*,*y*) on matrix *m*

Parameters

<i>m</i>	- Matrix index.
<i>x</i>	- x-coordinate on sensor matrix.
<i>y</i>	- y-coordinate on sensor matrix.

Returns

Position [*x*,*y*,*z*] relative to hand's origin.

6.14.3.5 `void ForwardKinematics::setAngles (std::vector< double > & all_angles)`

Initializes dynamic transformation matrices

Parameters

<i>all_angles</i>	Angles [<i>phi</i> 0 .. <i>phi</i> 6] in Radians.
-------------------	--

Returns

void

The documentation for this class was generated from the following files:

- forwardKinematics.h
- forwardKinematics.cpp

6.15 FrameGrabberDSA Class Reference

A frame grabber/recorder class for tactile sensor based on boost thread.

```
#include <framegrabberDSA.h>
```

Public Member Functions

- [FrameGrabberDSA](#) (cDSA *dsa)
- [FrameGrabberDSA](#) (cDSA *dsa, [FrameManager](#) *fm)
- [~FrameGrabberDSA](#) ()
- void [setFrameManager](#) ([FrameManager](#) *fm)
- void [setFramerate](#) (double frameRate)
- void [start](#) (double frameRate, bool startPaused=false, bool startRecording=false)
- void [execute](#) ()
- void [pause](#) ()
- void [resume](#) ()
- void [finish](#) ()
- void [enableRecording](#) ()
- void [suspendRecording](#) ()
- bool [isRunning](#) ()
- bool [isCapturing](#) ()
- bool [isRecording](#) ()
- int [getFrameNumber](#) ()

6.15.1 Detailed Description

A frame grabber/recorder class for tactile sensor based on boost thread.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 [FrameGrabberDSA::FrameGrabberDSA](#) (cDSA * dsa)

Constructor (not recording yet).

Parameters

<i>dsa</i>	The tactile sensor controller.
------------	--------------------------------

6.15.2.2 [FrameGrabberDSA::FrameGrabberDSA](#) (cDSA * dsa, [FrameManager](#) * fm)

Constructor (without frame manager).

Parameters

<i>dsa</i>	The tactile sensor controller.
------------	--------------------------------

<i>fm</i>	The frame manager.
-----------	--------------------

6.15.2.3 FrameGrabberDSA::~~FrameGrabberDSA ()

Deconstructor: Interrupts thread in as soon as possible.

6.15.3 Member Function Documentation

6.15.3.1 void FrameGrabberDSA::enableRecording ()

Automatically request new frames and store them permanently.

Returns

void

6.15.3.2 void FrameGrabberDSA::execute ()

Executes the grabbing / recording thread.

Returns

void

6.15.3.3 void FrameGrabberDSA::finish ()

Stops the grabber thread. Interrupts thread during next wait/sleep state.

Returns

void

6.15.3.4 int FrameGrabberDSA::getFrameNumber () [inline]

Returns the number of already captured frames.

Returns

The number of already captured frames.

6.15.3.5 `bool FrameGrabberDSA::isCapturing ()`

Reports if the thread is capturing or not.

Returns

The state.

6.15.3.6 `bool FrameGrabberDSA::isRecording ()`

Reports if the thread is recording or not.

Returns

The state.

6.15.3.7 `bool FrameGrabberDSA::isRunning ()`

Reports if the thread is running or paused.

Returns

The state.

6.15.3.8 `void FrameGrabberDSA::pause ()`

Stop DSA push-mode (if active) and halt thread execution.

Returns

void

6.15.3.9 `void FrameGrabberDSA::resume ()`

Resume thread execution with current configuration.

Returns

void

6.15.3.10 `void FrameGrabberDSA::setFrameManager (FrameManager * fm)`

Sets the frame manager.

Parameters

<i>fm</i>	The frame manager.
-----------	--------------------

Returns

void

6.15.3.11 void FrameGrabberDSA::setFramerate (double *frameRate*)

Sets the frame rate.

For frame rates < 30 , tactile sensor frames are manually requested (pull mode). Otherwise, the DSA controller switches to an automatic push mode.

Parameters

<i>frameRate</i>	The desired frame rate.
------------------	-------------------------

Returns

void

6.15.3.12 void FrameGrabberDSA::start (double *frameRate*, bool *startPaused* = false, bool *startRecording* = false)

Initializes DSA [Controller](#) and starts the execution of the grabber thread.

For frame rates < 30 , tactile sensor frames are manually requested (pull mode). Otherwise, the DSA controller switches to an automatic push mode.

Parameters

<i>frameRate</i>	The desired frame rate.
<i>startPaused</i>	Should the thread start paused or immediately start grabbing?
<i>startRecording</i>	Should the grabbing thread immediately start recording?

Returns

void

6.15.3.13 void FrameGrabberDSA::suspendRecording ()

Pause storing of new frames.

Returns

void

The documentation for this class was generated from the following files:

- framegrabberDSA.h
- framegrabberDSA.cpp

6.16 FrameGrabberSDH Class Reference

A frame grabber/recorder class for joint angles and temperatures based on boost thread.

```
#include <framegrabberSDH.h>
```

Public Member Functions

- [FrameGrabberSDH](#) (cSDH *sdh)
- [FrameGrabberSDH](#) (cSDH *sdh, [FrameManager](#) *fm)
- [~FrameGrabberSDH](#) ()
- void [setFrameManager](#) ([FrameManager](#) *fm)
- void [setFramerateJointAngles](#) (double frameRate)
- void [setFramerateTemperature](#) (double frameRate)
- void [setTemperature](#) (bool enable)
- void [setJointAngle](#) (bool enable)
- void [start](#) (double FPSJointAngles, double FPSTemperature, bool startPaused=false, bool startRecording=false)
- void [execute](#) ()
- void [pause](#) ()
- void [pauseBlocking](#) ()
- void [resume](#) ()
- void [finish](#) ()
- void [enableRecording](#) ()
- void [suspendRecording](#) ()
- bool [isRunning](#) ()
- bool [isCapturing](#) ()
- bool [isRecording](#) ()
- int [getFrameNumber](#) ()

6.16.1 Detailed Description

A frame grabber/recorder class for joint angles and temperatures based on boost thread.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 FrameGrabberSDH::FrameGrabberSDH (cSDH * *sdh*)

Constructor (without frame manager).

Parameters

<i>sdh</i>	The SCHUNK SDH-2.
------------	-------------------

6.16.2.2 FrameGrabberSDH::FrameGrabberSDH (cSDH * *sdh*, FrameManager * *fm*)

Constructor (ready to record).

Parameters

<i>dsa</i>	The tactile sensor controller.
<i>fm</i>	The frame manager.

6.16.2.3 FrameGrabberSDH::~~FrameGrabberSDH ()

Deconstructor: Interrupts thread in as soon as possible.

6.16.3 Member Function Documentation**6.16.3.1 void FrameGrabberSDH::enableRecording ()**

Automatically request new frames and store them permanently.

Returns

void

6.16.3.2 void FrameGrabberSDH::execute ()

Executes the grabbing / recording thread.

Returns

void

6.16.3.3 void FrameGrabberSDH::finish ()

Stops the grabber thread. Interrupts thread during next wait/sleep state.

Returns

void

6.16.3.4 int FrameGrabberSDH::getFrameNumber () [inline]

Returns the number of already captured frames.

Returns

The number of already captured frames.

6.16.3.5 bool FrameGrabberSDH::isCapturing ()

Reports if the thread is capturing or not.

Returns

The state.

6.16.3.6 bool FrameGrabberSDH::isRecording ()

Reports if the thread is recording or not.

Returns

The state.

6.16.3.7 bool FrameGrabberSDH::isRunning ()

Reports if the thread is running or paused.

Returns

The state.

6.16.3.8 void FrameGrabberSDH::pause ()

Halt thread execution.

Returns

void

6.16.3.9 void FrameGrabberSDH::pauseBlocking ()

Halt thread execution and wait for end of transmission.

Returns

void

6.16.3.10 void FrameGrabberSDH::resume ()

Resume thread execution with current configuration.

Returns

void

6.16.3.11 void FrameGrabberSDH::setFrameManager (FrameManager * *fm*)

Sets the frame manager.

Parameters

<i>fm</i>	The frame manager.
-----------	--------------------

Returns

void

6.16.3.12 void FrameGrabberSDH::setFramerateJointAngles (double *frameRate*)

Sets the frame rate of the joint angle requests.

Parameters

<i>frameRate</i>	The desired frame rate.
------------------	-------------------------

6.16.3.13 void FrameGrabberSDH::setFramerateTemperature (double *frameRate*)

Sets the frame rate of the joint angle requests.

Parameters

<i>frameRate</i>	The desired frame rate.
------------------	-------------------------

6.16.3.14 void FrameGrabberSDH::setJointAngle (bool *enable*)

Separate flag to capture/record joint angle readings

Parameters

<i>enable</i>	The capturing state.
---------------	----------------------

6.16.3.15 void FrameGrabberSDH::setTemperature (bool *enable*)

Separate flag to capture/record temperature readings

Parameters

<i>enable</i>	The capturing state.
---------------	----------------------

6.16.3.16 `void FrameGrabberSDH::start (double FPSJointAngles, double FPSTemperature,
bool startPaused = false, bool startRecording = false)`

Initializes the SDH-2 and starts the execution of the grabber thread.

Parameters

<i>FPSJointAngles</i>	The desired joint angle frame rate.
<i>FPSTemperatureThe</i>	desired temperature frame rate.
<i>startPaused</i>	Should the thread start paused or immediately start grabbing?
<i>startRecording</i>	Should the grabbing thread immediately start recording?

6.16.3.17 `void FrameGrabberSDH::suspendRecording ()`

Pause storing of new frames.

Returns

void

The documentation for this class was generated from the following files:

- framegrabberSDH.h
- framegrabberSDH.cpp

6.17 FrameManager Class Reference

The heart of this project.

```
#include <framemanager.h>
```

Public Member Functions

- [FrameManager](#) ()
- void [resetOffline](#) ()
- void [resetOnline](#) ()
- void [setSDH](#) (cSDH *sdh)
- void [setDSA](#) (cDSA *dsa)
- bool [isConnectedSDH](#) ()
- bool [isConnectedDSA](#) ()
- void [queryDSAInfo](#) ()
- void [setSensitivity](#) (uint matrixID, float sensitivity)
- void [setThreshold](#) (uint matrixID, float threshold)

- void [setFrameGrabberDSA](#) ([FrameGrabberDSA](#) *fgDSA)
- void [setFrameGrabberSDH](#) ([FrameGrabberSDH](#) *fgSDH)
- [sensorInfo](#) & [getSensorInfo](#) ()
- [matrixInfo](#) & [getMatrixInfo](#) (uint i)
- uint [getNumMatrices](#) ()
- uint [getNumCells](#) ()
- uint [getCurrentFrameID](#) ()
- void [setCurrentFrameID](#) (uint frameID)
- bool [getTSFrameAvailable](#) ()
- void [setTSFrameAvailable](#) (bool value)
- bool [getJointAngleFrameAvailable](#) ()
- void [setJointAngleFrameAvailable](#) (bool value)
- void [setLiveFrame](#) ()
- [TSFrame](#) & [allocateTSFrame](#) ()
- void [addTSFrame](#) ()
- void [deleteTSFrame](#) (uint frameID)
- void [cropToFrames](#) (uint64_t timestamp_from, uint64_t timestamp_to)
- [TSFrame](#) * [getFrame](#) (uint frameID)
- [TSFrame](#) * [getCurrentFrame](#) ()
- [TSFrame](#) * [getFilteredFrame](#) (uint frameID)
- [TSFrame](#) * [getCurrentFilteredFrame](#) ()
- float [getTexel](#) (uint frameID, uint cellID)
- float [getTexel](#) (uint frameID, uint m, uint x, uint y)
- float [getFilteredTexel](#) (uint frameID, uint cellID)
- float [getFilteredTexel](#) (uint frameID, uint m, uint x, uint y)
- uint [getFrameCountTS](#) ()
- uint [getFrameCountTemperature](#) ()
- uint [getFrameCountJointAngles](#) ()
- void [requestTemperatureFrame](#) (bool record)
- [TemperatureFrame](#) * [getTemperatureFrame](#) (uint tempID)
- void [requestJointAngleFrame](#) (bool record)
- [JointAngleFrame](#) * [getJointAngleFrame](#) (uint angleID)
- [JointAngleFrame](#) * [getCurrentJointAngleFrame](#) ()
- void [createTemperatureMapping](#) ()
- void [createJointAngleMapping](#) ()
- [TemperatureFrame](#) * [getCorrespondingTemperature](#) (uint frameID)
- [JointAngleFrame](#) * [getCorrespondingJointAngle](#) (uint frameID)
- void [convertCellIndex](#) (uint cellID, uint &m, uint &x, uint &y)
- uint [convertCellIndex](#) (uint m, uint x, uint y)
- void [selectCell](#) (uint cellID, bool value)
- bool [isSelected](#) (int cellID)
- int [getNumSelectedCells](#) ()
- std::vector< bool > & [getSelection](#) ()
- std::vector< int > [createSelectedCellsIdx](#) ()
- void [setDynamicMask](#) (std::vector< bool > &bitmask)
- bool [getStaticMask](#) (uint m, uint x, uint y)

- bool [getDynamicMask](#) (uint m, uint x, uint y)
- [FrameProcessor](#) * [getFrameProcessor](#) ()
- void [setFilterNone](#) ()
- void [setFilterMedian](#) (int kernelRadius, bool masked)
- void [setFilterMedian3D](#) (bool masked)
- void [setFilterGaussian](#) (int kernelRadius, double sigma, int borderType)
- void [setFilterBilateral](#) (int kernelSize, double sigmaColor, double sigmaSpace, int borderType)
- void [setFilterMorphological](#) (int kernelType, int kernelRadius, bool masked, int borderType)
- boost::shared_ptr< [SlipDetector](#) > [getSlipDetector](#) (uint matrixID)
- void [enableSlipDetection](#) (uint matrixID)
- void [disableSlipDetection](#) (uint matrixID)
- bool [getSlipDetectionState](#) ()
- bool [getSlipDetectionState](#) (uint matrixID)
- void [setSlipThresholdReference](#) (double thresh)
- void [setSlipThresholdConsecutive](#) (double thresh)
- bool [setSlipReferenceFrameLive](#) (uint matrixID)
- [slipResult](#) [computeSlipLive](#) (uint matrixID)
- std::vector< boost::optional< [slipResult](#) > > [getSlipLive](#) ()
- bool [getSlipLiveBinary](#) ()
- void [slipResultProducer](#) ()
- std::vector< boost::optional< [slipResult](#) > > [slipResultConsumer](#) ()
- bool [setSlipReferenceFrame](#) (uint matrixID, uint frameID)
- [slipResult](#) [computeSlip](#) (uint matrixID, uint frameID)
- const std::string & [getProfileName](#) ()
- void [loadFrames](#) (const std::string &filename)
- void [storeFrames](#) (const std::string &filename)
- void [print_tree](#) (const boost::property_tree::ptree &pt, int level)
- void [print_tree](#) (const boost::property_tree::ptree &pt)
- float [hex2float](#) (std::string &s)
- std::string [float2hex](#) (float f)
- void [decodeTSFrame](#) (const std::string &hexdata, std::vector< float > &decoded_frame)
- void [encodeTSFrame](#) (const std::vector< float > &cells, std::string &hexdata)
- void [loadProfile](#) (const std::string &filename)
- void [storeProfile](#) (const std::string &filename)
- void [storeProfileSelection](#) (const string &filename, uint fromIdxTS, uint toIdxTS)
- void [printJointAngleFrame](#) ([JointAngleFrame](#) &jointAngleFrame)
- void [printTemperatureFrame](#) ([TemperatureFrame](#) &temperatureFrame)
- void [printTSMatrix](#) (uint frameID, uint m)
- void [printTSMatrices](#) (uint frameID)

6.17.1 Detailed Description

The heart of this project.

Note

A word on copy constructors: Since `boost::mutex` and `boost::condition_variable` are not copyable simply delete the copy constructor. Alternative: write a copy constructor that copies the data but constructs a new mutex, i.e. `FrameManager(const FrameManager&) = delete;`

6.17.2 Constructor & Destructor Documentation

6.17.2.1 FrameManager::FrameManager ()

Constructor. Calls [resetOffline\(\)](#).

6.17.3 Member Function Documentation

6.17.3.1 void FrameManager::addTSFrame ()

Adds the current frame to the record

Returns

void

6.17.3.2 TSFrame & FrameManager::allocateTSFrame ()

Allocates space for new tactile sensor frame on the queue and returns reference.

Returns

Reference to the allocated tactile sensor frame.

6.17.3.3 slipResult FrameManager::computeSlip (uint *matrixID*, uint *frameID*)

Performs both translational and rotational slip-detection (offline version).

Parameters

<i>matrixID</i>	The matrix ID.
<i>frameID</i>	The frame ID.

Returns

The combined results of the rotational and translational slip-detection.

6.17.3.4 `slipResult` `FrameManager::computeSlipLive (uint matrixID)`

Performs both translational and rotational slip-detection (live version).

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

Returns

The combined results of the rotational and translational slip-detection.

6.17.3.5 `void` `FrameManager::convertCellIndex (uint cellID, uint & m, uint & x, uint & y)` [inline]

Convert *cellID* of entire frame to (*matrixID*, *x*, *y*)

Parameters

<i>cellID</i>	The cell ID in range [0, 486]
<i>m</i>	The matrix ID.
<i>x,y</i>	The Taxel coordinates.

Returns

`void`

6.17.3.6 `uint` `FrameManager::convertCellIndex (uint m, uint x, uint y)` [inline]

Convert (*matrixID*, *x*, *y*) to *cellID* of entire frame

Parameters

<i>m</i>	The matrix ID.
<i>x,y</i>	The Taxel coordinates.

Returns

The *cellID*

6.17.3.7 `void` `FrameManager::createJointAngleMapping ()`

Creates a mapping between tactile sensor frame and associated joint angles.

Returns

`void`

6.17.3.8 `std::vector< int > FrameManager::createSelectedCellsIdx ()`

Creates a list of indices of selected taxels.

Returns

The vector containing cell IDs of selected taxels.

6.17.3.9 `void FrameManager::createTemperatureMapping ()`

Creates a mapping between tactile sensor frame and associated temperature.

Returns

void

6.17.3.10 `void FrameManager::cropToFrames (uint64_t timestamp_from, uint64_t timestamp_to)`

Trim frames (tactile sensor, temperature, joint angles) to selection [from to] including borders.

Parameters

<i>timestamp_ - from</i>	From.
<i>timestamp_ - to</i>	To.

Returns

void

6.17.3.11 `void FrameManager::decodeTSFrame (const std::string & hexdata, std::vector< float > & decoded_frame)`

Decode "Enhanced RLE" compressed sensor frame

Only zero-valued matrix elements are run length encoded. Each token t consists of a single precision floating point value (IEEE-754). $t < 0$ indicates $|t|$ consecutive zeros. $t > 0$: The value of t represents a single observation.

Note

Tested with Little-Endian byte order only.

Parameters

<i>hexdata</i>	Encoded hexadecimal string.
<i>decoded_ - frame</i>	The resulting decoded vector of taxel values.

Returns

void

6.17.3.12 void FrameManager::deleteTSFrame (uint *frameID*)

Deletes a single [TSFrame](#) without deleting corresponding temperatures/angles Use deleteTSFrames(from, to) to remove multiple frames.

Note

Operation might be expensive due to the used queue data structure.

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

void

6.17.3.13 void FrameManager::disableSlipDetection (uint *matrixID*)

Disables slip-detection on specified sensor matrix.

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

6.17.3.14 void FrameManager::enableSlipDetection (uint *matrixID*)

Enables slip-detection on specified sensor matrix.

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

6.17.3.15 void FrameManager::encodeTSFrame (const std::vector< float > & *cells*, std::string & *hexdata*)

Frames are encoded using an "Enhanced RLE Compression"

Meaning only zero-valued matrix elements are run length encoded. Each token t consists of a single precision floating point value (IEEE-754). $t < 0$ indicates $|t|$ consecutive zeros. $t > 0$: The value of t represents a single observation.

Note

Tested with Little-Endian byte order only.

Parameters

<i>cells</i>	A vector of taxel values.
<i>hexdata</i>	The hexadecimal representation.

Returns

void

6.17.3.16 string FrameManager::float2hex (float *f*)

Converts a float to a hex string.

Parameters

<i>f</i>	The converted float.
----------	----------------------

Returns

The hex string.

6.17.3.17 JointAngleFrame * FrameManager::getCorrespondingJointAngle (uint *frameID*)

Given a tactile sensor frame, returns the associated joint angle frame.

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

Pointer to the recorded joint angle frame.

6.17.3.18 TemperatureFrame * FrameManager::getCorrespondingTemperature (uint *frameID*)

Given a tactile sensor frame, returns the associated temperature frame.

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

Pointer to the recorded temperature frame.

6.17.3.19 TSFrame * FrameManager::getCurrentFilteredFrame ()

Get current frame of recorded frame history (specified filter is applied).

Returns

Pointer to the tactile sensor frame.

6.17.3.20 TSFrame * FrameManager::getCurrentFrame ()

Get current frame of recorded frame history.

Returns

Pointer to the tactile sensor frame.

6.17.3.21 uint FrameManager::getCurrentFrameID ()

Returns the current frame ID.

Returns

The current frame ID.

6.17.3.22 JointAngleFrame * FrameManager::getCurrentJointAngleFrame ()

Returns current joint angle frame.

Returns

Pointer to the joint angle frame.

6.17.3.23 bool FrameManager::getDynamicMask (uint *m*, uint *x*, uint *y*)

Gets dynamic mask of specified taxel.

Parameters

<i>m</i>	The matrix ID.
<i>x,y</i>	The taxel coordinates.

Returns

The masking state.

6.17.3.24 TSFrame * FrameManager::getFilteredFrame (uint *frameID*)

Get specified frame of recorded frame history (specified filter is applied).

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

Pointer to the tactile sensor frame.

6.17.3.25 float FrameManager::getFilteredTexel (uint *frameID*, uint *cellID*)

Returns filtered taxel value of specified frame and cell ID.

Parameters

<i>frameID</i>	The frame ID.
<i>cellID</i>	The cell ID in Range [0, 486]

Returns

The taxel value.

6.17.3.26 float FrameManager::getFilteredTexel (uint *frameID*, uint *m*, uint *x*, uint *y*)

Returns filtered taxel value of specified frame, matrix and coordinate.

Parameters

<i>frameID</i>	
<i>m</i>	The matrix ID.
<i>x,y</i>	Taxel coordinates.

Returns

The taxel value.

6.17.3.27 TSFrame * FrameManager::getFrame (uint *frameID*)

Get specified frame of recorded frame history.

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

Pointer to the tactile sensor frame.

6.17.3.28 `uint FrameManager::getFrameCountJointAngles ()`

Returns the number of recorded joint angle readings.

Returns

The joint angle frame count.

6.17.3.29 `uint FrameManager::getFrameCountTemperature ()`

Returns the number of recorded temperature readings.

Returns

The temperature frame count.

6.17.3.30 `uint FrameManager::getFrameCountTS ()`

Returns the number of recorded tactile sensor frames.

Returns

The tactile sensor frame count.

6.17.3.31 `FrameProcessor * FrameManager::getFrameProcessor ()`

Gets the frame processor.

Returns

The frame processor.

6.17.3.32 `JointAngleFrame * FrameManager::getJointAngleFrame (uint angleID)`

Returns specified joint angle frame.

Parameters

<i>angleID</i>	The joint angle frame ID.
----------------	---------------------------

Returns

Pointer to the joint angle frame.

6.17.3.33 `bool FrameManager::getJointAngleFrameAvailable ()`

Checks if a joint angle frame is available.

Returns

The state.

6.17.3.34 `matrixInfo& FrameManager::getMatrixInfo (uint i) [inline]`

Gets the queried matrix info.

Parameters

<i>i</i>	The matrix.
----------	-------------

Returns

Reference to [matrixInfo](#).

6.17.3.35 `uint FrameManager::getNumCells () [inline]`

Returns the number of taxels.

Returns

The number of taxels.

6.17.3.36 `uint FrameManager::getNumMatrices () [inline]`

Returns the number of matrices.

Returns

The number of matrices.

6.17.3.37 `int FrameManager::getNumSelectedCells ()`

Returns the number of selected taxels.

Returns

The number of selected taxels.

6.17.3.38 `const string & FrameManager::getProfileName ()`

Gets the pressure profile's file name.

Returns

The file name.

6.17.3.39 `std::vector< bool > & FrameManager::getSelection ()`

Returns a mask of selected taxels.

Returns

Reference to mask.

6.17.3.40 `sensorInfo& FrameManager::getSensorInfo ()` `[inline]`

Gets the queried sensor info.

Returns

Reference to [sensorInfo](#).

6.17.3.41 `bool FrameManager::getSlipDetectionState ()`

Checks the combined slip-state.

Returns

The slip-state.

6.17.3.42 `bool FrameManager::getSlipDetectionState (uint matrixID)`

Checks the slip-state on specified sensor matrix.

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

Returns

The slip-state.

6.17.3.43 `boost::shared_ptr< SlipDetector > FrameManager::getSlipDetector (uint matrixID)`

Returns the slip-detector of specified matrix.

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

Returns

The slip-detector.

6.17.3.44 `std::vector< boost::optional< slipResult > > FrameManager::getSlipLive ()`

Returns the results of the last slip computation (live version).

Note

Note: Might lead to a deadlock in combination with [getSlipLiveBinary\(\)](#)

Returns

The combined results.

6.17.3.45 `bool FrameManager::getSlipLiveBinary ()`

Computes a binary slip indicator from the slip results and the given thresholds

Note

Note: Might lead to a deadlock in combination with [getSlipLive\(\)](#)

Returns

The combined slip state.

6.17.3.46 `bool FrameManager::getStaticMask (uint m, uint x, uint y)`

Gets static mask of specified taxel.

Parameters

<i>m</i>	The matrix ID.
<i>x,y</i>	The taxel coordinates.

Returns

The masking state.

6.17.3.47 `TemperatureFrame * FrameManager::getTemperatureFrame (uint tempID)`

Returns specified temperature frame.

Parameters

<i>tempID</i>	The temperature frame ID.
---------------	---------------------------

Returns

Pointer to the temperature frame.

6.17.3.48 `float FrameManager::getTexel (uint frameID, uint m, uint x, uint y)`

Returns taxel value of specified frame, matrix and coordinate.

Parameters

<i>frameID</i>	
<i>m</i>	The matrix ID.
<i>x,y</i>	Taxel coordinates.

Returns

The taxel value.

6.17.3.49 `float FrameManager::getTexel (uint frameID, uint cellID)`

Returns taxel value of specified frame and cell ID.

Parameters

<i>frameID</i>	The frame ID.
<i>cellID</i>	The cell ID in range [0, 486]

Returns

The taxel value.

6.17.3.50 `bool FrameManager::getTSFrameAvailable ()`

Checks if a tactile sensor frame is available.

Returns

The state.

6.17.3.51 `float FrameManager::hex2float (std::string & s)`

Converts a hex string to a float using stringstream.

Parameters

<i>s</i>	The hex string.
----------	-----------------

Returns

The converted float.

6.17.3.52 bool FrameManager::isConnectedDSA ()

Checks if the DSA-controller is connected.

Returns

The state.

6.17.3.53 bool FrameManager::isConnectedSDH ()

Checks if the hand is connected.

Returns

The state.

6.17.3.54 bool FrameManager::isSelected (int *cellID*)

Checks if the specified taxel is selected.

Parameters

<i>cellID</i>	The cell ID.
---------------	--------------

Returns

The selection state.

6.17.3.55 void FrameManager::loadFrames (const std::string & *filename*)

Load pressure profile from file.

Parameters

<i>filename</i>	The *.dsa file.
-----------------	-----------------

Returns

void

6.17.3.56 void FrameManager::loadProfile (const std::string & *filename*)

Loads a SDH-2 pressure profile stored in *.dsa files

The file format is xml based and inspired by Weiss's dsa3 format. By removing the additional temperature and joint angle readings from the xml tree the profiles can be opened in the DSA Explorer by Weiss.

Parameters

<i>filename</i>	The filename.
-----------------	---------------

Returns

void

6.17.3.57 void FrameManager::print_tree (const boost::property_tree::ptree & *pt*)

Pretty-prints an entire property tree

Parameters

<i>pt</i>	The property tree.
-----------	--------------------

Returns

void

6.17.3.58 void FrameManager::print_tree (const boost::property_tree::ptree & *pt*, int *level*)

Pretty-prints property tree from a certain level onwards.

Parameters

<i>pt</i>	The property tree.
<i>level</i>	The tree level.

Returns

void

6.17.3.59 void FrameManager::printJointAngleFrame (JointAngleFrame & *jointAngleFrame*)

Pretty-printing of joint angle readings.

Parameters

<i>jointAngle-Frame</i>	The joint angle frame.
-------------------------	------------------------

Returns

void

6.17.3.60 void FrameManager::printTemperatureFrame (TemperatureFrame & *temperatureFrame*)

Pretty-printing of temperature readings.

Parameters

<i>temperature-Frame</i>	The temperature frame.
--------------------------	------------------------

Returns

void

6.17.3.61 void FrameManager::printTSMatrices (uint *frameID*)

Pretty-printing of all tactile sensor matrices.

Parameters

<i>frameID</i>	
----------------	--

Returns

void

6.17.3.62 void FrameManager::printTSMatrix (uint *frameID*, uint *m*)

Pretty-printing of a single tactile sensor matrix.

Parameters

<i>frameID</i>	The temperature frame.
<i>m</i>	The matrix ID.

Returns

void

6.17.3.63 void FrameManager::queryDSAInfo ()

Queries sensor controller and matrix info.

Returns

void

6.17.3.64 void FrameManager::requestJointAngleFrame (bool *record*)

Requests a joint angle frame from the connected SDH-2.

0 : common base axis of finger 0 and 2
1 : proximal axis of finger 0
2 : distal axis of finger 0
3 : proximal axis of finger 1
4 : distal axis of finger 1
5 : proximal axis of finger 2
6 : distal axis of finger 2

Parameters

<i>record</i>	Should the frame be stored?
---------------	-----------------------------

6.17.3.65 void FrameManager::requestTemperatureFrame (bool *record*)

Requests a temperature frame from the connected SDH-2.

Temperatures 0-6: close to axes motors, Temperature 7: FPGA, Temperature 8: Printed circuit board.

Parameters

<i>record</i>	Should the frame be stored?
---------------	-----------------------------

6.17.3.66 void FrameManager::resetOffline ()

Resets frame manager (offline state).

Returns

void

6.17.3.67 void FrameManager::resetOnline ()

Resets frame manager (online state).

Returns

void

6.17.3.68 void FrameManager::selectCell (uint *cellID*, bool *value*)

Select specified taxel.

see [convertCellIndex\(\)](#).

Parameters

<i>cellID</i>	
<i>value</i>	

Returns

void.

6.17.3.69 void FrameManager::setCurrentFrameID (uint *frameID*)

Sets the current frame ID.

Parameters

<i>frameID</i>	The current frame ID.
----------------	-----------------------

Returns

void

6.17.3.70 void FrameManager::setDSA (cDSA * *dsa*)

Sets the DSA-controller. Calls [queryDSAInfo\(\)](#) and initializes frame manager accordingly.

Parameters

<i>dsa</i>	The DSA instance.
------------	-------------------

Returns

void

6.17.3.71 void FrameManager::setDynamicMask (std::vector< bool > & *bitmask*)

Sets dynamic bitmask of selected taxels.

Parameters

<i>bitmask</i>	Bitmask of active taxels.
----------------	---------------------------

6.17.3.72 void FrameManager::setFilterBilateral (int *kernelSize*, double *sigmaColor*, double *sigmaSpace*, int *borderType*)

Enables the Bilateral filter. Just a wrapper around frame processor.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>sigmaColor</i>	The color standard deviation parameter.
<i>sigmaSpace</i>	The spatial standard deviation parameter.
<i>borderType</i>	OpenCV border type.

Returns

void

6.17.3.73 void `FrameManager::setFilterGaussian (int kernelRadius, double sigma, int borderType)`

Enables the Gaussian filter. Just a wrapper around frame processor.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>sigma</i>	The standard deviation.
<i>borderType</i>	OpenCV border type.

Returns

void

6.17.3.74 void `FrameManager::setFilterMedian (int kernelRadius, bool masked)`

Enables the 2D Median filter. Just a wrapper around frame processor.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>masked</i>	Taxels that survived the filtering process retain their original values.

Returns

void

6.17.3.75 void `FrameManager::setFilterMedian3D (bool masked)`

Enables the spatio-temporal 3x3x3 Median filter. Just a wrapper around frame processor.

Parameters

<i>masked</i>	Taxels that survived the filtering process retain their original values.
---------------	--

Returns

void

6.17.3.76 void FrameManager::setFilterMorphological (int *kernelType*, int *kernelRadius*, bool *masked*, int *borderType*)

Enables the opening operation. Just a wrapper around frame processor.

Parameters

<i>kernelType</i>	OpenCV kernel type.
<i>kernelRadius</i>	The filtering kernel' radius.
<i>masked</i>	Taxels that survived the filtering process retain their original values.
<i>borderType</i>	OpenCV border type.

Returns

void

6.17.3.77 void FrameManager::setFilterNone ()

Disable filtering.

Returns

void

6.17.3.78 void FrameManager::setFrameGrabberDSA (FrameGrabberDSA * *fgDSA*)

Sets the DSA frame grabber.

Parameters

<i>fgDSA</i>	The DSA frame grabber.
--------------	------------------------

Returns

void

6.17.3.79 void FrameManager::setFrameGrabberSDH (FrameGrabberSDH * *fgSDH*)

Sets the SDH-2 frame grabber.

Parameters

<i>fgSDH</i>	The SDH frame grabber.
--------------	------------------------

Returns

void

6.17.3.80 void FrameManager::setJointAngleFrameAvailable (bool *value*)

Sets the availability of the joint angle frames.

Parameters

<i>The</i>	state.
------------	--------

Returns

void

6.17.3.81 void FrameManager::setLiveFrame ()

Creates a copy of the current frame, the live frame. Access to live frame has to be synchronized.

Returns

void

6.17.3.82 void FrameManager::setSDH (cSDH * *sdh*)

Sets the SDH-2

Parameters

<i>sdh</i>	The hand instance.
------------	--------------------

Returns

void

6.17.3.83 void FrameManager::setSensitivity (uint *matrixID*, float *sensitivity*)

Sets the sensitivity of the specified matrix.

Parameters

<i>matrixID</i>	The matrix ID.
<i>sensitivity</i>	The matrix sensitivity threshold in range [0.0, 1.0].

Returns

void

6.17.3.84 `bool FrameManager::setSlipReferenceFrame (uint matrixID, uint frameID)`

(Re)sets the reference frame for both translational and rotational slip-detection (offline version).

Parameters

<i>matrixID</i>	The matrix ID.
<i>frameID</i>	The frame ID.

Returns

Success.

6.17.3.85 `bool FrameManager::setSlipReferenceFrameLive (uint matrixID)`

(Re)sets the reference frame for both translational and rotational slip-detection (live version).

Parameters

<i>matrixID</i>	The matrix ID.
-----------------	----------------

Returns

Success.

6.17.3.86 `void FrameManager::setSlipThresholdConsecutive (double thresh)`

Sets the threshold for comparison with previous sensor matrix.

Parameters

<i>thresh</i>	The threshold.
---------------	----------------

Returns

void

6.17.3.87 `void FrameManager::setSlipThresholdReference (double thresh)`

Sets the threshold for comparison with reference sensor matrix.

Parameters

<i>thresh</i>	The theshold.
---------------	---------------

Returns

void

6.17.3.88 void `FrameManager::setThreshold (uint matrixID, float threshold)`

Sets the sensor value threshold.

Note

Program might crash if the connection is choking due to the failing run length encoding in case of noise. This problem is hidden somewhere in the SDH-2's black-box.

Parameters

<i>matrixID</i>	The matrix ID.
<i>threshold</i>	The matrix threshold.

Returns

void

6.17.3.89 void `FrameManager::setTSFrameAvailable (bool value)`

Sets the availability of a tactile sensor frame.

Parameters

<i>The</i>	state.
------------	--------

Returns

void

6.17.3.90 `std::vector< boost::optional< slipResult > > FrameManager::slipResultConsumer ()`

Removes slip detection results from queue. (producer-consumer pattern)

Returns

The last slip-detection result.

6.17.3.91 void FrameManager::slipResultProducer ()

Computes slip detection results and pushes them on queue. (producer-consumer pattern)

Returns

void

6.17.3.92 void FrameManager::storeFrames (const std::string & filename)

Store pressure profile.

Parameters

<i>filename</i>	The *.dsa file.
-----------------	-----------------

Returns

void

6.17.3.93 void FrameManager::storeProfile (const std::string & filename)

Store SDH-2 pressure profile in *.dsa file

The file format is xml based and inspired by Weiss's dsa3 format. By removing the additional temperature and joint angle readings from the xml tree the profiles can be opened in the DSA Explorer by Weiss.

Parameters

<i>filename</i>	The filename.
-----------------	---------------

Returns

void

6.17.3.94 void FrameManager::storeProfileSelection (const string & filename, uint fromIdxTS, uint toIdxTS)

Only stores a selection of tactile sensor frames and corresponding temperature and joint angles (including both limits). See [storeProfile\(\)](#) for comparison.

Parameters

<i>filename</i>	The filename.
<i>fromIdxTS</i>	Frame ID from.
<i>toIdxTS</i>	Frame ID to.

Returns

void

The documentation for this class was generated from the following files:

- framemanager.h
- framemanager.cpp

6.18 FrameManagerWrapper Class Reference

Defines Boost.Python wrappers around the FrameManage class. See Python examples for usage.

Public Member Functions

- **FrameManagerWrapper** (std::string filename)
- **FrameManager** & **get_framemanager** ()
- void **load_profile** (std::string filename)
- uint **get_tsframe_count** ()
- uint64_t **get_tsframe_timestamp** (int frameID)
- np::ndarray **get_tsframe_timestamp_list** ()
- np::ndarray **get_tsframe** (int frameID, int matrixID)
- bp::list **get_tsframe_list** (int frameID)
- void **set_filter_none** ()
- void **set_filter_median** (int kernel_radius, bool masked)
- void **set_filter_gaussian** (int kernel_radius, double sigma)
- void **set_filter_bilateral** (int kernel_radius, double sigma_color, double sigma_space)
- void **set_filter_morphological** (int kernel_type, int kernel_radius, bool masked)
- np::ndarray **get_filtered_tsframe** (uint frameID, int matrixID)
- bp::list **get_filtered_tsframe_list** (int frameID)
- double **get_texel** (uint frameID, uint matrixID, uint x, uint y)
- np::ndarray **get_texel_list** (uint matrixID, uint x, uint y)
- double **get_average_frame** (uint frameID)
- np::ndarray **get_average_frame_list** ()
- double **get_average_matrix** (uint frameID, uint matrixID)
- np::ndarray **get_average_matrix_list** (uint matrixID)
- double **get_min_frame** (uint frameID)
- np::ndarray **get_min_frame_list** ()
- double **get_min_matrix** (uint frameID, uint matrixID)
- np::ndarray **get_min_matrix_list** (uint matrixID)
- double **get_max_frame** (uint frameID)
- np::ndarray **get_max_frame_list** ()
- double **get_max_matrix** (uint frameID, uint matrixID)

- np::ndarray **get_max_matrix_list** (uint matrixID)
- int **get_num_active_cells_frame** (uint frameID)
- int **get_num_active_cells_matrix** (uint frameID, uint matrixID)
- int **get_jointangle_frame_count** ()
- np::ndarray **get_jointangle_frame** (int angleID)
- np::ndarray **get_jointangle_frame_list** ()
- uint64_t **get_jointangle_frame_timestamp** (int angleID)
- np::ndarray **get_jointangle_frame_timestamp_list** ()
- int **get_temperature_frame_count** ()
- np::ndarray **get_temperature_frame** (int tempID)
- np::ndarray **get_temperature_frame_list** ()
- uint64_t **get_temperature_frame_timestamp** (int tempID)
- np::ndarray **get_temperature_frame_timestamp_list** ()
- np::ndarray **get_corresponding_jointangles** (int tsframeID)
- np::ndarray **get_corresponding_jointangles_list** ()
- np::ndarray **get_corresponding_temperatures** (int tsframeID)
- np::ndarray **get_corresponding_temperatures_list** ()

6.18.1 Detailed Description

Defines Boost.Python wrappers around the FrameManage class. See Python examples for usage.

The documentation for this class was generated from the following file:

- framemanager_python.cpp

6.19 FrameProcessor Class Reference

Manages temporal, spatial and spatio-temporal filtering.

```
#include <frameprocessor.h>
```

Public Member Functions

- [FrameProcessor](#) ()
- void [setFrameManager](#) ([FrameManager](#) *fm)
- int [getNumActiveCells](#) (uint frameID)
- int [getMatrixNumActiveCells](#) (uint frameID, uint matrixID)
- void [calcCharacteristics](#) (uint frameID)
- double [getAverage](#) (uint frameID)
- double [getMatrixAverage](#) (uint frameID, uint matrixID)
- double [getMin](#) (uint frameID)
- double [getMatrixMin](#) (uint frameID, uint matrixID)
- double [getMax](#) (uint frameID)

- double [getMatrixMax](#) (uint frameID, uint matrixID)
- void [setFilterNone](#) ()
- void [setFilterMedian](#) (int kernelRadius, bool masked)
- void [setFilterMedian3D](#) (bool masked)
- void [setFilterGaussian](#) (int kernelRadius, double sigma, int borderType)
- void [setFilterBilateral](#) (int kernelRadius, double sigmaColor, double sigmaSpace, int borderType)
- void [setFilterOpening](#) (int kernelType, int kernelRadius, bool masked, int borderType)
- FilterType [getFilterType](#) ()
- double [calcGaussianSigma](#) (int kernelRadius)
- void [applyFilter](#) (TSFrame *tsFrame, int frameID)

6.19.1 Detailed Description

Manages temporal, spatial and spatio-temporal filtering.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 FrameProcessor::FrameProcessor ()

Constructor.

6.19.3 Member Function Documentation

6.19.3.1 void FrameProcessor::applyFilter (TSFrame * *tsFrame*, int *frameID*)

Performs the actual filtering based on the current filter settings.

Parameters

<i>tsFrame</i>	Pointer to the tactile sensor frame.
<i>frameID</i>	The frame ID.

Returns

void

6.19.3.2 void FrameProcessor::calcCharacteristics (uint *frameID*)

Calculates all characteristic values at once in a single iteration. Characteristic values are: Per matrix as well as per frame averages, minimum and maximum values

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

void

6.19.3.3 double FrameProcessor::getAverage (uint *frameID*)

Returns frame average. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

void

6.19.3.4 FilterType FrameProcessor::getFilterType ()

Getter: filter type.

Returns

The filter type.

6.19.3.5 double FrameProcessor::getMatrixAverage (uint *frameID*, uint *matrixID*)

Returns the matrix average. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
<i>matrixID</i>	The matrix ID.

Returns

void

6.19.3.6 double FrameProcessor::getMatrixMax (uint *frameID*, uint *matrixID*)

Returns the matrix maximum value. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
<i>matrixID</i>	The matrix ID.

Returns

void

6.19.3.7 double FrameProcessor::getMatrixMin (uint *frameID*, uint *matrixID*)

Returns the matrix minimum value. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
<i>matrixID</i>	The matrix ID.

Returns

void

6.19.3.8 int FrameProcessor::getMatrixNumActiveCells (uint *frameID*, uint *matrixID*)

Returns the number of active taxels of the specified matrix.

Parameters

<i>frameID</i>	The frame ID.
<i>matrixID</i>	The matrix ID.

Returns

The number of active taxels.

6.19.3.9 double FrameProcessor::getMax (uint *frameID*)

Returns frame maximum value. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

void

6.19.3.10 double FrameProcessor::getMin (uint *frameID*)

Returns frame minimum value. See [calcCharacteristics\(\)](#).

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

void

6.19.3.11 `int FrameProcessor::getNumActiveCells (uint frameID)`

Returns the number of active taxels of the entire frame

Parameters

<i>frameID</i>	The frame ID.
----------------	---------------

Returns

The number of active taxels.

6.19.3.12 `void FrameProcessor::setFilterBilateral (int kernelRadius, double sigmaColor, double sigmaSpace, int borderType)`

Enables the Bilateral filter.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>sigmaColor</i>	The color standard deviation parameter.
<i>sigmaSpace</i>	The spatial standard deviation parameter.
<i>borderType</i>	OpenCV border type.

Returns

void

6.19.3.13 `void FrameProcessor::setFilterGaussian (int kernelRadius, double sigma, int borderType)`

Enables the Gaussian filter.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>sigma</i>	The standard deviation.
<i>borderType</i>	OpenCV border type.

Returns

void

6.19.3.14 void FrameProcessor::setFilterMedian (int *kernelRadius*, bool *masked*)

Enables the 2D Median filter.

Parameters

<i>kernelRadius</i>	The filtering kernel' radius.
<i>masked</i>	Taxels that survived the filtering process retain their original values.

Returns

void

6.19.3.15 void FrameProcessor::setFilterMedian3D (bool *masked*)

Enables the spatio-temporal 3x3x3 Median filter.

Parameters

<i>masked</i>	Taxels that survived the filtering process retain their original values.
---------------	--

Returns

void

6.19.3.16 void FrameProcessor::setFilterNone ()

Disables filtering.

6.19.3.17 void FrameProcessor::setFilterOpening (int *kernelType*, int *kernelRadius*, bool *masked*, int *borderType*)

Enables the opening operation.

Parameters

<i>kernelType</i>	OpenCV kernel type.
<i>kernelRadius</i>	The filtering kernel' radius.
<i>masked</i>	Taxels that survived the filtering process retain their original values.
<i>borderType</i>	OpenCV border type.

Returns

void

6.19.3.18 void FrameProcessor::setFrameManager (FrameManager * *fm*)

Sets the frame manager.

Parameters

<i>fm</i>	The frame manager.
-----------	--------------------

Returns

void

The documentation for this class was generated from the following files:

- frameprocessor.h
- frameprocessor.cpp

6.20 guiChart Class Reference

The chart containing the the zoom, crop and export buttons as well as the graph. Manages the dataset before it is displayed in the graph.

```
#include <guiChart.h>
```

Public Member Functions

- **guiChart** ([Controller](#) *c, [guiMain](#) *gui)
- void **initDataset** ()
- void **updateDataset** ()
- void **setMarkerPosition** (int frameID)
- bool **getActiveSelection** ()
- uint **getSelectionFrom** ()
- uint **getSelectionTo** ()

Protected Member Functions

- void **on_button_zoom_in_clicked** ()
- void **on_button_zoom_out_clicked** ()
- bool **on_slider_value_changed** (Gtk::ScrollType type, double value)
- void **on_button_crop_clicked** ()
- void **on_button_export_clicked** ()
- void **on_checkbutton_selection_clicked** ()

6.20.1 Detailed Description

The chart containing the the zoom, crop and export buttons as well as the graph. Manages the dataset before it is displayed in the graph.

6.20.2 Member Function Documentation

6.20.2.1 void guiChart::initDataset ()

Init chart

6.20.2.2 void guiChart::updateDataset ()

Only collect/copy missing data series

The documentation for this class was generated from the following files:

- guiChart.h
- guiChart.cpp

6.21 guiController Class Reference

GUI to control the SDH-2.

```
#include <guiController.h>
```

Public Member Functions

- **guiController** ([Controller](#) *c)
- void **connectSDH** ()
- void **disconnectSDH** ()
- void **connectDSA** ()
- void **disconnectDSA** ()

Protected Member Functions

- void **worker_grasp** (int graspID, double closeRatio, double velocity)
- void **on_worker_grasp_done** ()
- void **worker_grasp_reactive** (int graspID, double velocity, double limit)
- void **on_worker_grasp_reactive_done** ()
- void **worker_grasp_slip** (int graspID, double velocity, double limitLow, double limitHigh, double stepSize)
- void **on_worker_grasp_slip_done** ()
- void **on_button_pause_SDH_clicked** ()
- void **on_button_stop_SDH_clicked** ()
- void **on_button_record_SDH_clicked** ()
- void **on_checkbox_temperature_clicked** ()
- void **on_checkbox_joint_angles_clicked** ()
- void **on_button_pause_DSA_clicked** ()
- void **on_button_stop_DSA_clicked** ()

- void **on_button_record_DSA_clicked** ()
- void **on_combo_grasp_changed** ()
- void **on_button_grasp_clicked** ()
- void **on_button_grasp_reactive_clicked** ()
- void **on_button_grasp_slip_clicked** ()
- void **on_button_relax_clicked** ()
- bool **on_slider_close_clicked** (GdkEventButton *event)
- bool **on_slider_close_released** (GdkEventButton *event)
- bool **on_slider_close_value_changed** (Gtk::ScrollType type, double value)
- bool **on_slider_velocity_clicked** (GdkEventButton *event)
- bool **on_slider_velocity_released** (GdkEventButton *event)
- bool **on_slider_velocity_value_changed** (Gtk::ScrollType type, double value)
- bool **on_slider_reactive_clicked** (GdkEventButton *event)
- bool **on_slider_reactive_released** (GdkEventButton *event)
- bool **on_slider_reactive_value_changed** (Gtk::ScrollType type, double value)
- bool **on_slider_sensitivity_clicked** (GdkEventButton *event)
- bool **on_slider_sensitivity_released** (GdkEventButton *event)
- bool **on_slider_sensitivity_value_changed** (Gtk::ScrollType type, double value)

- bool **on_slider_threshold_clicked** (GdkEventButton *event)
- bool **on_slider_threshold_released** (GdkEventButton *event)
- bool **on_slider_threshold_value_changed** (Gtk::ScrollType type, double value)

- void **on_button_threshold_clicked** ()
- void **on_button_threshold_reset_clicked** ()

Protected Attributes

- [Controller](#) * **controller**
- [FrameManager](#) * **frameManager**
- [FrameProcessor](#) * **frameProcessor**
- [FrameGrabberDSA](#) * **frameGrabberDSA**
- [FrameGrabberSDH](#) * **frameGrabberSDH**
- Gtk::Image **m_Image_Play_SDH**
- Gtk::Image **m_Image_Pause_SDH**
- Gtk::Image **m_Image_Stop_SDH**
- Gtk::Image **m_Image_Record_SDH**
- Gtk::Image **m_Image_Play_DSA**
- Gtk::Image **m_Image_Pause_DSA**
- Gtk::Image **m_Image_Stop_DSA**
- Gtk::Image **m_Image_Record_DSA**
- Gtk::VBox **m_VBox_Left_Sidebar**
- Gtk::Frame **m_Frame_SDH**
- Gtk::Frame **m_Frame_DSA**
- Gtk::VBox **m_VBox_SDH**
- Gtk::VBox **m_VBox_DSA**

- Gtk::Frame **m_Frame_Recorder_SDH**
- Gtk::Frame **m_Frame_Recorder_DSA**
- Gtk::VBox **m_VBox_Recorder_SDH**
- Gtk::VBox **m_VBox_Recorder_DSA**
- Gtk::HButtonBox **m_ButtonBox_Recording_SDH**
- Gtk::Button **m_Button_Pause_SDH**
- Gtk::Button **m_Button_Stop_SDH**
- Gtk::Button **m_Button_Record_SDH**
- bool **recorder_paused_SDH**
- bool **recorder_recording_SDH**
- Gtk::CheckButton **m_CheckButton_Temperature**
- Gtk::CheckButton **m_CheckButton_JointAngles**
- Gtk::HButtonBox **m_ButtonBox_Recording_DSA**
- Gtk::Button **m_Button_Pause_DSA**
- Gtk::Button **m_Button_Stop_DSA**
- Gtk::Button **m_Button_Record_DSA**
- bool **recorder_paused_DSA**
- bool **recorder_recording_DSA**
- Gtk::Frame **m_Frame_Grasp**
- Gtk::VBox **m_VBox_Grasp**
- Gtk::HBox **m_HBox_Grasp**
- Gtk::ComboBoxText **m_Combo_Grasp**
- Gtk::Button **m_Button_Grasp**
- Gtk::Button **m_Button_Grasp_Reactive**
- bool **m_ToggleButton_Grasp_Slip_Failed**
- Gtk::ToggleButton **m_ToggleButton_Grasp_Slip**
- Gtk::Button **m_Button_Relax**
- Glib::Thread * **m_Thread_Grasp**
- Glib::Dispatcher **m_Thread_Grasp_Dispatcher**
- Glib::Thread * **m_Thread_Grasp_Reactive**
- Glib::Dispatcher **m_Thread_Grasp_Reactive_Dispatcher**
- bool **m_stop_thread_grasp_slip**
- Glib::Mutex **m_mutex_thread_grasp_slip**
- Glib::Thread * **m_Thread_Grasp_Slip**
- Glib::Dispatcher **m_Thread_Grasp_Slip_Dispatcher**
- Gtk::Frame **m_Frame_Close**
- Gtk::Adjustment **m_Adjustment_Close**
- Gtk::HScale **m_Slider_Close**
- Gtk::Frame **m_Frame_Velocity**
- Gtk::Adjustment **m_Adjustment_Velocity**
- Gtk::HScale **m_Slider_Velocity**
- Gtk::Frame **m_Frame_Reactive**
- Gtk::Adjustment **m_Adjustment_Reactive**
- Gtk::HScale **m_Slider_Reactive**
- Gtk::Frame **m_Frame_Sensitivity**
- Gtk::Adjustment **m_Adjustment_Sensitivity**

- Gtk::HScale **m_Slider_Sensitivity**
- Gtk::Label **m_Label_Sensitivity**
- float **m_sensitivity**
- Gtk::Frame **m_Frame_Threshold**
- Gtk::VBox **m_VBox_Threshold**
- Gtk::Adjustment **m_Adjustment_Threshold**
- Gtk::HScale **m_Slider_Threshold**
- Gtk::Label **m_Label_Threshold**
- UInt16 **m_threshold**
- Gtk::HButtonBox **m_ButtonBox_Threshold**
- Gtk::Button **m_Button_Threshold**
- Gtk::Button **m_Button_Threshold_Reset**

6.21.1 Detailed Description

GUI to control the SDH-2.

6.21.2 Member Function Documentation

6.21.2.1 void guiController::on_button_threshold_clicked () [protected]

Measure current temperature and set sensor threshold to individually calibrated values
The sensitivity of all matrices is subsequently set to 1.0 In order to get rid of ghosting
have a look at setFilterOpening() of the frame processor

The documentation for this class was generated from the following files:

- guiController.h
- guiController.cpp

6.22 guiGraph Class Reference

The graph.

```
#include <guiGraph.h>
```

Public Member Functions

- **guiGraph** (**Controller** *c, **guiMain** *gui, Timeseries &collection)
- void **updateSamples** (const Timeseries &inSample)
- void **setZoom** (int zoom)
- void **setMarkerPosition** (int frameID)
- void **setActiveSelection** (bool active)
- bool **getActiveSelection** ()
- void **moveLeftBoundary** (int pos)

- int **getLeftBoundary** ()
- double **getStepSize** ()
- int **getSampleRange** ()
- int **getSelectionFrom** ()
- int **getSelectionTo** ()

Protected Member Functions

- virtual bool **on_expose_event** (GdkEventExpose *event)
- virtual bool **on_button_press_event** (GdkEventButton *event)
- virtual bool **on_button_release_event** (GdkEventButton *event)
- virtual bool **on_motion_notify_event** (GdkEventMotion *event)

6.22.1 Detailed Description

The graph.

The drawing is rather complex and adopts to the number of samples per pixel. In the overview mode, lines between samples are drawn individually. Otherwise a pyramidal linear subsampling scheme is applied to draw the time series.

6.22.2 Member Function Documentation

6.22.2.1 bool guiGraph::on_button_press_event (GdkEventButton * *event*)
[protected, virtual]

Mouse button pressed

6.22.2.2 bool guiGraph::on_button_release_event (GdkEventButton * *event*)
[protected, virtual]

Mouse button released

6.22.2.3 bool guiGraph::on_motion_notify_event (GdkEventMotion * *event*)
[protected, virtual]

Moving the mouse with pressed buttons

6.22.2.4 void guiGraph::setMarkerPosition (int *frameID*)

Map marker from sample- to pixel space

The documentation for this class was generated from the following files:

- guiGraph.h
- guiGraph.cpp

6.23 guiMain Class Reference

The main window. Inherits from `Gtk::Window`.

```
#include <guiMain.h>
```

Public Member Functions

- **guiMain** ([Controller](#) *controller)
- void **resetGUIOnline** ()
- void [resetGUIOffline](#) ()
- void [updateGUIOffline](#) ()
- void **setCurrentFrame** (int frameID)
- void **updateDataset** ()
- bool **getActiveSelection** ()
- uint **getSelectionFrom** ()
- uint **getSelectionTo** ()
- void **setCharacteristics** (std::vector< std::vector< int > > c)
- std::vector< std::vector< int > > **getCharacteristics** ()
- void **saveCurrentFramePDF** ()

Protected Member Functions

- void **on_menu_take_screenshot_2D_clicked** ()
- void **on_menu_take_screenshot_3D_clicked** ()
- void **on_screenshot_delete_clicked** ()
- void **on_menu_connect_SDH** ()
- void **on_menu_connect_DSA** ()
- void [on_menu_new_profile](#) ()
- void [on_menu_load_profile](#) ()
- void **on_menu_save_profile_as** ()
- void **on_menu_file_quit** ()
- void **on_menu_show_tools** ()
- void **on_menu_show_slip_detection** ()
- void **on_menu_show_sensor_view** ()
- void **on_menu_show_chart_view** ()
- void **on_menu_show_tree_view** ()
- void [embedPython](#) ()
- void **on_menu_classify** ()
- void **on_notebook_switch_page** (GtkNotebookPage *page, guint page_num)
- void **on_vpaned_size_allocate** (Gtk::Allocation &allocation)
- void **on_vpaned_realize** ()
- void **on_resize_notify** (GdkEventConfigure *event)
- bool **on_tools_delete_clicked** (GdkEventAny *event)
- bool **on_slip_detection_delete_clicked** (GdkEventAny *event)
- virtual bool **on_key_press_event** (GdkEventKey *event)
- virtual bool **on_key_release_event** (GdkEventKey *event)

Protected Attributes

- [Controller](#) * **m_controller**
- [FrameManager](#) * **m_frameManager**
- [FrameProcessor](#) * **m_frameProcessor**
- [FeatureExtraction](#) **m_featureExtractor**
- uint **current_frame**
- Gtk::VBox **m_VBox_Main**
- Gtk::MenuBar **m_Menubar**
- Gtk::Toolbar **m_Toolbar**
- Gtk::ToggleToolButton **m_ToggleToolButton_Connect_SDH**
- Gtk::ToggleToolButton **m_ToggleToolButton_Connect_DSA**
- bool **m_ToggleToolButton_Connect_SDH_pressed**
- bool **m_ToggleToolButton_Connect_DSA_pressed**
- Gtk::ToggleToolButton **m_ToggleToolButton_Tools**
- Gtk::ToggleToolButton **m_ToggleToolButton_Slip_Detection**
- Gtk::ToggleToolButton **m_ToggleToolButton_Sensor_View**
- Gtk::ToggleToolButton **m_ToggleToolButton_Chart_View**
- Gtk::ToggleToolButton **m_ToggleToolButton_Tree_View**
- Gtk::HBox **m_HBox_Main**
- Gtk::VBox **m_VBox_Right_Sidebar**
- Gtk::VBox **m_VBox_Renderer**
- [guiController](#) * **m_Frame_Controller**
- bool **showSensorView**
- bool **showChartView**
- bool **showTreeView**
- bool **m_pythonEmbedded**
- bp::object **m_main**
- bp::object **m_global**
- **Renderer renderer**
- Gtk::VPaned **m_VPaned_Views**
- int **m_VPaned_Views_Divider_Pos**
- double **m_VPaned_Views_Ratio**
- bool **m_resized**
- Gtk::Notebook **m_Notebook_Renderer**
- Gtk::Frame **m_Frame_Renderer2D**
- [guiRenderer2D](#) * **m_guiRenderer2D**
- Gtk::Frame **m_Frame_Renderer3D**
- [guiRenderer3D](#) * **m_guiRenderer3D**
- [guiSeekBar](#) * **m_guiSeekBar**
- [guiChart](#) * **m_guiChart**
- [guiTreeView](#) * **m_guiTreeView**
- [guiTools](#) * **m_guiTools**
- [guiSlipDetection](#) * **m_guiSlipDetection**
- [guiScreenshot](#) * **m_guiScreenshot**
- std::vector< std::vector< int > > **characteristics**

6.23.1 Detailed Description

The main window. Inherits from Gtk::Window.

6.23.2 Member Function Documentation

6.23.2.1 void guiMain::embedPython () [protected]

Embed python interpreter, load trained state of SVM and classify feature vector

6.23.2.2 void guiMain::on_menu_load_profile () [protected]

File chooser for *.dsa pressure profile file

6.23.2.3 void guiMain::on_menu_new_profile () [protected]

Reset [FrameManager](#)

6.23.2.4 void guiMain::resetGUIOffline ()

Reset to initial state (no frames available)

6.23.2.5 void guiMain::updateGUIOffline ()

Update state (frames available)

The documentation for this class was generated from the following files:

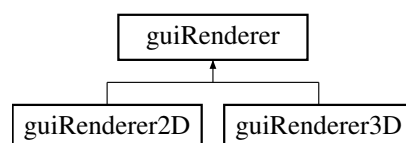
- guiMain.h
- guiMain.cpp

6.24 guiRenderer Class Reference

Base class for [guiRenderer2D](#) and [guiRenderer3D](#).

```
#include <guiRenderer.h>
```

Inheritance diagram for guiRenderer:



Public Member Functions

- **guiRenderer** ([FrameManager](#) *fm)
- **guiRenderer** ([FrameManager](#) *fm, [guiMain](#) *gui)
- void **startRendering** (bool live)
- void **stopRendering** ()
- virtual void **invalidate** ()
- virtual void **update** ()
- virtual void **renderFrame** ()
- virtual void **renderFrame** (uint frameID)
- [RGB](#) **determineColor** (float value)
- virtual bool **on_key_press_event** (GdkEventKey *event)
- virtual bool **on_key_release_event** (GdkEventKey *event)

Public Attributes

- [FrameManager](#) * **m_frameManager**
- [FrameProcessor](#) * **m_frameProcessor**
- [guiMain](#) * **m_mainGUI**
- [Colormap](#) **m_colormap**
- bool **m_liveMode**
- bool **m_isRendering**

Protected Member Functions

- virtual bool **on_idle** ()
- virtual bool **on_map_event** (GdkEventAny *event)
- virtual bool **on_unmap_event** (GdkEventAny *event)
- virtual bool **on_visibility_notify_event** (GdkEventVisibility *event)

Protected Attributes

- sigc::connection **m_ConnectionIdle**

6.24.1 Detailed Description

Base class for [guiRenderer2D](#) and [guiRenderer3D](#).

The documentation for this class was generated from the following files:

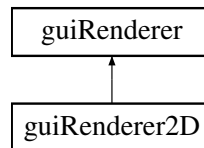
- [guiRenderer.h](#)
- [guiRenderer.cpp](#)

6.25 guiRenderer2D Class Reference

Renders visualization of tactile sensor profiles with the help of Cairo, a vector graphics library.

```
#include <guiRenderer2D.h>
```

Inheritance diagram for guiRenderer2D:



Public Member Functions

- **guiRenderer2D** ([FrameManager](#) *fm)
- **guiRenderer2D** ([FrameManager](#) *fm, [guiMain](#) *gui)
- virtual void **renderFrame** ()
- virtual void **renderFrame** (uint frameID)
- void **drawMatrices** (const Cairo::RefPtr< Cairo::Context > &cr, int width, int height, bool screenshot)
- void **takeScreenshot** (const string &filename)
- void **init** ()

Protected Member Functions

- virtual void **invalidate** ()
- virtual void **update** ()
- virtual bool **on_expose_event** (GdkEventExpose *event)
- virtual bool **on_button_press_event** (GdkEventButton *event)
- virtual bool **on_button_release_event** (GdkEventButton *event)
- virtual bool **on_motion_notify_event** (GdkEventMotion *event)

6.25.1 Detailed Description

Renders visualization of tactile sensor profiles with the help of Cairo, a vector graphics library.

Not very suitable for real-time rendering, but great for rendering PDFs.

The documentation for this class was generated from the following files:

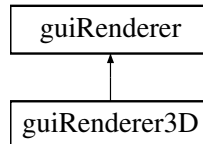
- guiRenderer2D.h
- guiRenderer2D.cpp

6.26 guiRenderer3D Class Reference

Renders visualization of tactile sensor profiles in OpenGL.

```
#include <guiRenderer3D.h>
```

Inheritance diagram for guiRenderer3D:



Public Member Functions

- EIGEN_MAKE_ALIGNED_OPERATOR_NEW **guiRenderer3D** ([FrameManager](#) *frameManager)
- **guiRenderer3D** ([FrameManager](#) *frameManager, [guiMain](#) *gui)
- virtual void **renderFrame** ()
- virtual void **renderFrame** (uint frameID)
- void **takeScreenshot** (std::string filename)
- void **takeScreenshot** (int width, int height, std::string filename)
- void **setOffscreenSize** (int width, int height)
- void **init** ()
- bool **on_key_press_event** (GdkEventKey *event)
- bool **on_key_release_event** (GdkEventKey *event)

Protected Member Functions

- virtual void **invalidate** ()
- virtual void **update** ()
- virtual void **on_realize** ()
- virtual bool **on_expose_event** (GdkEventExpose *event)
- virtual bool **on_configure_event** (GdkEventConfigure *event)
- virtual bool **on_button_press_event** (GdkEventButton *event)
- virtual bool **on_button_release_event** (GdkEventButton *event)
- virtual bool **on_motion_notify_event** (GdkEventMotion *event)

6.26.1 Detailed Description

Renders visualization of tactile sensor profiles in OpenGL.

Uses the old OpenGL immediate mode. Given the (recorded) joint angles the computed miniball can be visualized.

6.26.2 Member Function Documentation

6.26.2.1 `bool guiRenderer3D::on_button_press_event (GdkEventButton * event)`
[protected, virtual]

Mouse button pressed

6.26.2.2 `bool guiRenderer3D::on_button_release_event (GdkEventButton * event)`
[protected, virtual]

Mouse button released

6.26.2.3 `bool guiRenderer3D::on_configure_event (GdkEventConfigure * event)`
[protected, virtual]

Called when resizing

6.26.2.4 `bool guiRenderer3D::on_expose_event (GdkEventExpose * event)`
[protected, virtual]

Draw scene

6.26.2.5 `bool guiRenderer3D::on_motion_notify_event (GdkEventMotion * event)`
[protected, virtual]

Moving the mouse with pressed buttons

6.26.2.6 `void guiRenderer3D::on_realize ()` [protected, virtual]

Initialization (called only once)

The documentation for this class was generated from the following files:

- guiRenderer3D.h
- guiRenderer3D.cpp

6.27 guiScreenshot Class Reference

Take screen shot GUI.

```
#include <guiScreenshot.h>
```

Public Member Functions

- **guiScreenshot** ([Controller](#) *C, [guiRenderer3D](#) *renderer, uint from, uint to)

Protected Member Functions

- void **on_button_render_clicked** ()
- void **on_button_close_clicked** ()

Protected Attributes

- Gtk::Button **m_Button_Render**
- Gtk::Button **m_Button_Close**
- Gtk::Label **m_Label_Width**
- Gtk::Label **m_Label_Height**
- Gtk::Label **m_Label_From**
- Gtk::Label **m_Label_To**
- Gtk::Label **m_Label_From_Value**
- Gtk::Label **m_Label_To_Value**
- Gtk::Adjustment **m_Adjustment_Width**
- Gtk::SpinButton **m_SpinButton_Width**
- Gtk::Adjustment **m_Adjustment_Height**
- Gtk::SpinButton **m_SpinButton_Height**
- Gtk::Table **m_Table**
- Gtk::HButtonBox **m_ButtonBox_Dialog**
- Gtk::VBox **m_VBox_Dialog**

6.27.1 Detailed Description

Take screen shot GUI.

The documentation for this class was generated from the following files:

- guiScreenshot.h
- guiScreenshot.cpp

6.28 guiSeekBar Class Reference

The seekbar.

```
#include <guiSeekBar.h>
```


Public Member Functions

- **guiSeekBar** ([Controller](#) *c, [guiMain](#) *gui)
- void **initSeekBar** ()
- void **resetSeekBar** ()
- void **setSliderPosition** (int frameID)

Protected Member Functions

- bool **on_idle** ()
- bool **on_signal_timeout** ()
- void **on_button_play_clicked** ()
- void **on_button_next_clicked** ()
- void **on_button_prev_clicked** ()
- bool **on_slider_clicked** (GdkEventButton *event)
- bool **on_slider_released** (GdkEventButton *event)
- bool **on_slider_value_changed** (Gtk::ScrollType type, double value)

6.28.1 Detailed Description

The seekbar.

6.28.2 Member Function Documentation

6.28.2.1 bool guiSeekBar::on_signal_timeout () [protected]

this timer callback function is called every 1/fps seconds and updates the the current video frame (Gtk::Image) periodically

6.28.2.2 void guiSeekBar::setSliderPosition (int *frameID*)

Update navigation bar

The documentation for this class was generated from the following files:

- guiSeekBar.h
- guiSeekBar.cpp

6.29 guiSlipDetection Class Reference

The Slip-detection control GUI.

```
#include <guiSlipDetection.h>
```

Public Member Functions

- **guiSlipDetection** ([Controller](#) *c, [guiMain](#) *gui)
- void **clearTrajectory** (uint m)
- void **setModeOnline** ()
- void **setModeOffline** ()
- bool **runSlipDetectionOnline** ()
- bool **drawTrajectoryOnline** (uint m)
- void **setCurrentFrameOffline** (uint frameID)
- void **runSlipDetectionOffline** (uint m, uint startFrame, uint stopFrame)
- bool **drawTrajectoryOffline** (uint m, uint currentFrameID)

Protected Member Functions

- bool **on_slider_threshold_reference_value_changed** (Gtk::ScrollType type, double value)
- bool **on_slider_threshold_reference_clicked** (GdkEventButton *event)
- bool **on_slider_threshold_reference_released** (GdkEventButton *event)
- bool **on_slider_threshold_consecutive_value_changed** (Gtk::ScrollType type, double value)
- bool **on_slider_threshold_consecutive_clicked** (GdkEventButton *event)
- bool **on_slider_threshold_consecutive_released** (GdkEventButton *event)
- void **on_checkbutton_enable_clicked** (uint m)
- void **on_button_set_reference_clicked** (uint m)
- void **on_togglebutton_details_clicked** (uint m)
- bool **on_delete_detail_clicked** (GdkEventAny *event, uint m)

6.29.1 Detailed Description

The Slip-detection control GUI.

The documentation for this class was generated from the following files:

- guiSlipDetection.h
- guiSlipDetection.cpp

6.30 guiSlipDetectionMultiPlot Class Reference

Combines individual widgets.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- void **drawTrajectory** ([slipResult](#) &slip, std::deque< slip_trajectory > &slipvectors, std::deque< double > &slipangles, uint currentFrameID)
- void **updateTrajectory** ([slipResult](#) &slip, std::deque< slip_trajectory > &slipvectors, std::deque< double > &slipangles, uint currentFrameID)
- void **drawTrajectoryReference** ([slipResult](#) &slip, std::deque< slip_trajectory > &slipvectors, std::deque< double > &slipangles, uint currentFrameID)
- void **reset** ()
- void **setAxisLimits** (int x_lower, int x_upper, int y_lower, int y_upper)

Protected Attributes

- Gtk::VBox **m_VBox_Main**
- Gtk::HBox **m_HBox_Upper**
- Gtk::HBox **m_HBox_Lower**
- Gtk::HBox **m_HBox_UpperLeft**
- Gtk::HBox **m_HBox_UpperRight**
- Gtk::HBox **m_HBox_LowerLeft**
- Gtk::HBox **m_HBox_LowerRight**
- Gtk::AspectFrame **m_AspectFrame_UpperLeft**
- Gtk::AspectFrame **m_AspectFrame_UpperRight**
- Gtk::AspectFrame **m_AspectFrame_LowerLeft**
- Gtk::AspectFrame **m_AspectFrame_LowerRight**

6.30.1 Detailed Description

Combines individual widgets.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.31 guiTools Class Reference

Dialog for OpenCV filters.

```
#include <guiTools.h>
```

Public Member Functions

- **guiTools** ([Controller](#) *c, [guiMain](#) *gui)

6.31.1 Detailed Description

Dialog for OpenCV filters.

The documentation for this class was generated from the following files:

- guiTools.h
- guiTools.cpp

6.32 guiTreeView Class Reference

Tree view of matrix characteristics. Follows the MVC pattern.

```
#include <guiTreeView.h>
```

Classes

- struct **ModelColumns**

Public Member Functions

- **guiTreeView** ([Controller](#) *c, [guiMain](#) *gui)
- void **init** ()
- void **updateCharacteristics** ()

Protected Member Functions

- void **on_cell_toggled** (const Glib::ustring &path_string)
- void **notifyMain** ()

6.32.1 Detailed Description

Tree view of matrix characteristics. Follows the MVC pattern.

The documentation for this class was generated from the following files:

- guiTreeView.h
- guiTreeView.cpp

6.33 HSL Class Reference

Simple [HSL](#) color management class. Hue, Saturation and Luminance are doubles in the range [0.0, 1.0].

```
#include <colormap.h>
```

Public Member Functions

- [HSL](#) (double hue=0.0, double saturation=0.0, double luminance=0.0)

Public Attributes

- union {
 double **data** [3]
 struct {
 double **h**
 double **s**
 double **l**
 }
 };

6.33.1 Detailed Description

Simple [HSL](#) color management class. Hue, Saturation and Luminance are doubles in the range [0.0, 1.0].

6.33.2 Constructor & Destructor Documentation

6.33.2.1 `HSL::HSL (double hue = 0.0, double saturation = 0.0, double luminance = 0.0)`
`[inline]`

Constructor

Parameters

<i>h,s,l</i>	Hue, saturation and luminance in the range [0.0, 1.0].
--------------	--

The documentation for this class was generated from the following file:

- colormap.h

6.34 JointAngleFrame Struct Reference

Joint angles in accordance with SDHLibrary.

```
#include <framemanager.h>
```

Public Attributes

- `std::vector< double >` **angles**
- `uint64_t` **timestamp**

6.34.1 Detailed Description

Joint angles in accordance with SDHLibrary.

The documentation for this struct was generated from the following file:

- framemanager.h

6.35 matrixInfo Struct Reference

Individual sensor matrices (see dsa.h)

```
#include <framemanager.h>
```

Public Attributes

- UInt8 **uid** [6]
- uint **hw_revision**
- uint **cells_x**
- uint **cells_y**
- float **texel_width**
- float **texel_height**
- float **matrix_center_x**
- float **matrix_center_y**
- float **matrix_center_z**
- float **matrix_theta_x**
- float **matrix_theta_y**
- float **matrix_theta_z**
- float **fullscale**
- std::vector< bool > **static_mask**
- std::vector< bool > **dynamic_mask**
- uint **num_cells**
- uint **texel_offset**

6.35.1 Detailed Description

Individual sensor matrices (see dsa.h)

The documentation for this struct was generated from the following file:

- framemanager.h

6.36 Miniball::Miniball< CoordAccessor > Class Template Reference

Public Types

- typedef std::list< Pit >::const_iterator **SupportPointIterator**

Public Member Functions

- **Miniball** (int d_, Pit begin, Pit end, [CoordAccessor](#) ca=[CoordAccessor](#)())
- const NT * **center** () const
- NT **squared_radius** () const
- int **nr_support_points** () const
- SupportPointIterator **support_points_begin** () const
- SupportPointIterator **support_points_end** () const
- NT **relative_error** (NT &subopt) const
- bool **is_valid** (NT tol=NT(10)*std::numeric_limits< NT >::epsilon()) const
- double **get_time** () const

```
template<typename CoordAccessor> class Miniball::Miniball< CoordAccessor >
```

The documentation for this class was generated from the following file:

- Miniball.hpp

6.37 NiceScale Class Reference

Pretty axis tick labels. Graphics Gems, Volume 1 by Andrew S. Glassner.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- **NiceScale** (double min, double max, int maxTicks)
- void **computeScale** (double min, double max, int maxTicks)
- double **getNiceMin** ()
- double **getNiceMax** ()
- double **getTickSpacing** ()
- int **getNumTicks** ()

6.37.1 Detailed Description

Pretty axis tick labels. Graphics Gems, Volume 1 by Andrew S. Glassner.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.38 NumPyArrayData< T > Class Template Reference

Public Member Functions

- **NumPyArrayData** (const np::ndarray &arr)
- T * **data** ()
- const Py_intptr_t * **strides** ()
- T & **operator()** (int i)
- T & **operator()** (int i, int j)
- T & **operator()** (int i, int j, int k)
- T & **operator()** (int i, int j, int k, int l)

```
template<typename T> class NumPyArrayData< T >
```

The documentation for this class was generated from the following file:

- NumPyArrayData.h

6.39 Orientation Class Reference

Visualizes current orientation.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- void **reset** ()
- void **drawAxes** (const Cairo::RefPtr< Cairo::Context > &cr, int width, int height)
- bool **drawOrientation** (bool success, double angle, double lambda1, double lambda2, double skew_x, double skew_y)

Protected Member Functions

- virtual bool **on_expose_event** (GdkEventExpose *event)

6.39.1 Detailed Description

Visualizes current orientation.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.40 OrientationTrajectory Class Reference

Visualizes rotation trajectory.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- **OrientationTrajectory** (std::deque< double > &slipangles)
- void **reset** ()
- void **resetAxisLimits** (int x_lower, int x_upper, int y_lower, int y_upper)
- void **drawBackgroundSurface** ()
- void **drawAxes** (const Cairo::RefPtr< Cairo::Context > &cr, int width, int height)
- bool **drawTrajectory** (std::deque< double > &slipangles, uint currentFrameID)
- bool **updateTrajectory** (std::deque< double > &slipangles, uint currentFrameID)

Protected Member Functions

- virtual bool **on_expose_event** (GdkEventExpose *event)

6.40.1 Detailed Description

Visualizes rotation trajectory.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.41 RGB Class Reference

Simple [RGB](#) color management class. Red Green Blue are floats in the range [0.0, 1.0].

```
#include <colormap.h>
```

Public Member Functions

- [RGB](#) (float red=0.0, float green=0.0, float blue=0.0)

Public Attributes

- union {
 float **color** [3]
 struct {
 float **r**
 float **g**
 float **b**
 }
};

6.41.1 Detailed Description

Simple [RGB](#) color management class. Red Green Blue are floats in the range [0.0, 1.0].

6.41.2 Constructor & Destructor Documentation

6.41.2.1 [RGB::RGB](#) (float *red* = 0.0, float *green* = 0.0, float *blue* = 0.0) `[inline]`

Constructor

Parameters

<i>r,g,b</i>	Red, Green and Blue in the range [0.0, 1.0].
--------------	--

The documentation for this class was generated from the following file:

- colormap.h

6.42 Rotation Class Reference

Implements the rotational slip detection based on the principal axis method.

```
#include <slipdetection.h>
```

Public Member Functions

- [Rotation](#) ()
- bool [setReferenceFrame](#) (cv::Mat &referenceFrame)
- shapeFeatures [rotationFromMoments](#) (cv::Mat &frame)
- rotationResult [computeRotation](#) (cv::Mat ¤tFrame)

6.42.1 Detailed Description

Implements the rotational slip detection based on the principal axis method.

6.42.2 Constructor & Destructor Documentation

6.42.2.1 `Rotation::Rotation ()`

Constructor.

6.42.3 Member Function Documentation

6.42.3.1 `rotationResult Rotation::computeRotation (cv::Mat & currentFrame)`

Computes slip angles by evaluating the orientation using [rotationFromMoments\(\)](#) and tracking the rotation.

Parameters

<i>current-Frame</i>	The current tactile sensor matrix.
----------------------	------------------------------------

Returns

A tuple containing shape features, orientation and slip angles.

6.42.3.2 `shapeFeatures Rotation::rotationFromMoments (cv::Mat & frame)`

Computes the shape's orientation using the principal axis method. Shape features such as eccentricity and compactness can be used for quality evaluation.

Parameters

<i>frame</i>	The current tactile sensor matrix.
--------------	------------------------------------

Returns

A tuple containing shape features and orientation.

6.42.3.3 `bool Rotation::setReferenceFrame (cv::Mat & referenceFrame)`

Initialization / resetting of angle tracking

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

Returns

Success.

The documentation for this class was generated from the following files:

- slipdetection.h
- slipdetection.cpp

6.43 RowData Class Reference

Public Member Functions

- **RowData** (Glib::ustring label, bool plot, Glib::ustring value)
- **RowData** (Glib::ustring label, const std::vector< [RowData](#) > &children)
- **RowData** (const [RowData](#) &src)
- [RowData](#) & **operator=** (const [RowData](#) &src)

Public Attributes

- Glib::ustring **m_label**
- bool **m_plot**
- Glib::ustring **m_value**
- std::vector< [RowData](#) > **m_children**

The documentation for this class was generated from the following files:

- guiTreeView.h
- guiTreeView.cpp

6.44 Ext::sControllerInfo Struct Reference

A data structure describing the controller info about the remote DSACON32m controller.

```
#include <extension.h>
```

Public Attributes

- UInt16 **error_code**
- UInt32 **serial_no**
- UInt8 **hw_version**
- UInt16 **sw_version**
- UInt8 **status_flags**
- UInt8 **feature_flags**

- UInt8 **senscon_type**
- UInt8 **active_interface**
- UInt32 **can_baudrate**
- UInt16 **can_id**

6.44.1 Detailed Description

A data structure describing the controller info about the remote DSACON32m controller.

The documentation for this struct was generated from the following file:

- extension.h

6.45 sensorInfo Struct Reference

Tactile sensor [Controller](#) info (see dsa.h)

```
#include <framemanager.h>
```

Public Attributes

- uint **nb_matrices**
- uint **nb_cells**
- uint **generated_by**
- uint **hw_revision**
- uint **serial_no**
- uint **converter_resolution**

6.45.1 Detailed Description

Tactile sensor [Controller](#) info (see dsa.h)

The documentation for this struct was generated from the following file:

- framemanager.h

6.46 SlipDetector Class Reference

Combined Slip-Detection class ([Translation](#) + [Rotation](#))

```
#include <slipdetection.h>
```

Public Member Functions

- [SlipDetector](#) (uint cols, uint rows)
- void [reset](#) ()
- bool [setReferenceFrame](#) (cv::Mat &referenceFrame)
- bool [setReferenceFrameTranslation](#) (cv::Mat &referenceFrame)
- bool [setReferenceFrameTranslation](#) (cv::Mat &referenceFrame, int activeCells)
- bool [setReferenceFrameRotation](#) (cv::Mat &referenceFrame)
- bool [setReferenceFrameRotation](#) (cv::Mat &referenceFrame, int activeCells)
- [slipResult](#) [computeSlip](#) (cv::Mat ¤tFrame)

6.46.1 Detailed Description

Combined Slip-Detection class ([Translation](#) + [Rotation](#))

6.46.2 Constructor & Destructor Documentation

6.46.2.1 SlipDetector::SlipDetector (uint cols, uint rows)

Constructor. It calls the translational and rotational slip-detection constructors.

Parameters

<i>cols</i>	Tactile sensor width.
<i>rows</i>	Tactile sensor height.

6.46.3 Member Function Documentation

6.46.3.1 slipResult SlipDetector::computeSlip (cv::Mat & currentFrame)

Performs both translational and rotational slip-detection. It is not necessary to set the reference or previous tactile sensor matrix beforehand. In this case, the methods are initialized with the current frame and the actual slip vector/rotation angle is computed between the very same tactile image. The real slip-detection then starts with the next call to this function, assuming the tactile sensor matrix satisfies the constraints.

Parameters

<i>current-Frame</i>	The current tactile sensor matrix.
----------------------	------------------------------------

Returns

The combined results of the rotational and translational slip-detection.

6.46.3.2 void SlipDetector::reset ()

Invalidates the reference frame, the previous frame as well as the tracked angle.

Returns

void

6.46.3.3 bool SlipDetector::setReferenceFrame (cv::Mat & *referenceFrame*)

(Re)sets the reference frame for both translational and rotational slip-detection. Checks the number of active taxels

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

Returns

Success.

6.46.3.4 bool SlipDetector::setReferenceFrameRotation (cv::Mat & *referenceFrame*)

(Re)sets the reference frame for the rotational slip-detection. Counts the number of active taxels. [Rotation](#) fails if frame is empty and/or shape is circular.

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

Returns

Success.

6.46.3.5 bool SlipDetector::setReferenceFrameRotation (cv::Mat & *referenceFrame*, int *activeCells*)

(Re)sets the reference frame for the rotational slip-detection. Expects the number of active taxels. [Rotation](#) fails if frame is empty and/or shape is circular.

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

Returns

Success.

6.46.3.6 `bool SlipDetector::setReferenceFrameTranslation (cv::Mat & referenceFrame, int activeCells)`

(Re)sets the reference frame for the translational slip-detection. Expects the number of active taxels. [Translation](#) fails only if frames are empty.

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
<i>activeCells</i>	Number of active taxels.

Returns

Success.

6.46.3.7 `bool SlipDetector::setReferenceFrameTranslation (cv::Mat & referenceFrame)`

(Re)sets the reference frame for the translational slip-detection. Counts the number of active taxels. [Translation](#) fails only if frames are empty.

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

Returns

Success.

The documentation for this class was generated from the following files:

- slipdetection.h
- slipdetection.cpp

6.47 slipResult Struct Reference

Final return type struct: Boost tuples are limited to 10 elements, so this is why..

```
#include <slipdetection.h>
```

Public Attributes

- bool **successTranslation**

- bool **successRotation**
- double **slipVector_x**
- double **slipVector_y**
- double **slipVectorReference_x**
- double **slipVectorReference_y**
- double **slipAngle**
- double **slipAngleReference**
- double **orientation**
- double **centroid_x**
- double **centroid_y**
- double **skew_x**
- double **skew_y**
- double **lambda1**
- double **lambda2**
- double **eccentricity**
- double **compactness**

6.47.1 Detailed Description

Final return type struct: Boost tuples are limited to 10 elements, so this is why...

The documentation for this struct was generated from the following file:

- slipdetection.h

6.48 SlipVectorLive Class Reference

Visualizes current slip vector.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- void **reset** ()
- void **drawAxes** (const Cairo::RefPtr< Cairo::Context > &cr, int width, int height)
- bool **drawVector** (double x, double y)

Protected Member Functions

- virtual bool **on_expose_event** (GdkEventExpose *event)

6.48.1 Detailed Description

Visualizes current slip vector.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.49 SlipVectorTrajectory Class Reference

Visualizes slip vector trajectory.

```
#include <guiSlipDetectionMultiPlot.h>
```

Public Member Functions

- **SlipVectorTrajectory** (std::deque< slip_trajectory > &slipvectors)
- void **reset** ()
- void **drawBackgroundSurface** ()
- void **drawAxes** (const Cairo::RefPtr< Cairo::Context > &cr, int width, int height)
- bool **drawTrajectory** (std::deque< slip_trajectory > &slipvectors, uint current-FrameID)
- bool **updateTrajectory** (std::deque< slip_trajectory > &slipvectors, uint current-FrameID)

Protected Member Functions

- virtual bool **on_expose_event** (GdkEventExpose *event)

6.49.1 Detailed Description

Visualizes slip vector trajectory.

The documentation for this class was generated from the following files:

- guiSlipDetectionMultiPlot.h
- guiSlipDetectionMultiPlot.cpp

6.50 TemperatureFrame Struct Reference

Temperatures 0-6: close to axes motors, Temperature 7: FPGA, Temperature 8: Printed circuit board.

```
#include <framemanager.h>
```

Public Attributes

- `std::vector< double > values`
- `uint64_t timestamp`

6.50.1 Detailed Description

Temperatures 0-6: close to axes motors, Temperature 7: FPGA, Temperature 8: Printed circuit board.

The documentation for this struct was generated from the following file:

- `framemanager.h`

6.51 TemperatureNoise Struct Reference

A data structure containing the linear regression parameters as well as the RMS Error of the prediction band.

```
#include <calibration.h>
```

Public Attributes

- `double slope`
- `double intercept`
- `double RMSE`

6.51.1 Detailed Description

A data structure containing the linear regression parameters as well as the RMS Error of the prediction band.

The documentation for this struct was generated from the following file:

- `calibration.h`

6.52 TimeSeriesDataset Struct Reference

Public Member Functions

- **TimeSeriesDataset** (`std::string name`, [RGB](#) `rgb`, `uint size`)

Public Attributes

- std::string **description**
- **RGB color**
- std::vector< float > **rawData**
- std::vector< float > **sampleIntervalMin**
- std::vector< float > **sampleIntervalMax**
- std::vector< float > **filteredSamples**
- bool **calculateOverview**
- bool **calculateFiltering**

The documentation for this struct was generated from the following file:

- guiGraph.h

6.53 TimestampComparator< Frame > Struct Template Reference

Timestamp comparator functor.

```
#include <framemanager.h>
```

Public Member Functions

- bool **operator()** (const Frame &frame1, const Frame &frame2) const
- bool **operator()** (const Frame &frame, uint64_t timestamp) const
- bool **operator()** (uint64_t timestamp, const Frame &frame) const

6.53.1 Detailed Description

```
template<typename Frame>struct TimestampComparator< Frame >
```

Timestamp comparator functor.

The documentation for this struct was generated from the following file:

- framemanager.h

6.54 Translation Class Reference

Implements the translational slip detection based on tracking the convolution matrix's center of gravity.

```
#include <slipdetection.h>
```

Public Member Functions

- [Translation](#) (uint cols, uint rows)
- void [init](#) (uint cols, uint rows)
- void [setReferenceFrame](#) (cv::Mat &referenceFrame)
- cv::Point2d [computeSlipReference](#) (cv::Mat ¤tFrame)
- cv::Point2d [computeSlip](#) (cv::Mat ¤tFrame)

6.54.1 Detailed Description

Implements the translational slip detection based on tracking the convolution matrix's center of gravity.

6.54.2 Constructor & Destructor Documentation

6.54.2.1 Translation::Translation (uint *cols*, uint *rows*)

Constructor. Calls [init\(\)](#)

Parameters

<i>cols</i>	Tactile sensor width.
<i>rows</i>	Tactile sensor height.

6.54.3 Member Function Documentation

6.54.3.1 cv::Point2d Translation::computeSlip (cv::Mat & *currentFrame*)

Computes the slip vector between the current and the previous tactile sensor matrix. (Normalized Cross Correlation) Use this method in conjunction with [setReferenceFrame\(\)](#).

Parameters

<i>current-Frame</i>	The current tactile sensor matrix.
----------------------	------------------------------------

Returns

The corresponding slip vector.

6.54.3.2 cv::Point2d Translation::computeSlipReference (cv::Mat & *currentFrame*)

Computes the slip vector between the current and the reference tactile sensor matrix. (Normalized Cross Correlation) Use this method in conjunction with [setReferenceFrame\(\)](#).

Parameters

<i>current-Frame</i>	The current tactile sensor matrix.
----------------------	------------------------------------

Returns

The corresponding slip vector.

6.54.3.3 void Translation::init (uint *cols*, uint *rows*)

Creates index matrices of corresponding taxel positions in convolution matrix.

Parameters

<i>cols</i>	Tactile sensor width.
<i>rows</i>	Tactile sensor height.

6.54.3.4 void Translation::setReferenceFrame (cv::Mat & *referenceFrame*)

(Re)sets the reference tactile sensor matrix. Computes the reference frames's convolution with itself.

Parameters

<i>reference-Frame</i>	The reference tactile sensor matrix.
------------------------	--------------------------------------

The documentation for this class was generated from the following files:

- slipdetection.h
- slipdetection.cpp

6.55 TSFrame Struct Reference

Tactile sensor frame.

```
#include <framemanager.h>
```

Public Member Functions

- **TSFrame** (uint nb_cells)

Public Attributes

- std::vector< float > **cells**
- uint64_t **timestamp**

6.55.1 Detailed Description

Tactile sensor frame.

The documentation for this struct was generated from the following file:

- framemanager.h

Chapter 7

File Documentation

7.1 sdhoptions.cpp File Reference

Implementation of a class to parse common SDH related command line options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <fstream>
#include "sdh/sdh.h"
#include "sdh/sdhlibrary_settings.h"
#include "sdh/release.h"
#include "sdh/dsa.h"
#include "sdhoptions.h"
```

Defines

- #define [XSTRINGIFY](#)(_x) STRINGIFY(_x)
- #define [STRINGIFY](#)(_s) #_s
helper macro for XSTRINGIFY, see there

7.1.1 Detailed Description

Implementation of a class to parse common SDH related command line options.

Author

Dirk Osswald

Date

2008-05-05

7.1.2 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

7.1.3 Define Documentation**7.1.3.1 #define XSTRINGIFY(_x) STRINGIFY(_x)**macro for stringification of `_x`allows to stringify the **value** of a macro:

```
#define foo 4

STRINGIFY( foo ) // yields "foo"
XSTRINGIFY( foo ) // yields "4"
```

7.2 sdhoptions.h File Reference

Implementation of a class to parse common SDH related command line options.

```
#include <getopt.h>
#include <assert.h>
#include <iostream>
#include <string>
#include <sdh/sdh.h>
```

Classes

- class [cSDHOptions](#)

*class for command line option parsing holding option parsing results***Defines**

- #define [SDHUSAGE_DEFAULT](#) "general sdhcom_serial sdhcom_common sdhcom_esdcan sdhcom_peakcan sdhcom_canccommon sdhcom_tcp"
- #define **SDH_DEFAULT_TCP_ADR** "192.168.1.1"
- #define **SDH_DEFAULT_TCP_PORT** 23

7.2.1 Detailed Description

Implementation of a class to parse common SDH related command line options. Taken from SDHLibrary (modified version).

7.2.2 General file information

Author

Dirk Osswald

Date

2008-05-05

7.2.3 Copyright

Copyright (c) 2008 SCHUNK GmbH & Co. KG

7.2.4 Define Documentation

7.2.4.1 `#define SDHUSAGE_DEFAULT "general sdhcom_serial sdhcom_common sdhcom_esdcan
sdhcom_peakcan sdhcom_canccommon sdhcom_tcp"`

string defining all the usage helptexts included by default

Bug

When compiled with VCC then the macros WITH_ESD_CAN / WITH_PEAK_CAN used above are not available since these are defined in the VCC project settings of the SDHLibrary VCC-Project. Therefore the value of SDHUSAGE_DEFAULT is incorrect and thus the [cSDHOptions](#) will display an incomplete usage string when called with -h/--help.

Workaround: use the online help contained in the doxygen documentation: [Online help of demonstration programs](#)

7.3 utils.h File Reference

```
#include <cmath>
#include <algorithm>
#include <limits>
#include <string>
#include <stdint.h>
```

Functions

- uint64_t [utils::getCurrentTimeMilliseconds](#) ()
- bool [utils::almostEqual](#) (float x, float y, int ulp)
- template<typename T >
std::string [utils::numberToString](#) (T number)
- template<typename T >
T [utils::stringToNumber](#) (const std::string &text)
- void [utils::splitFilename](#) (const std::string &filename, std::string &basename, std::string &extension)
- double [utils::degToRad](#) (double d)

7.3.1 Detailed Description

Just a few utilities functions. Possibly copy-pasted from somewhere else...

Index

- ~FrameGrabberDSA
 - FrameGrabberDSA, [43](#)
- ~FrameGrabberSDH
 - FrameGrabberSDH, [47](#)
- addComport
 - Ext, [33](#)
- addTSFrame
 - FrameManager, [53](#)
- allocateTSFrame
 - FrameManager, [53](#)
- applyFilter
 - FrameProcessor, [78](#)
- calcCharacteristics
 - FrameProcessor, [78](#)
- Calibration, [13](#)
 - Calibration, [13](#)
 - getTemperatureNoise, [13](#)
 - readTemperatureNoise, [14](#)
- Camera, [14](#)
 - Camera, [15](#)
 - getPosition, [15](#)
 - getView, [16](#)
 - move, [16](#)
 - moveX, [16](#)
 - moveY, [16](#)
 - moveZ, [17](#)
 - rotateX, [17](#)
 - rotateY, [17](#)
 - setPosition, [17](#)
 - setup, [18](#)
 - setView, [18](#)
- Chebyshev, [18](#)
 - Chebyshev, [19](#)
 - computeInvariants, [19](#)
 - computeReconstruction, [19](#)
 - initialize, [20](#)
- Close
 - Ext, [33](#)
- Colormap, [20](#)
 - createColorTable, [21](#)
 - getColorFromColormap, [21](#)
 - getColorFromTable, [21](#)
 - hsl2rgb, [22](#)
 - interpolate, [22](#)
 - limitColorRange, [22](#)
 - rgb2hsl, [23](#)
- computeCentroid
 - FeatureExtraction, [36](#)
- computeInvariants
 - Chebyshev, [19](#)
- computeMiniball
 - FeatureExtraction, [36](#)
- computeMiniballCentroid
 - FeatureExtraction, [36](#), [37](#)
- computeMiniballPoints
 - FeatureExtraction, [37](#)
- computeMoments
 - FeatureExtraction, [38](#)
- computeReconstruction
 - Chebyshev, [19](#)
- computeRotation
 - Rotation, [107](#)
- computeSlip
 - FrameManager, [53](#)
 - SlipDetector, [110](#)
 - Translation, [117](#)
- computeSlipLive
 - FrameManager, [53](#)
- computeSlipReference
 - Translation, [117](#)
- computeStandardDeviation
 - FeatureExtraction, [38](#)
- computeTransformationMatrixPointOnSensorPlaneXYZ
 - ForwardKinematics, [40](#)
- computeTransformationMatrixTaxelXYZ
 - ForwardKinematics, [40](#)
- connectDSA
 - Controller, [24](#)
- connectSDH
 - Controller, [24](#)

- Controller, 23
 - connectDSA, 24
 - connectSDH, 24
 - Controller, 24
 - disconnectDSA, 25
 - disconnectSDH, 25
 - getPreshape, 25
 - grasp, 25
 - graspReactive, 26
 - isAvailableDSA, 26
 - isAvailableSDH, 26
 - isConnectedDSA, 26
 - isConnectedSDH, 27
 - loadProfile, 27
- convertCellIndex
 - FrameManager, 54
- createColorTable
 - Colormap, 21
- createJointAngleMapping
 - FrameManager, 54
- createSelectedCellsIdx
 - FrameManager, 54
- createTemperatureMapping
 - FrameManager, 55
- cropToFrames
 - FrameManager, 55
- cSDHOptions, 28
 - cSDHOptions, 29
 - OpenCommunication, 30
 - Parse, 30
- decodeTSFrame
 - FrameManager, 55
- deleteTSFrame
 - FrameManager, 56
- Device, 31
- disableSlipDetection
 - FrameManager, 56
- disconnectDSA
 - Controller, 25
- disconnectSDH
 - Controller, 25
- embedPython
 - guiMain, 91
- enableRecording
 - FrameGrabberDSA, 43
 - FrameGrabberSDH, 47
- enableSlipDetection
 - FrameManager, 56
- encodeTSFrame
 - FrameManager, 56
- execute
 - FrameGrabberDSA, 43
 - FrameGrabberSDH, 47
- Ext, 31
 - addComport, 33
 - Close, 33
 - getComportDriver, 33
 - IdentifyDevices, 33
 - listComports, 33
 - Open, 34
 - probeSerial8250Comports, 34
- Ext::sControllerInfo, 108
- ExtException, 34
- FeatureExtraction, 35
 - computeCentroid, 36
 - computeMiniball, 36
 - computeMiniballCentroid, 36, 37
 - computeMiniballPoints, 37
 - computeMoments, 38
 - computeStandardDeviation, 38
 - FeatureExtraction, 35
- FeatureExtractionWrapper, 39
- finish
 - FrameGrabberDSA, 43
 - FrameGrabberSDH, 47
- float2hex
 - FrameManager, 57
- ForwardKinematics, 39
 - computeTransformationMatrixPointOn-SensorPlaneXYZ, 40
 - computeTransformationMatrixTaxelXYZ, 40
 - ForwardKinematics, 40
 - GetPointOnSensorPlaneXYZ, 40
 - GetTaxelXYZ, 41
 - setAngles, 41
- FrameGrabberDSA, 42
 - ~FrameGrabberDSA, 43
 - enableRecording, 43
 - execute, 43
 - finish, 43
 - FrameGrabberDSA, 42
 - getFrameNumber, 43
 - isCapturing, 43
 - isRecording, 44
 - isRunning, 44
 - pause, 44

- resume, [44](#)
- setFrameManager, [44](#)
- setFramerate, [45](#)
- start, [45](#)
- suspendRecording, [45](#)
- FrameGrabberSDH, [46](#)
 - ~FrameGrabberSDH, [47](#)
 - enableRecording, [47](#)
 - execute, [47](#)
 - finish, [47](#)
 - FrameGrabberSDH, [46](#)
 - getFrameNumber, [47](#)
 - isCapturing, [47](#)
 - isRecording, [48](#)
 - isRunning, [48](#)
 - pause, [48](#)
 - pauseBlocking, [48](#)
 - resume, [48](#)
 - setFrameManager, [48](#)
 - setFramerateJointAngles, [49](#)
 - setFramerateTemperature, [49](#)
 - setJointAngle, [49](#)
 - setTemperature, [49](#)
 - start, [49](#)
 - suspendRecording, [50](#)
- FrameManager, [50](#)
 - addTSFrame, [53](#)
 - allocateTSFrame, [53](#)
 - computeSlip, [53](#)
 - computeSlipLive, [53](#)
 - convertCellIndex, [54](#)
 - createJointAngleMapping, [54](#)
 - createSelectedCellsIdx, [54](#)
 - createTemperatureMapping, [55](#)
 - cropToFrames, [55](#)
 - decodeTSFrame, [55](#)
 - deleteTSFrame, [56](#)
 - disableSlipDetection, [56](#)
 - enableSlipDetection, [56](#)
 - encodeTSFrame, [56](#)
 - float2hex, [57](#)
 - FrameManager, [53](#)
 - getCorrespondingJointAngle, [57](#)
 - getCorrespondingTemperature, [57](#)
 - getCurrentFilteredFrame, [57](#)
 - getCurrentFrame, [58](#)
 - getCurrentFrameID, [58](#)
 - getCurrentJointAngleFrame, [58](#)
 - getDynamicMask, [58](#)
 - getFilteredFrame, [58](#)
 - getFilteredTexel, [59](#)
 - getFrame, [59](#)
 - getFrameCountJointAngles, [59](#)
 - getFrameCountTemperature, [60](#)
 - getFrameCountTS, [60](#)
 - getFrameProcessor, [60](#)
 - getJointAngleFrame, [60](#)
 - getJointAngleFrameAvailable, [60](#)
 - getMatrixInfo, [61](#)
 - getNumCells, [61](#)
 - getNumMatrices, [61](#)
 - getNumSelectedCells, [61](#)
 - getProfileName, [61](#)
 - getSelection, [62](#)
 - getSensorInfo, [62](#)
 - getSlipDetectionState, [62](#)
 - getSlipDetector, [62](#)
 - getSlipLive, [63](#)
 - getSlipLiveBinary, [63](#)
 - getStaticMask, [63](#)
 - getTemperatureFrame, [63](#)
 - getTexel, [64](#)
 - getTSFrameAvailable, [64](#)
 - hex2float, [64](#)
 - isConnectedDSA, [65](#)
 - isConnectedSDH, [65](#)
 - isSelected, [65](#)
 - loadFrames, [65](#)
 - loadProfile, [65](#)
 - print_tree, [66](#)
 - printJointAngleFrame, [66](#)
 - printTemperatureFrame, [67](#)
 - printTSMatrices, [67](#)
 - printTSMatrix, [67](#)
 - queryDSAInfo, [67](#)
 - requestJointAngleFrame, [68](#)
 - requestTemperatureFrame, [68](#)
 - resetOffline, [68](#)
 - resetOnline, [68](#)
 - selectCell, [68](#)
 - setCurrentFrameID, [69](#)
 - setDSA, [69](#)
 - setDynamicMask, [69](#)
 - setFilterBilateral, [69](#)
 - setFilterGaussian, [70](#)
 - setFilterMedian, [70](#)
 - setFilterMedian3D, [70](#)
 - setFilterMorphological, [71](#)
 - setFilterNone, [71](#)
 - setFrameGrabberDSA, [71](#)

- setFrameGrabberSDH, 71
- setJointAngleFrameAvailable, 72
- setLiveFrame, 72
- setSDH, 72
- setSensitivity, 72
- setSlipReferenceFrame, 73
- setSlipReferenceFrameLive, 73
- setSlipThresholdConsecutive, 73
- setSlipThresholdReference, 73
- setThreshold, 74
- setTSFrameAvailable, 74
- slipResultConsumer, 74
- slipResultProducer, 74
- storeFrames, 75
- storeProfile, 75
- storeProfileSelection, 75
- FrameManagerWrapper, 76
- FrameProcessor, 77
 - applyFilter, 78
 - calcCharacteristics, 78
 - FrameProcessor, 78
 - getAverage, 79
 - getFilterType, 79
 - getMatrixAverage, 79
 - getMatrixMax, 79
 - getMatrixMin, 80
 - getMatrixNumActiveCells, 80
 - getMax, 80
 - getMin, 80
 - getNumActiveCells, 81
 - setFilterBilateral, 81
 - setFilterGaussian, 81
 - setFilterMedian, 81
 - setFilterMedian3D, 82
 - setFilterNone, 82
 - setFilterOpening, 82
 - setFrameManager, 82
- getAverage
 - FrameProcessor, 79
- getColorFromColormap
 - Colormap, 21
- getColorFromTable
 - Colormap, 21
- getComportDriver
 - Ext, 33
- getCorrespondingJointAngle
 - FrameManager, 57
- getCorrespondingTemperature
 - FrameManager, 57
- getCurrentFilteredFrame
 - FrameManager, 57
- getCurrentFrame
 - FrameManager, 58
- getCurrentFrameID
 - FrameManager, 58
- getCurrentJointAngleFrame
 - FrameManager, 58
- getDynamicMask
 - FrameManager, 58
- getFilteredFrame
 - FrameManager, 58
- getFilteredTexel
 - FrameManager, 59
- getFilterType
 - FrameProcessor, 79
- getFrame
 - FrameManager, 59
- getFrameCountJointAngles
 - FrameManager, 59
- getFrameCountTemperature
 - FrameManager, 60
- getFrameCountTS
 - FrameManager, 60
- getFrameNumber
 - FrameGrabberDSA, 43
 - FrameGrabberSDH, 47
- getFrameProcessor
 - FrameManager, 60
- getJointAngleFrame
 - FrameManager, 60
- getJointAngleFrameAvailable
 - FrameManager, 60
- getMatrixAverage
 - FrameProcessor, 79
- getMatrixInfo
 - FrameManager, 61
- getMatrixMax
 - FrameProcessor, 79
- getMatrixMin
 - FrameProcessor, 80
- getMatrixNumActiveCells
 - FrameProcessor, 80
- getMax
 - FrameProcessor, 80
- getMin
 - FrameProcessor, 80
- getNumActiveCells
 - FrameProcessor, 81
- getNumCells

- FrameManager, 61
- getNumMatrices
 - FrameManager, 61
- getNumSelectedCells
 - FrameManager, 61
- GetPointOnSensorPlaneXYZ
 - ForwardKinematics, 40
- getPosition
 - Camera, 15
- getPreshape
 - Controller, 25
- getProfileName
 - FrameManager, 61
- getSelection
 - FrameManager, 62
- getSensorInfo
 - FrameManager, 62
- getSlipDetectionState
 - FrameManager, 62
- getSlipDetector
 - FrameManager, 62
- getSlipLive
 - FrameManager, 63
- getSlipLiveBinary
 - FrameManager, 63
- getStaticMask
 - FrameManager, 63
- GetTaxelXYZ
 - ForwardKinematics, 41
- getTemperatureFrame
 - FrameManager, 63
- getTemperatureNoise
 - Calibration, 13
- getTexel
 - FrameManager, 64
- getTSFrameAvailable
 - FrameManager, 64
- getView
 - Camera, 16
- grasp
 - Controller, 25
- graspReactive
 - Controller, 26
- guiChart, 83
 - initDataset, 84
 - updateDataset, 84
- guiController, 84
 - on_button_threshold_clicked, 87
- guiGraph, 87
 - on_button_press_event, 88
- on_button_release_event, 88
- on_motion_notify_event, 88
- setMarkerPosition, 88
- guiMain, 89
 - embedPython, 91
 - on_menu_load_profile, 91
 - on_menu_new_profile, 91
 - resetGUIOffline, 91
 - updateGUIOffline, 91
- guiRenderer, 91
- guiRenderer2D, 93
- guiRenderer3D, 94
 - on_button_press_event, 95
 - on_button_release_event, 95
 - on_configure_event, 95
 - on_expose_event, 95
 - on_motion_notify_event, 95
 - on_realize, 95
- guiScreenshot, 95
- guiSeekBar, 96
 - on_signal_timeout, 97
 - setSliderPosition, 97
- guiSlipDetection, 97
- guiSlipDetectionMultiPlot, 98
- guiTools, 99
- guiTreeView, 100
- hex2float
 - FrameManager, 64
- HSL, 100
 - HSL, 101
- hsl2rgb
 - Colormap, 22
- IdentifyDevices
 - Ext, 33
- init
 - Translation, 118
- initDataset
 - guiChart, 84
- initialize
 - Chebyshev, 20
- interpolate
 - Colormap, 22
- isAvailableDSA
 - Controller, 26
- isAvailableSDH
 - Controller, 26
- isCapturing
 - FrameGrabberDSA, 43

- FrameGrabberSDH, 47
- isConnectedDSA
 - Controller, 26
 - FrameManager, 65
- isConnectedSDH
 - Controller, 27
 - FrameManager, 65
- isRecording
 - FrameGrabberDSA, 44
 - FrameGrabberSDH, 48
- isRunning
 - FrameGrabberDSA, 44
 - FrameGrabberSDH, 48
- isSelected
 - FrameManager, 65
- JointAngleFrame, 101
- limitColorRange
 - Colormap, 22
- listComports
 - Ext, 33
- loadFrames
 - FrameManager, 65
- loadProfile
 - Controller, 27
 - FrameManager, 65
- matrixInfo, 102
- Miniball::CoordAccessor, 27
- Miniball::CoordAccessor< Pit_, Cit_ * >, 28
- Miniball::Miniball, 103
- move
 - Camera, 16
- moveX
 - Camera, 16
- moveY
 - Camera, 16
- moveZ
 - Camera, 17
- NiceScale, 103
- NumPyArrayData, 104
- on_button_press_event
 - guiGraph, 88
 - guiRenderer3D, 95
- on_button_release_event
 - guiGraph, 88
 - guiRenderer3D, 95
- on_button_threshold_clicked
 - guiController, 87
- on_configure_event
 - guiRenderer3D, 95
- on_expose_event
 - guiRenderer3D, 95
- on_menu_load_profile
 - guiMain, 91
- on_menu_new_profile
 - guiMain, 91
- on_motion_notify_event
 - guiGraph, 88
 - guiRenderer3D, 95
- on_realize
 - guiRenderer3D, 95
- on_signal_timeout
 - guiSeekbar, 97
- Open
 - Ext, 34
- OpenCommunication
 - cSDHOptions, 30
- Orientation, 104
- OrientationTrajectory, 105
- Parse
 - cSDHOptions, 30
- pause
 - FrameGrabberDSA, 44
 - FrameGrabberSDH, 48
- pauseBlocking
 - FrameGrabberSDH, 48
- print_tree
 - FrameManager, 66
- printJointAngleFrame
 - FrameManager, 66
- printTemperatureFrame
 - FrameManager, 67
- printTSMatrices
 - FrameManager, 67
- printTSMatrix
 - FrameManager, 67
- probeSerial8250Comports
 - Ext, 34
- queryDSAInfo
 - FrameManager, 67
- readTemperatureNoise
 - Calibration, 14
- requestJointAngleFrame

- FrameManager, 68
- requestTemperatureFrame
 - FrameManager, 68
- reset
 - SlipDetector, 110
- resetGUIOffline
 - guiMain, 91
- resetOffline
 - FrameManager, 68
- resetOnline
 - FrameManager, 68
- resume
 - FrameGrabberDSA, 44
 - FrameGrabberSDH, 48
- RGB, 105
 - RGB, 106
- rgb2hsl
 - Colormap, 23
- rotateX
 - Camera, 17
- rotateY
 - Camera, 17
- Rotation, 106
 - computeRotation, 107
 - Rotation, 107
 - rotationFromMoments, 107
 - setReferenceFrame, 107
- rotationFromMoments
 - Rotation, 107
- RowData, 108
- sdhoptions.cpp, 121
 - XSTRINGIFY, 122
- sdhoptions.h, 122
 - SDHUSAGE_DEFAULT, 123
- SDHUSAGE_DEFAULT
 - sdhoptions.h, 123
- selectCell
 - FrameManager, 68
- sensorInfo, 109
- setAngles
 - ForwardKinematics, 41
- setCurrentFrameID
 - FrameManager, 69
- setDSA
 - FrameManager, 69
- setDynamicMask
 - FrameManager, 69
- setFilterBilateral
 - FrameManager, 69
- FrameProcessor, 81
- setFilterGaussian
 - FrameManager, 70
 - FrameProcessor, 81
- setFilterMedian
 - FrameManager, 70
 - FrameProcessor, 81
- setFilterMedian3D
 - FrameManager, 70
 - FrameProcessor, 82
- setFilterMorphological
 - FrameManager, 71
- setFilterNone
 - FrameManager, 71
 - FrameProcessor, 82
- setFilterOpening
 - FrameProcessor, 82
- setFrameGrabberDSA
 - FrameManager, 71
- setFrameGrabberSDH
 - FrameManager, 71
- setFrameManager
 - FrameGrabberDSA, 44
 - FrameGrabberSDH, 48
 - FrameProcessor, 82
- setFramerate
 - FrameGrabberDSA, 45
- setFramerateJointAngles
 - FrameGrabberSDH, 49
- setFramerateTemperature
 - FrameGrabberSDH, 49
- setJointAngle
 - FrameGrabberSDH, 49
- setJointAngleFrameAvailable
 - FrameManager, 72
- setLiveFrame
 - FrameManager, 72
- setMarkerPosition
 - guiGraph, 88
- setPosition
 - Camera, 17
- setReferenceFrame
 - Rotation, 107
 - SlipDetector, 111
 - Translation, 118
- setReferenceFrameRotation
 - SlipDetector, 111
- setReferenceFrameTranslation
 - SlipDetector, 112
- setSDH

- FrameManager, [72](#)
- setSensitivity
 - FrameManager, [72](#)
- setSliderPosition
 - guiSeekBar, [97](#)
- setSlipReferenceFrame
 - FrameManager, [73](#)
- setSlipReferenceFrameLive
 - FrameManager, [73](#)
- setSlipThresholdConsecutive
 - FrameManager, [73](#)
- setSlipThresholdReference
 - FrameManager, [73](#)
- setTemperature
 - FrameGrabberSDH, [49](#)
- setThreshold
 - FrameManager, [74](#)
- setTSFrameAvailable
 - FrameManager, [74](#)
- setup
 - Camera, [18](#)
- setView
 - Camera, [18](#)
- SlipDetector, [109](#)
 - computeSlip, [110](#)
 - reset, [110](#)
 - setReferenceFrame, [111](#)
 - setReferenceFrameRotation, [111](#)
 - setReferenceFrameTranslation, [112](#)
 - SlipDetector, [110](#)
- slipResult, [112](#)
- slipResultConsumer
 - FrameManager, [74](#)
- slipResultProducer
 - FrameManager, [74](#)
- SlipVectorLive, [113](#)
- SlipVectorTrajectory, [114](#)
- start
 - FrameGrabberDSA, [45](#)
 - FrameGrabberSDH, [49](#)
- storeFrames
 - FrameManager, [75](#)
- storeProfile
 - FrameManager, [75](#)
- storeProfileSelection
 - FrameManager, [75](#)
- suspendRecording
 - FrameGrabberDSA, [45](#)
 - FrameGrabberSDH, [50](#)
- TemperatureFrame, [114](#)
- TemperatureNoise, [115](#)
- TimeSeriesDataset, [115](#)
- TimestampComparator, [116](#)
- Translation, [116](#)
 - computeSlip, [117](#)
 - computeSlipReference, [117](#)
 - init, [118](#)
 - setReferenceFrame, [118](#)
 - Translation, [117](#)
- TSFrame, [118](#)
- updateDataset
 - guiChart, [84](#)
- updateGUIOffline
 - guiMain, [91](#)
- utils.h, [123](#)
- XSTRINGIFY
 - sdhoptions.cpp, [122](#)