Rigid Track

Generated by Doxygen 1.8.13

Contents

1	Rigio	d Track	Doxygen	Docum	nentat	ion										1
	1.1	Introdu	iction						 	 	 	 		 		 1
	1.2	Rigid 1	rack Insta	llation					 	 	 	 		 		 1
	1.3	Source	Code .						 	 	 	 		 		 1
2	Hiera	archica	l Index													3
	2.1	Class	Hierarchy						 	 	 	 		 		 3
3	Clas	s Index														5
	3.1	Class	List						 	 	 	 		 		 5
4	File	Index														7
	4.1	File Lis	st						 	 	 	 		 		 7
5	Clas	s Docu	mentatior	1												9
	5.1	commo	Object Cla	ss Refe	rence				 	 	 	 		 		 9
		5.1.1	Detailed	Descrip	otion				 	 	 	 		 		 9
		5.1.2	Member	Functio	n Doc	umer	ntatio	n .	 	 	 	 		 		 9
			5.1.2.1	addLo	og() .				 	 	 	 		 		 10
			5.1.2.2	chang	jelmag	је() .			 	 	 	 		 		 10
			5.1.2.3	chang	jeStati	us() .			 	 	 	 		 		 11
			5.1.2.4	clearL	-og()				 	 	 	 		 		 11
			5.1.2.5	enable	eP3P()			 	 	 	 		 		 12
			5.1.2.6	image	Chan	ged .			 	 	 	 		 		 12
			5.1.2.7	logAd	ded .				 	 	 	 		 		 12

ii CONTENTS

		5.1.2.8	logCleared	. 13
		5.1.2.9	P3Penabled	. 13
		5.1.2.10	progressUpdate()	. 14
		5.1.2.11	progressUpdated	. 14
		5.1.2.12	statusChanged	. 15
5.2	CvMod	delEstimato	or2 Class Reference	. 15
	5.2.1	Detailed	Description	. 16
	5.2.2	Construc	ctor & Destructor Documentation	. 16
		5.2.2.1	CvModelEstimator2()	. 16
		5.2.2.2	~CvModelEstimator2()	. 16
	5.2.3	Member	Function Documentation	. 16
		5.2.3.1	checkSubset()	. 16
		5.2.3.2	computeReprojError()	. 17
		5.2.3.3	findInliers()	. 17
		5.2.3.4	getSubset()	. 17
		5.2.3.5	refine()	. 17
		5.2.3.6	runKernel()	. 18
		5.2.3.7	runLMeDS()	. 18
		5.2.3.8	runRANSAC()	. 18
		5.2.3.9	setSeed()	. 18
	5.2.4	Member	Data Documentation	. 18
		5.2.4.1	checkPartialSubsets	. 18
		5.2.4.2	maxBasicSolutions	. 19
		5.2.4.3	modelPoints	. 19
		5.2.4.4	modelSize	. 19
		5.2.4.5	rng	. 19
5.3	RigidTi	rack Class	Reference	. 19
	5.3.1	Detailed	Description	. 20
	5.3.2	Construc	ctor & Destructor Documentation	. 20
		5.3.2.1	RigidTrack()	. 20

CONTENTS

5.3.3	Member	Function Documentation	21
	5.3.3.1	clearLog	21
	5.3.3.2	enableP3P	21
	5.3.3.3	on_actionAbout_Rigid_Track_triggered	21
	5.3.3.4	on_actionOpen_Installation_Folder_triggered	22
	5.3.3.5	on_actionOpen_Log_Folder_triggered	22
	5.3.3.6	on_actionShow_Help_triggered	22
	5.3.3.7	on_btnCalibrate_clicked	23
	5.3.3.8	on_btnCalibrateGround_clicked	23
	5.3.3.9	on_btnLoadCalib_clicked	24
	5.3.3.10	on_btnStartCamera_clicked	25
	5.3.3.11	on_btnZero_clicked	25
	5.3.3.12	on_cbInvert_stateChanged	26
	5.3.3.13	on_cbSafety2_stateChanged	26
	5.3.3.14	on_cbSafety_stateChanged	27
	5.3.3.15	on_dsbDimension_valueChanged	28
	5.3.3.16	on_leIPObject_returnPressed	28
	5.3.3.17	on_leIPSafety2_returnPressed	29
	5.3.3.18	on_leIPSafety_returnPressed	30
	5.3.3.19	on_pbLoadMarker_clicked	31
	5.3.3.20	on_rbEPnP_clicked	31
	5.3.3.21	on_rblterative_clicked	32
	5.3.3.22	on_rbP3P_clicked	32
	5.3.3.23	on_sbAngle_valueChanged	33
	5.3.3.24	on_sbHeadingOffset_valueChanged	33
	5.3.3.25	progressUpdate	33
	5.3.3.26	setImage	34
	5.3.3.27	setLog	34
Surface	e Class Re	eference	34
5.4.1	Detailed	Description	35

5.4

iv CONTENTS

		5.4.2	Construc	ctor & Destructor Documentation	35
			5.4.2.1	Surface()	35
			5.4.2.2	~Surface()	36
		5.4.3	Member	Function Documentation	36
			5.4.3.1	CalculateSize()	36
			5.4.3.2	GetBuffer()	37
			5.4.3.3	GetTexture()	37
			5.4.3.4	Height()	38
			5.4.3.5	PixelSpan()	38
			5.4.3.6	PutPixel()	39
			5.4.3.7	RebindTexture()	39
			5.4.3.8	Resize()	40
			5.4.3.9	SurfaceHeight()	41
			5.4.3.10	SurfaceWidth()	42
			5.4.3.11	Width()	42
6	File	Docum	entation		43
6	File 6.1			elest.h File Reference	
6		RigidTr	ack/_mode	elest.h File Reference	. 43
6	6.1	RigidTr RigidTr	rack/_mode	nunication.cpp File Reference	43
6	6.1	RigidTr RigidTr RigidTr	rack/_moderack/comm	nunication.cpp File Reference	43 44
6	6.16.26.3	RigidTr RigidTr RigidTr RigidTr	rack/_mode rack/comm rack/comm rack/Doxyg	nunication.cpp File Reference	43 43 44 45
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr	rack/_moderack/comm rack/comm rack/Doxyg rack/main.e	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference	43 43 44 45 45
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr	rack/_mode rack/comm rack/comm rack/Doxyg rack/main.e	nunication.cpp File Reference	43 43 44 45 45 49
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_mode rack/comm rack/comm rack/Doxyg rack/main.e	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description	43 43 44 45 45 49 50
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_mode rack/comm rack/comm rack/Doxyg rack/main.c Detailed Function 6.5.2.1	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions()	43 43 44 45 45 49 50
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_moderack/commerack/commerack/Doxygerack/main.c	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions() calibrateCamera()	43 43 44 45 45 49 50 50
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_moderack/commerack/commerack/Doxygorack/main.commerack/commerack/	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions() calibrateCamera() calibrateGround()	43 44 45 45 45 50 50 51
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_moderack/commerack/commerack/Doxygorack/main.commerack/commerack/main.commerack/comme	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions() calibrateCamera() calibrateGround() closeUDP()	43 44 45 45 49 50 50 51 53
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_moderack/commerack/commerack/commerack/Doxygerack/main.cc Detailed Function 6.5.2.1 6.5.2.2 6.5.2.3 6.5.2.4 6.5.2.5	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions() calibrateCamera() calibrateGround() closeUDP() determineExposure()	43 44 45 45 49 50 51 53 57
6	6.1 6.2 6.3 6.4	RigidTr RigidTr RigidTr RigidTr 6.5.1	rack/_moderack/commerack/commerack/Doxygorack/main.commerack/commerack/main.commerack/comme	nunication.cpp File Reference nunication.h File Reference genMain.md File Reference cpp File Reference Description Documentation calcBoardCornerPositions() calibrateCamera() calibrateGround() closeUDP()	43 44 45 45 45 50 50 51 57 57

CONTENTS

	6.5.2.8	getEulerAngles()	62
	6.5.2.9	loadCalibration()	63
	6.5.2.10	loadCameraPosition()	64
	6.5.2.11	loadMarkerConfig()	65
	6.5.2.12	main()	67
	6.5.2.13	Mat2QPixmap()	69
	6.5.2.14	projectCoordinateFrame()	70
	6.5.2.15	sendDataUDP()	71
	6.5.2.16	setHeadingOffset()	71
	6.5.2.17	setReference()	72
	6.5.2.18	setUpUDP()	75
	6.5.2.19	startStopCamera()	76
	6.5.2.20	startTracking()	78
	6.5.2.21	testAlgorithms()	83
6.5.3	Variable I	Documentation	86
	6.5.3.1	BACKBUFFER BITSPERPIXEL	96
	0.0.0.1	DAORDOIT EIT_BITOI EIT IACE	86
	6.5.3.2	camera_started	86
	6.5.3.2	camera_started	86
	6.5.3.2 6.5.3.3	camera_started	86 86
	6.5.3.2 6.5.3.3 6.5.3.4	camera_started	86 86
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5	camera_started	86 86 86 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6	camera_started	86 86 86 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7	camera_started	86 86 86 87 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7 6.5.3.8	camera_started	86 86 87 87 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7 6.5.3.8 6.5.3.9	camera_started cameraMatrix commObj coordinateFrame coordinateFrameProjected currentMinIndex currentPointDistance data	86 86 86 87 87 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7 6.5.3.8 6.5.3.9 6.5.3.10	camera_started cameraMatrix commObj . coordinateFrame . coordinateFrameProjected . currentMinIndex . currentPointDistance . data datagram	86 86 86 87 87 87 87
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7 6.5.3.8 6.5.3.9 6.5.3.10 6.5.3.11	camera_started cameraMatrix commObj	86 86 87 87 87 87 88 88
	6.5.3.2 6.5.3.3 6.5.3.4 6.5.3.5 6.5.3.6 6.5.3.7 6.5.3.8 6.5.3.9 6.5.3.10 6.5.3.11 6.5.3.12	camera_started cameraMatrix commObj coordinateFrame coordinateFrameProjected currentMinIndex currentPointDistance data datagram distCoeffs distModel	86 86 87 87 87 87 88 88

vi

6.5.3.16	frameTime	89
6.5.3.17	gotOrder	89
6.5.3.18	headingOffset	89
6.5.3.19	intExposure	89
6.5.3.20	intFrameRate	90
6.5.3.21	intIntensity	90
6.5.3.22	intThreshold	90
6.5.3.23	invertZ	90
6.5.3.24	IPAdressObject	90
6.5.3.25	IPAdressSafety	91
6.5.3.26	IPAdressSafety2	91
6.5.3.27	list_points2d	91
6.5.3.28	list_points2dDifference	91
6.5.3.29	list_points2dOld	91
6.5.3.30	list_points2dProjected	92
6.5.3.31	list_points2dUnsorted	92
6.5.3.32	list_points3d	92
6.5.3.33	logDate	92
6.5.3.34	logfile	92
6.5.3.35	logFileName	93
6.5.3.36	logName	93
6.5.3.37	M_CN	93
6.5.3.38	M_HeadingOffset	93
6.5.3.39	methodPNP	93
6.5.3.40	minPointDistance	94
6.5.3.41	numberMarkers	94
6.5.3.42	pointOrderIndices	94
6.5.3.43	pointOrderIndicesNew	94
6.5.3.44	portObject	94
6.5.3.45	portSafety	95

CONTENTS vii

		6.5.3.46	portSafety2	95
		6.5.3.47	position	95
		6.5.3.48	positionOld	95
		6.5.3.49	posRef	95
		6.5.3.50	Rmat	96
		6.5.3.51	RmatRef	96
		6.5.3.52	Rvec	96
		6.5.3.53	RvecOriginal	96
		6.5.3.54	safety2Enable	96
		6.5.3.55	safetyAngle	97
		6.5.3.56	safetyBoxLength	97
		6.5.3.57	safetyEnable	97
		6.5.3.58	SS	97
		6.5.3.59	strBuf	97
		6.5.3.60	timeFirstFrame	98
		6.5.3.61	timeOld	98
		6.5.3.62	Tvec	98
		6.5.3.63	TvecOriginal	98
		6.5.3.64	udpSocketObject	98
		6.5.3.65	udpSocketSafety	99
		6.5.3.66	udpSocketSafety2	99
		6.5.3.67	useGuess	99
		6.5.3.68	velocity	99
6.6	RigidTr	ack/main.h	h File Reference	100
	6.6.1	Detailed	Description	102
	6.6.2	Function	Documentation	102
		6.6.2.1	calibrateCamera()	103
		6.6.2.2	calibrateGround()	105
		6.6.2.3	closeUDP()	109
		6.6.2.4	determineExposure()	109

viii CONTENTS

		6.6.2.5	determineOrder()	112
		6.6.2.6	drawPositionText()	113
		6.6.2.7	loadCalibration()	114
		6.6.2.8	loadCameraPosition()	115
		6.6.2.9	loadMarkerConfig()	116
		6.6.2.10	projectCoordinateFrame()	118
		6.6.2.11	sendDataUDP()	119
		6.6.2.12	setHeadingOffset()	120
		6.6.2.13	setReference()	121
		6.6.2.14	setUpUDP()	124
		6.6.2.15	startStopCamera()	125
		6.6.2.16	startTracking()	126
		6.6.2.17	testAlgorithms()	131
	6.6.3	Variable I	Documentation	134
		6.6.3.1	commObj	134
		6.6.3.2	invertZ	134
		6.6.3.3	IPAdressObject	134
		6.6.3.4	IPAdressSafety	135
		6.6.3.5	IPAdressSafety2	135
		6.6.3.6	methodPNP	135
		6.6.3.7	portObject	135
		6.6.3.8	portSafety	135
		6.6.3.9	portSafety2	136
		6.6.3.10	safety2Enable	136
		6.6.3.11	safetyAngle	136
		6.6.3.12	safetyBoxLength	136
		6.6.3.13	safetyEnable	136
6.7	RigidTr	ack/precor	mp.hpp File Reference	137
	6.7.1	Macro De	efinition Documentation	137
		6.7.1.1	GET_OPTIMIZED	137

CONTENTS

6.8	RigidTra	ack/resourc	ce.h File Reference	 . 138
	6.8.1	Macro De	finition Documentation	 . 138
		6.8.1.1	IDI_ICON1	 . 138
6.9	RigidTra	ack/RigidTr	rack.cpp File Reference	 . 138
	6.9.1	Detailed D	Description	 . 138
6.10	RigidTra	ack/RigidTr	rack.h File Reference	 . 139
	6.10.1	Detailed D	Description	 . 139
6.11	RigidTra	ack/suppor	rtcode.cpp File Reference	 . 140
	6.11.1	Function [Documentation	 . 141
		6.11.1.1	CBTHookProc()	 . 141
		6.11.1.2	CloseWindow()	 . 142
		6.11.1.3	CreateAppWindow()	 . 143
		6.11.1.4	DrawGLScene()	 . 146
		6.11.1.5	FullscreenToggle()	 . 147
		6.11.1.6	InitGL()	 . 148
		6.11.1.7	LoadGLTextures()	 . 149
		6.11.1.8	main()	 . 149
		6.11.1.9	PopWaitingDialog()	 . 151
		6.11.1.10	PumpMessages()	 . 152
		6.11.1.11	ReSizeGLScene()	 . 152
		6.11.1.12	TimerProc()	 . 153
		6.11.1.13	WinMain()	 . 153
		6.11.1.14	WndProc()	 . 154
	6.11.2	Variable D	Documentation	 . 156
		6.11.2.1	gActive	 . 156
		6.11.2.2	gFullscreen	 . 156
		6.11.2.3	gSoftwareDecimate	 . 156
		6.11.2.4	gWindowHeight	 . 156
		6.11.2.5	gWindowWidth	 . 157
		6.11.2.6	hDC	 . 157

CONTENTS

	6.11.2.7 hHook
	6.11.2.8 hInstance
	6.11.2.9 hRC
	6.11.2.10 hWnd
	6.11.2.11 keys
	6.11.2.12 texture
	6.11.2.13 windowHeight
	6.11.2.14 windowName
	6.11.2.15 windowWidth
6.12 RigidT	rack/supportcode.h File Reference
6.12.1	Macro Definition Documentation
	6.12.1.1 BYTESPERPIXEL
	6.12.1.2 RGBA
	6.12.1.3 WIN32_LEAN_AND_MEAN
6.12.2	Function Documentation
	6.12.2.1 CloseWindow()
	6.12.2.2 CreateAppWindow()
	6.12.2.3 DrawGLScene()
	6.12.2.4 FullscreenToggle()
	6.12.2.5 PopWaitingDialog()
	6.12.2.6 PumpMessages()
	6.12.2.7 WndProc()
6.12.3	Variable Documentation
	6.12.3.1 gActive
	6.12.3.2 gFullscreen
	6.12.3.3 hDC
	6.12.3.4 keys

Index

171

Chapter 1

Rigid Track Doxygen Documentation

1.1 Introduction

Rigid Track is a software that provides, combined with an OptiTrack camera, the pose estimation of one object in three dimensional space. This is achieved with only one camera in combination with reflective markers. Those are attached to the object ought to be tracked. The accuracy in the range of millimeters and the high update rate of 100 Hz enable use cases for fast and agile objects. The main application is navigation for drones that rely on high precision position data. Where GPS is not available, e.g. indoors or due to a lacking GPS receiver, this setup substitutes for it. Another use case is the pure pose logging when the drone does not depend on the position, e.g. when it is remote piloted by hand. While this setup contains one OptiTrack Flex 3 camera, every other model of OptiTrack should work, despite not tested. With better camera models, e.g. the Prime Series, even outdoor usage is possible. When the capabilities are not sufficient please refer to OptiTracks Software Motive. But keep in mind that this solution needs at least 3 cameras as Rigid Track works with only one.

1.2 Rigid Track Installation

Start the RigidTrack_setup.exe from the enclosed SD card and follow the instructions given in the installation assistant. Default parameters like installation directory or shortcuts to be created can be chosen. But normally clicking Next and keeping the default values should be sufficient. When the installation is completed a shortcut in the start menu and the desktop can be used to start Rigid Track. The program is then successfully installed in C:/Program Files (x86)/TU Munich FSD/Rigid Track.

1.3 Source Code

The most interesting file for you is main.cpp. It contains the relevant functions for pose estimation. Camera calibration and other functional aspects are also implemented there. The GUI program code is found in RigidTrack.cpp. communication.cpp deals only with communication from main.cpp to the GUI.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

CvModelEstimator2	15
QMainWindow	
RigidTrack	19
QObject	
commObject	
Surface	34

4 Hierarchical Index

Chapter 3

Class Index

3.1 Class List

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

commObject	9
CvModelEstimator2	15
RigidTrack	19
Surface	34

6 Class Index

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

8 File Index

Chapter 5

Class Documentation

5.1 commObject Class Reference

#include <communication.h>

Inherits QObject.

Signals

- void statusChanged (QString newText)
- void imageChanged (QPixmap image)
- void logAdded (QString LogText)
- void logCleared ()
- void P3Penabled (bool value)
- void progressUpdated (int value)

Public Member Functions

- void changeStatus (QString newText)
- void changeImage (QPixmap image)
- void addLog (QString LogText)
- void clearLog ()
- void enableP3P (bool value)
- void progressUpdate (int value)

5.1.1 Detailed Description

Definition at line 7 of file communication.h.

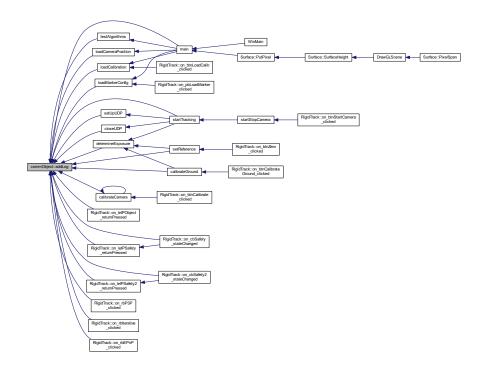
5.1.2 Member Function Documentation

5.1.2.1 addLog()

Definition at line 20 of file communication.cpp.

```
20
21
22    emit logAdded(LogText);
23    QCoreApplication::processEvents();
24 }
```

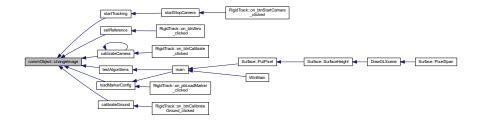
Here is the caller graph for this function:



5.1.2.2 changelmage()

Definition at line 14 of file communication.cpp.

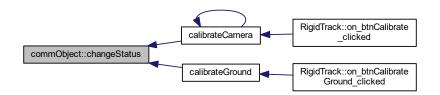
Here is the caller graph for this function:



5.1.2.3 changeStatus()

Definition at line 8 of file communication.cpp.

Here is the caller graph for this function:



5.1.2.4 clearLog()

```
void commObject::clearLog ( )
```

Definition at line 26 of file communication.cpp.

5.1.2.5 enableP3P()

```
void commObject::enableP3P (
          bool value )
```

Definition at line 32 of file communication.cpp.

```
33 {
34         emit P3Penabled(value);
35         QCoreApplication::processEvents();
36 }
```

Here is the caller graph for this function:



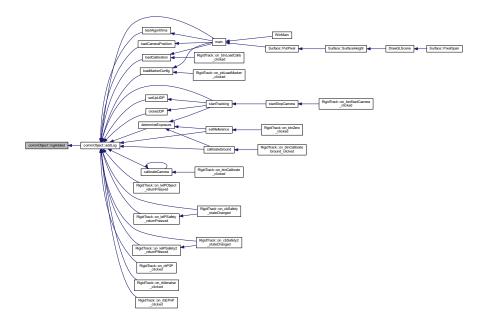
5.1.2.6 imageChanged

Here is the caller graph for this function:



5.1.2.7 logAdded

Here is the caller graph for this function:



5.1.2.8 logCleared

```
void commObject::logCleared ( ) [signal]
```

Here is the caller graph for this function:



5.1.2.9 P3Penabled

```
void commObject::P3Penabled (
          bool value ) [signal]
```

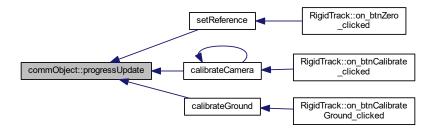
Here is the caller graph for this function:



5.1.2.10 progressUpdate()

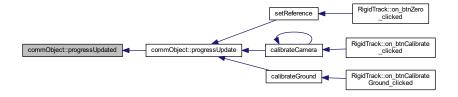
Definition at line 38 of file communication.cpp.

Here is the caller graph for this function:



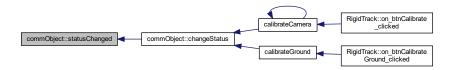
5.1.2.11 progressUpdated

Here is the caller graph for this function:



5.1.2.12 statusChanged

Here is the caller graph for this function:



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

- · RigidTrack/communication.h
- RigidTrack/communication.cpp

5.2 CvModelEstimator2 Class Reference

```
#include <_modelest.h>
```

Public Member Functions

- CvModelEstimator2 (int _modelPoints, CvSize _modelSize, int _maxBasicSolutions)
- virtual ∼CvModelEstimator2 ()
- virtual int runKernel (const CvMat *m1, const CvMat *m2, CvMat *model)=0
- virtual bool runLMeDS (const CvMat *m1, const CvMat *m2, CvMat *model, CvMat *mask, double confidence=0.99, int maxIters=2000)
- virtual bool runRANSAC (const CvMat *m1, const CvMat *m2, CvMat *model, CvMat *mask, double threshold, double confidence=0.99, int maxIters=2000)
- virtual bool refine (const CvMat *, const CvMat *, CvMat *, int)
- virtual void setSeed (int64 seed)

Protected Member Functions

- virtual void computeReprojError (const CvMat *m1, const CvMat *m2, const CvMat *model, CvMat *error)=0
- virtual int findInliers (const CvMat *m1, const CvMat *m2, const CvMat *model, CvMat *error, CvMat *mask, double threshold)
- virtual bool getSubset (const CvMat *m1, const CvMat *m2, CvMat *ms1, CvMat *ms2, int max
 — Attempts=1000)
- virtual bool checkSubset (const CvMat *ms1, int count)

Protected Attributes

- CvRNG rng
- int modelPoints
- CvSize modelSize
- · int maxBasicSolutions
- · bool checkPartialSubsets

5.2.1 Detailed Description

Definition at line 48 of file _modelest.h.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 CvModelEstimator2()

```
CvModelEstimator2::CvModelEstimator2 (
    int _modelPoints,
    CvSize _modelSize,
    int _maxBasicSolutions )
```



```
virtual CvModelEstimator2::~CvModelEstimator2 ( ) [virtual]
```

5.2.3 Member Function Documentation

5.2.3.1 checkSubset()

5.2.3.2 computeReprojError()

5.2.3.3 findInliers()

5.2.3.4 getSubset()

5.2.3.5 refine()

Definition at line 60 of file _modelest.h.

```
60 { return true; }
```

5.2.3.6 runKernel()

5.2.3.7 runLMeDS()

5.2.3.8 runRANSAC()

5.2.3.9 setSeed()

5.2.4 Member Data Documentation

5.2.4.1 checkPartialSubsets

```
bool CvModelEstimator2::checkPartialSubsets [protected]
```

Definition at line 77 of file _modelest.h.

5.2.4.2 maxBasicSolutions

int CvModelEstimator2::maxBasicSolutions [protected]

Definition at line 76 of file _modelest.h.

5.2.4.3 modelPoints

int CvModelEstimator2::modelPoints [protected]

Definition at line 74 of file _modelest.h.

5.2.4.4 modelSize

CvSize CvModelEstimator2::modelSize [protected]

Definition at line 75 of file _modelest.h.

5.2.4.5 rng

CvRNG CvModelEstimator2::rng [protected]

Definition at line 73 of file _modelest.h.

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

• RigidTrack/_modelest.h

5.3 RigidTrack Class Reference

#include <RigidTrack.h>

Inherits QMainWindow.

Public Slots

- void on_btnStartCamera_clicked ()
- void on_btnZero_clicked ()
- · void on_btnCalibrate_clicked ()
- void setImage (QPixmap image)
- void clearLog ()
- void progressUpdate (int value)
- · void on_btnLoadCalib_clicked ()
- void setLog (QString logText)
- void on sbHeadingOffset valueChanged (double d)
- void on_leIPObject_returnPressed ()
- · void on_leIPSafety_returnPressed ()
- · void on leIPSafety2 returnPressed ()
- void on_rbP3P_clicked ()
- void on_rblterative_clicked ()
- void on_rbEPnP_clicked ()
- void on_actionShow_Help_triggered ()
- void on_cbSafety_stateChanged (int state)
- void on cbSafety2 stateChanged (int state)
- void on_dsbDimension_valueChanged (double d)
- · void on_sbAngle_valueChanged (int i)
- void on_pbLoadMarker_clicked ()
- void on_cbInvert_stateChanged (int state)
- void enableP3P (bool value)
- void on_btnCalibrateGround_clicked ()
- void on_actionOpen_Log_Folder_triggered ()
- void on_actionAbout_Rigid_Track_triggered ()
- void on_actionOpen_Installation_Folder_triggered ()

Public Member Functions

RigidTrack (QWidget *parent=Q_NULLPTR)

5.3.1 Detailed Description

Definition at line 17 of file RigidTrack.h.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 RigidTrack()

Definition at line 20 of file RigidTrack.cpp.

```
21 : QMainWindow(parent)
22 {
23     ui.setupUi(this);
24
25 }
```

5.3.3 Member Function Documentation

5.3.3.1 clearLog

```
void RigidTrack::clearLog ( ) [slot]
```

Definition at line 42 of file RigidTrack.cpp.

```
43 {
44      ui.listLog->reset();
45 }
```

5.3.3.2 enableP3P

Definition at line 229 of file RigidTrack.cpp.

5.3.3.3 on_actionAbout_Rigid_Track_triggered

```
\verb"void RigidTrack::on_actionAbout_Rigid_Track_triggered" ( ) \quad [\verb|slot||]
```

Definition at line 245 of file RigidTrack.cpp.

5.3.3.4 on_actionOpen_Installation_Folder_triggered

```
void RigidTrack::on_actionOpen_Installation_Folder_triggered ( ) [slot]
```

Definition at line 253 of file RigidTrack.cpp.

```
254 {
255          QString command = "explorer.exe " + QDir::currentPath().replace("/", "\\");
256          QProcess::startDetached(command);
257
258 }
```

5.3.3.5 on_actionOpen_Log_Folder_triggered

```
void RigidTrack::on_actionOpen_Log_Folder_triggered ( ) [slot]
```

Definition at line 239 of file RigidTrack.cpp.

5.3.3.6 on_actionShow_Help_triggered

```
void RigidTrack::on_actionShow_Help_triggered ( ) [slot]
```

Definition at line 160 of file RigidTrack.cpp.

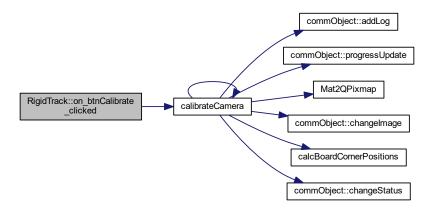
5.3.3.7 on_btnCalibrate_clicked

```
void RigidTrack::on_btnCalibrate_clicked ( ) [slot]
```

Definition at line 32 of file RigidTrack.cpp.

```
33 {
34     calibrateCamera();
35 }
```

Here is the call graph for this function:

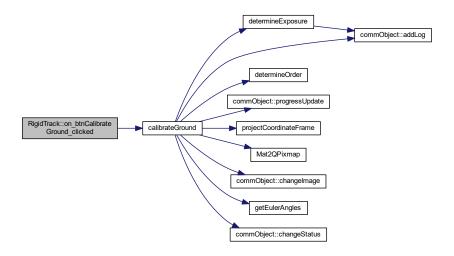


5.3.3.8 on_btnCalibrateGround_clicked

```
void RigidTrack::on_btnCalibrateGround_clicked ( ) [slot]
```

Definition at line 234 of file RigidTrack.cpp.

Here is the call graph for this function:



5.3.3.9 on_btnLoadCalib_clicked

```
void RigidTrack::on_btnLoadCalib_clicked ( ) [slot]
```

Definition at line 52 of file RigidTrack.cpp.

```
53 {
54         loadCalibration(1);
55 }
```

Here is the call graph for this function:



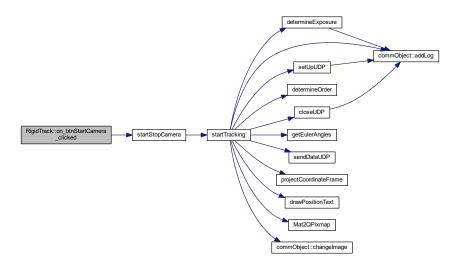
5.3.3.10 on_btnStartCamera_clicked

```
void RigidTrack::on_btnStartCamera_clicked ( ) [slot]
```

Definition at line 260 of file RigidTrack.cpp.

```
261 {
262     if(RigidTrack::ui.btnStartCamera->text() == "Start Tracking")
263     {
264         RigidTrack::ui.btnStartCamera->setText("Stop Tracking");
265     }
266     else
267     {
268         RigidTrack::ui.btnStartCamera->setText("Start Tracking");
269     }
270     startStopCamera();
```

Here is the call graph for this function:



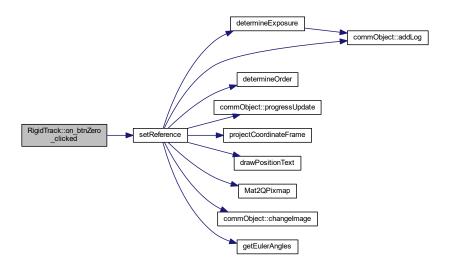
5.3.3.11 on_btnZero_clicked

```
void RigidTrack::on_btnZero_clicked ( ) [slot]
```

Definition at line 27 of file RigidTrack.cpp.

```
28 {
29     setReference();
30 }
```

Here is the call graph for this function:



5.3.3.12 on_cbInvert_stateChanged

Definition at line 217 of file RigidTrack.cpp.

```
218 {
219          if (state)
220          {
221               invertZ = -1;
222          }
223          else
224          {
225                invertZ = 1;
226          }
227 }
```

5.3.3.13 on_cbSafety2_stateChanged

Definition at line 187 of file RigidTrack.cpp.

```
188 {
189
        RigidTrack::ui.leIPSafety2->setEnabled(state);
190
        safety2Enable = state;
191
        if (state)
192
193
            commObj.addLog("Enabled second Receiver");
194
            on_leIPSafety2_returnPressed();
195
196
197
            commObj.addLog("Disabled second Receiver");
198
199
200 }
```

Here is the call graph for this function:



5.3.3.14 on_cbSafety_stateChanged

Definition at line 170 of file RigidTrack.cpp.

```
171 {
172
        RigidTrack::ui.dsbDimension->setEnabled(state);
173
        RigidTrack::ui.sbAngle->setEnabled(state);
174
        safetyEnable = state;
175
        RigidTrack::ui.leIPSafety->setEnabled(state);
176
        if (state)
177
            commObj.addLog("Enabled Safety Area Protection");
178
179
            on_leIPSafety_returnPressed();
180
181
        else
182
            commObj.addLog("Disabled Safety Area Protection");
183
184
185 }
```



5.3.3.15 on_dsbDimension_valueChanged

```
void RigidTrack::on_dsbDimension_valueChanged ( \label{eq:double} \mbox{double } d \; ) \quad [\mbox{slot}]
```

Definition at line 202 of file RigidTrack.cpp.

5.3.3.16 on_lelPObject_returnPressed

```
void RigidTrack::on_leIPObject_returnPressed ( ) [slot]
```

Definition at line 68 of file RigidTrack.cpp.

```
69 {
70
71
       QString adress = RigidTrack::ui.leIPObject->text();
       IPAdressObject = QHostAddress(adress.split(":")[0]);
if (IPAdressObject.isNull() || adress.split(":").length() == 1 || adress.split(":")[1]==0
72
73
74
       {
75
            throw 2;
76
       portObject = adress.split(":")[1].toInt();
78
       commObj.addLog("Object IP changed to:");
       commObj.addLog(IPAdressObject.toString());
79
80
       commObj.addLog("Object Port changed to:");
81
       commObj.addLog(QString::number(portObject));
82
83
       catch (...)
85
            commObj.addLog("Error Changing the IP Adress or Port! Restored Standard Values
       192.168.0.1:9155");
           IPAdressObject = QHostAddress("192.168.0.1");
portObject = 9155;
86
87
            RigidTrack::ui.leIPObject->setText("192.168.0.1:9155");
88
       }
90 }
```

```
RigidTrack::on_lelPObject commObject::addLog
```

5.3.3.17 on_lelPSafety2_returnPressed

```
void RigidTrack::on_leIPSafety2_returnPressed ( ) [slot]
```

Definition at line 117 of file RigidTrack.cpp.

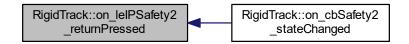
```
118 {
119
120
         QString adress = RigidTrack::ui.leIPSafety2->text();
121
         IPAdressSafety2 = QHostAddress(adress.split(":")[0]);
         if (IPAdressSafety2.isNull() || adress.split(":").length() == 1 || adress.split(":")[1]
122
       == 0)
123
         {
124
              throw 2;
125
126
         portSafety2 = adress.split(":")[1].toInt();
         commobj.addLog("Receiver 2 IP changed to:");
commobj.addLog(IPAdressSafety2.toString());
commobj.addLog("Receiver 2 Port changed to:");
127
128
129
         commObj.addLog(QString::number(portSafety2));
130
131
132
         catch (...)
133
134
               \verb|commObj.addLog("Error Changing the IP Adress or Port! Restored Standard Values| \\
        192.168.0.1:9155");

IPAdressSafety2 = QHostAddress("192.168.0.1");

portSafety2 = 9155;
135
136
137
              RigidTrack::ui.leIPSafety2->setText("192.168.0.1:9155");
138
139 }
```

Here is the call graph for this function:





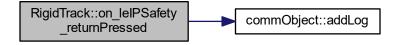
5.3.3.18 on_lelPSafety_returnPressed

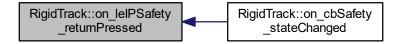
```
void RigidTrack::on_leIPSafety_returnPressed ( ) [slot]
```

Definition at line 93 of file RigidTrack.cpp.

```
94 {
9.5
        trv{
        QString adress = RigidTrack::ui.leIPSafety->text();
IPAdressSafety = QHostAddress(adress.split(":")[0]);
96
        if (IPAdressSafety.isNull() || adress.split(":").length() == 1 || adress.split(":")[1] ==
        0)
99
100
              throw 2;
101
         portSafety = adress.split(":")[1].toInt();
commObj.addLog("Safety Switch IP changed to:");
102
103
104
         commObj.addLog(IPAdressSafety.toString());
         commObj.addLog("Safety Switch Port changed to:");
105
106
         commObj.addLog(QString::number(portSafety));
107
108
         catch (...)
109
110
              commObj.addLog("Error Changing the IP Adress or Port! Restored Standard Values
        192.168.0.1:9155");
              IPAdressSafety = QHostAddress("192.168.0.1");
portSafety = 9155;
111
112
              RigidTrack::ui.leIPSafety->setText("192.168.0.1:9155");
113
114
115
```

Here is the call graph for this function:





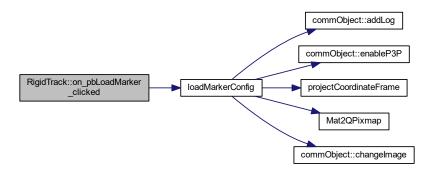
5.3.3.19 on_pbLoadMarker_clicked

```
void RigidTrack::on_pbLoadMarker_clicked ( ) [slot]
```

Definition at line 212 of file RigidTrack.cpp.

```
213 {
214         loadMarkerConfig(1);
215 }
```

Here is the call graph for this function:



5.3.3.20 on_rbEPnP_clicked

```
void RigidTrack::on_rbEPnP_clicked ( ) [slot]
```

Definition at line 154 of file RigidTrack.cpp.



5.3.3.21 on_rblterative_clicked

```
void RigidTrack::on_rbIterative_clicked ( ) [slot]
```

Definition at line 148 of file RigidTrack.cpp.

Here is the call graph for this function:

5.3.3.22 on_rbP3P_clicked

```
void RigidTrack::on_rbP3P_clicked ( ) [slot]
```

Definition at line 142 of file RigidTrack.cpp.



5.3.3.23 on_sbAngle_valueChanged

```
\label{local_constraint} \mbox{ void RigidTrack::on\_sbAngle\_valueChanged (} \\ \mbox{ int } i \mbox{ ) } \mbox{ [slot]}
```

Definition at line 207 of file RigidTrack.cpp.

5.3.3.24 on_sbHeadingOffset_valueChanged

```
\label{local_constraint} \mbox{ void RigidTrack::on\_sbHeadingOffset\_valueChanged (} \\ \mbox{ double } \mbox{ $d$ } \mbox{) } \mbox{ [slot]}
```

Definition at line 63 of file RigidTrack.cpp.

Here is the call graph for this function:



5.3.3.25 progressUpdate

Definition at line 47 of file RigidTrack.cpp.

```
48 {
49          RigidTrack::ui.progressBar->setValue(value);
50 }
```

5.3.3.26 setImage

Definition at line 37 of file RigidTrack.cpp.

```
38 {
39     ui.lbStatus->setPixmap(image);
40 }
```

5.3.3.27 setLog

Definition at line 57 of file RigidTrack.cpp.

```
58 {
59     RigidTrack::ui.listLog->addItem(logText);
60     RigidTrack::ui.listLog->scrollToBottom();
61 }
```

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

- RigidTrack/RigidTrack.h
- RigidTrack/RigidTrack.cpp

5.4 Surface Class Reference

```
#include <supportcode.h>
```

Public Member Functions

```
• Surface (int Width, int Height)
```

- ∼Surface ()
- GLuint GetTexture ()
- void Resize (int Width, int Height)
- int CalculateSize (int Width)
- int Width ()
- int Height ()
- int SurfaceWidth ()
- int SurfaceHeight ()
- void PutPixel (int X, int Y, PIXEL Color)
- unsigned char * GetBuffer ()
- void RebindTexture ()
- int PixelSpan ()

5.4.1 Detailed Description

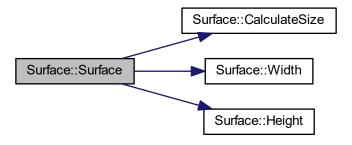
Definition at line 25 of file supportcode.h.

5.4.2 Constructor & Destructor Documentation

5.4.2.1 Surface()

Definition at line 417 of file supportcode.cpp.

```
: buffer(0), mDirty(true)
418 {
419
         //!/<== Use power of 2 texture sizes ==
420
         mSurfaceWidth = CalculateSize(Width);
mSurfaceHeight = CalculateSize(Height);
421
422
423
424
         mSpan = mSurfaceWidth;
425
         mWidth = Width;
mHeight = Height;
426
427
428
         buffer = (unsigned char*) malloc(mSurfaceWidth * mSurfaceHeight *
429
       BYTESPERPIXEL);
430
431
         memset(buffer, 0, mSurfaceWidth * mSurfaceHeight * BYTESPERPIXEL);
432
433
         if(buffer==0)
434
              throw(" nable to allocate surface buffer");
435
436
         glGenTextures(1, &mTexture);
         if(mTexture==0)
    throw(" nable to gen OpenGL texture");
437
438
439
440
441
         glBindTexture(GL_TEXTURE_2D, mTexture);
         glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, mSurfaceWidth, mSurfaceHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE,
         glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
443
444
445 }
```



5.4.2.2 \sim Surface()

```
Surface::~Surface ( )
```

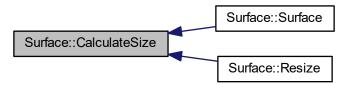
Definition at line 447 of file supportcode.cpp.

```
448 {
449
450 }
```

5.4.3 Member Function Documentation

5.4.3.1 CalculateSize()

Definition at line 494 of file supportcode.cpp.



5.4.3.2 GetBuffer()

```
unsigned char* Surface::GetBuffer ( ) [inline]
```

Definition at line 40 of file supportcode.h.

```
40 { return buffer; }
```

Here is the call graph for this function:



5.4.3.3 GetTexture()

```
GLuint Surface::GetTexture ( )
```

Definition at line 504 of file supportcode.cpp.



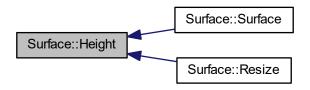
5.4.3.4 Height()

```
int Surface::Height ( ) [inline]
```

Definition at line 35 of file supportcode.h.

```
35 { return mHeight; }
```

Here is the caller graph for this function:

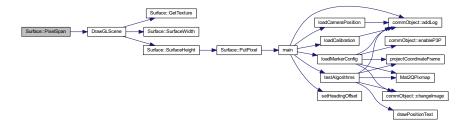


5.4.3.5 PixelSpan()

```
int Surface::PixelSpan ( ) [inline]
```

Definition at line 42 of file supportcode.h.

```
42 { return mSpan; }
```

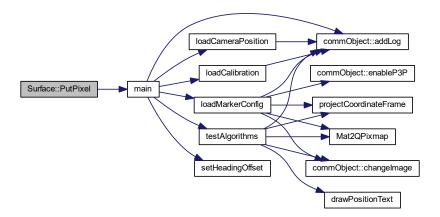


5.4.3.6 PutPixel()

```
void Surface::PutPixel (
          int X,
          int Y,
          PIXEL Color )
```

Definition at line 514 of file supportcode.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



5.4.3.7 RebindTexture()

```
void Surface::RebindTexture ( )
```

Definition at line 452 of file supportcode.cpp.

```
453 {
454
         glGenTextures(1, &mTexture);
455
          if (mTexture==0)
              throw(" nable to gen OpenGL texture");
456
457
458
         glBindTexture(GL_TEXTURE_2D, mTexture);
459
460
         glTexImage2D(GL_TEXTURE_2D, 0, GL_RGBA8, mSurfaceWidth, mSurfaceHeight, 0, GL_RGBA, GL_UNSIGNED_BYTE,
       buffer);
         glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MIN_FILTER,GL_LINEAR);
glTexParameteri(GL_TEXTURE_2D,GL_TEXTURE_MAG_FILTER,GL_LINEAR);
461
462
463 }
```

Here is the caller graph for this function:

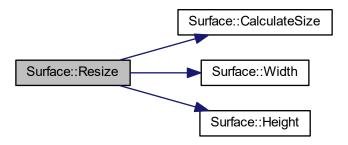


5.4.3.8 Resize()

Definition at line 466 of file supportcode.cpp.

```
467 {
468
        if (Width==mWidth && Height==mHeight)
469
             return;
470
471
        if(Width<1 || Height<1)</pre>
472
             return:
473
474
        int newSize = CalculateSize(Width);
475
476
         if(newSize>mSurfaceWidth || newSize>mSurfaceHeight)
477
478
             if(buffer!=0)
479
                 free (buffer);
480
481
             mSurfaceWidth = newSize;
482
             mSurfaceHeight = newSize;
             mSpan = mSurfaceWidth;
mWidth = Width;
483
484
             mHeight = Height;
buffer = (unsigned char*) malloc(mSurfaceWidth * mSurfaceHeight *
485
486
      BYTESPERPIXEL);
487
            if(buffer==0)
                 throw("Unable to allocate surface buffer");
488
489
490
             mDirty = true;
491
        }
492 }
```

Here is the call graph for this function:



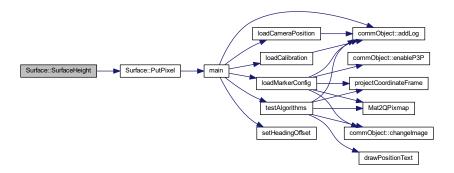
5.4.3.9 SurfaceHeight()

```
int Surface::SurfaceHeight ( ) [inline]
```

Definition at line 37 of file supportcode.h.

```
37 { return mSurfaceHeight; }
```

Here is the call graph for this function:





5.4.3.10 SurfaceWidth()

```
int Surface::SurfaceWidth ( ) [inline]
```

Definition at line 36 of file supportcode.h.

```
36 { return mSurfaceWidth; }
```

Here is the caller graph for this function:



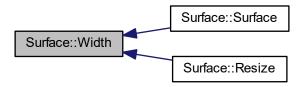
5.4.3.11 Width()

```
int Surface::Width ( ) [inline]
```

Definition at line 34 of file supportcode.h.

```
34 { return mWidth; }
```

Here is the caller graph for this function:



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

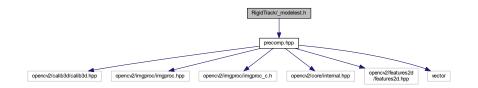
- RigidTrack/supportcode.h
- RigidTrack/supportcode.cpp

Chapter 6

File Documentation

6.1 RigidTrack/_modelest.h File Reference

#include "precomp.hpp"
Include dependency graph for _modelest.h:



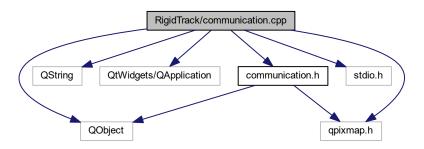
Classes

• class CvModelEstimator2

6.2 RigidTrack/communication.cpp File Reference

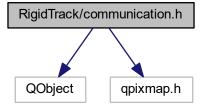
```
#include <QObject>
#include <QString>
#include <QtWidgets/QApplication>
#include <qpixmap.h>
#include <stdio.h>
```

#include "communication.h"
Include dependency graph for communication.cpp:

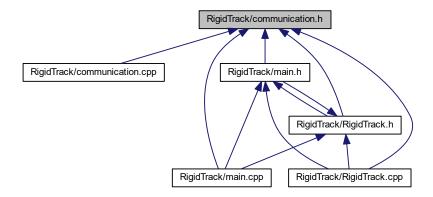


6.3 RigidTrack/communication.h File Reference

#include <QObject>
#include <qpixmap.h>
Include dependency graph for communication.h:



This graph shows which files directly or indirectly include this file:



Classes

· class commObject

6.4 RigidTrack/DoxygenMain.md File Reference

6.5 RigidTrack/main.cpp File Reference

Rigid Track main file that contains most functionallity.

```
#include "RigidTrack.h"
#include "main.h"
#include "communication.h"
#include "cameralibrary.h"
#include "modulevector.h"
#include "modulevectorprocessing.h"
#include "coremath.h"
#include <QtWidgets/QApplication>
#include <QDesktopServices>
#include <QInputDialog>
#include <QUrl>
#include <QThread>
#include <QUdpSocket>
#include <QFileDialog>
#include <opencv\cv.h>
#include "opencv2\core.hpp"
#include "opencv2\calib3d.hpp"
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2\video\tracking.hpp>
#include <fstream>
#include <windows.h>
```

```
#include <conio.h>
#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <stdarg.h>
#include <ctype.h>
#include <stdlib.h>
#include <gl/glu.h>
#include <sstream>
#include <time.h>
#include <cmath>
#include <vector>
#include <algorithm>
#include <random>
#include <thread>
#include <strsafe.h>
```

Include dependency graph for main.cpp:



Functions

• int main (int argc, char *argv[])

main initialises the GUI and values for the marker position etc

- QPixmap Mat2QPixmap (cv::Mat src)
- void calcBoardCornerPositions (Size boardSize, float squareSize, std::vector < Point3f > &corners)
- void getEulerAngles (Mat &rotCamerMatrix, Vec3d &eulerAngles)
- int startTracking ()
- void startStopCamera ()

Start or stop the tracking depending on if the camera is currently running or not.

- int setReference ()
- int calibrateCamera ()

Start the camera calibration routine that computes the camera matrix and distortion coefficients.

- void loadCalibration (int method)
- · void testAlgorithms ()
- void projectCoordinateFrame (Mat pictureFrame)
- void setUpUDP ()

Open the UDP ports for communication.

- void setHeadingOffset (double d)
- void sendDataUDP (cv::Vec3d &Position, cv::Vec3d &Euler)
- void closeUDP ()
- void loadMarkerConfig (int method)
- void drawPositionText (cv::Mat &Picture, cv::Vec3d &Position, cv::Vec3d &Euler, double error)
- void loadCameraPosition ()
- int determineExposure ()
- void determineOrder ()
- int calibrateGround ()

Variables

commObject commObj

class that handles the communication from main.cpp to the GUI

• bool safetyEnable = false

is the safety feature enabled

• bool safety2Enable = false

is the second receiver enabled

double safetyBoxLength = 1.5

length of the safety area cube in meters

• int safetyAngle = 30

bank and pitch angle protection in degrees

bool exitRequested = true

variable if tracking loop should be exited

• int invertZ = 1

dummy variable to invert Z direction on request

double frameTime = 0.01

100 Hz CoSy rate, is later on replaced with the hardware timestamp delivered by the camera

• double timeOld = 0.0

old time for finite differences velocity calculation. Is later on replaced with the hardware timestamp delivered by the camera

double timeFirstFrame = 0

Time stamp of the first frame. This value is then subtracted for every other frame so the time in the log start at zero.

Vec3d position = Vec3d()

position vector x,y,z for object position in O-CoSy, unit is meter

Vec3d eulerAngles = Vec3d()

Roll Pitch Heading in this order, units in degrees.

Vec3d positionOld = Vec3d()

old position in O-CoSy for finite differences velocity calculation

Vec3d velocity = Vec3d()

velocity vector of object in o-CoSy in respect to o-CoSy

• Vec3d posRef = Vec3d()

initial position of object in camera CoSy

Vec3d eulerRef = Vec3d()

initial euler angle of object respectivley to camera CoSy

• double headingOffset = 0

heading offset variable for aligning INS heading with tracking heading

• int intlntensity = 15

max infrared spot light intensity is 15 1-6 is strobe 7-15 is continuous 13 and 14 are meaningless

• int intExposure = 1

max is 480 increase if markers are badly visible but should be determined automatically during setReference()

• int intFrameRate = 100

CoSy rate of camera, maximum is 100 fps.

• int intThreshold = 200

threshold value for marker detection. If markers are badly visible lower this value but should not be necessary

• Mat Rmat = (cv::Mat < double > (3, 1) << 0.0, 0.0, 0.0)

Rotation, translation etc. matrix for PnP results.

Mat RmatRef = (cv::Mat_<double>(3, 3) << 1., 0., 0., 0., 1., 0., 0., 1.)

reference rotation matrix from camera CoSy to marker CoSy

• Mat M_CN = cv::Mat_<double>(3, 3)

rotation matrix from camera to ground, fixed for given camera position

Mat M_HeadingOffset = cv::Mat_<double>(3, 3)

rotation matrix that turns the ground system to the INS magnetic heading for alignment

Mat Rvec = (cv::Mat_<double>(3, 1) << 0.0, 0.0, 0.0)

rotation vector (axis-angle notation) from camera CoSy to marker CoSy

Mat Tvec = (cv::Mat_<double>(3, 1) << 0.0, 0.0, 0.0)

translation vector from camera CoSy to marker CoSy in camera CoSy

· Mat RvecOriginal

initial values as start values for algorithms and algorithm tests

Mat TvecOriginal

initial values as start values for algorithms and algorithm tests

• bool useGuess = true

set to true and the algorithm uses the last result as starting value

• int methodPNP = 0

solvePNP algorithm $0 = iterative \ 1 = EPNP \ 2 = P3P \ 4 = UPNP //! < 4$ and 1 are the same and not implemented correctly by OpenCV

• int numberMarkers = 4

number of markers. Is loaded during start up from the marker configuration file

std::vector< Point3d > list_points3d

marker positions in marker CoSy

std::vector< Point2d > list_points2d

marker positions projected in 2D in camera image CoSy

std::vector< Point2d > list_points2dOld

marker positions in previous picture in 2D in camera image CoSy

std::vector< double > list_points2dDifference

difference of the old and new 2D marker position to determine the order of the points

std::vector< Point2d > list_points2dProjected

3D marker points projected to 2D in camera image CoSy with the algorithm projectPoints

std::vector< Point2d > list_points2dUnsorted

 $marker\ points\ in\ 2D\ camera\ image\ CoSy,\ sorted\ with\ increasing\ x\ (camera\ image\ CoSy)\ but\ not\ sorted\ to\ correspond\ with\ list\ points3d$

std::vector< Point3d > coordinateFrame

coordinate visualisazion of marker CoSy

std::vector< Point2d > coordinateFrameProjected

marker CoSy projected from 3D to 2D camera image CoSy

int pointOrderIndices [] = { 0, 1, 2, 3 }

old correspondence from list_points3d and list_points_2d

• int pointOrderIndicesNew [] = $\{0, 1, 2, 3\}$

new correspondence from list_points3d and list_points_2d

double currentPointDistance = 5000

distance from the projected 3D points (hence in 2d) to the real 2d marker positions in camera image CoSy

• double minPointDistance = 5000

minimum distance from the projected 3D points (hence in 2d) to the real 2d marker positions in camera image CoSy

int currentMinIndex = 0

helper variable set to the point order that holds the current minimum point distance

• bool gotOrder = false

order of the list_points3d and list_points3d already tetermined or not, has to be done once

bool camera_started = false

variable thats needed to exit the main while loop

· Mat cameraMatrix

camera matrix of the camera

· Mat distCoeffs

distortion coefficients of the camera

• Core::DistortionModel distModel

distortion model of the camera

QUdpSocket * udpSocketObject

socket for the communication with receiver 1

QUdpSocket * udpSocketSafety

socket for the communication with safety receiver

QUdpSocket * udpSocketSafety2

socket for the communication with receiver 3

QHostAddress IPAdressObject = QHostAddress("127.0.0.1")

IPv4 adress of receiver 1.

QHostAddress IPAdressSafety = QHostAddress("192.168.4.1")

IPv4 adress of safety receiver.

QHostAddress IPAdressSafety2 = QHostAddress("192.168.4.4")

IPv4 adress of receiver 2.

• int portObject = 9155

Port of receiver 1.

• int portSafety = 9155

Port of the safety receiver.

int portSafety2 = 9155

Port of receiver 2.

· QByteArray datagram

data package that is sent to receiver 1 and 2

QByteArray data

data package that's sent to the safety receiver

• const int BACKBUFFER_BITSPERPIXEL = 8

8 bit per pixel and greyscale image from camera

· std::string strBuf

buffer that holds the strings that are sent to the Qt GUI

• std::stringstream ss

stream that sends the strBuf buffer to the Qt GUI

QString logFileName

Filename for the logfiles.

std::string logName

Filename for the logfiles as standard string.

• SYSTEMTIME logDate

Systemtime struct that saves the current date and time thats needed for the log file name creation.

• std::ofstream logfile

file handler for writing the log file

6.5.1 Detailed Description

Rigid Track main file that contains most functionallity.

This file contains allmost all functional code for pose estimation, calibration and so on. The GUI related part is in RigidTrack.cpp and the communication from main.cpp to GUI is done with the commObj class from communication.cpp.

Author

Florian J.T. Wachter

Version

1.0

Date

April, 8th 2017

6.5.2 Function Documentation

6.5.2.1 calcBoardCornerPositions()

Calculate the chess board corner positions, used for the camera calibration.

Parameters

in	boardSize	denotes how many squares are in each direction.
in	squareSize	is the square length in millimeters.
out	corners	returns the square corners in millimeters.

Definition at line 229 of file main.cpp.



6.5.2.2 calibrateCamera()

```
int calibrateCamera ( )
```

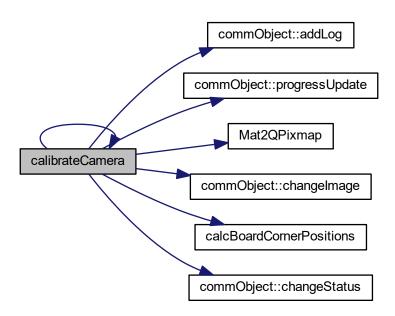
Start the camera calibration routine that computes the camera matrix and distortion coefficients.

Definition at line 774 of file main.cpp.

```
775 {
776
        commObj.addLog("Started camera calibration. 80 pictures are going to be captured.");
777
        CameraLibrary_EnableDevelopment();
778
779
        //! Initialize Camera SDK ==--
780
        CameraLibrary::CameraManager::X();
781
782
        //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
783
        //! them on it's own.
784
785
        //! Get a connected camera =======
786
        CameraManager::X().WaitForInitialization();
787
788
        Camera *camera = CameraManager::X().GetCamera();
789
        if (camera == 0)
790
        {
791
             commObj.addLog("No camera found!");
792
            return 1;
793
794
        //! Determine camera resolution
795
796
        int cameraWidth = camera->Width();
        int cameraHeight = camera->Height();
797
798
799
        //! Set Video Mode ==--
800
801
        //! We set the camera to Segment Mode here. This mode is support by all of our products.
        //! Depending on what device you have connected you might want to consider a different
802
        //! video mode to achieve the best possible tracking quality. All devices that support a
803
        //! mode that will achieve a better quality output with a mode other than Segment Mode are
805
            listed here along with what mode you should use if you're looking for the best head
806
        //! tracking:
807
                 V100:R1/R2
808
                                Precision Mode
                               Bit-Packed Precision Mode
809
                 TrackIR 5
810
                                Precision Mode
811
                               Precision Mode
812
                 S250e
                                Precision Mode
813
        //! If you have questions about a new device that might be conspicuously missing here or //! have any questions about head tracking, email support or participate in our forums.
814
815
816
817
        camera->SetVideoType(Core::GrayscaleMode);
818
819
        //! Start camera output ==--
820
        camera->Start();
821
822
        //! Camera Matrix creation
823
        cameraMatrix = Mat::eye(3, 3, CV_64F);
        distCoeffs = Mat::zeros(8, 1, CV_64F);
824
825
826
        //! Ok, start main loop. This loop fetches and displays ===---
827
        //! camera frames.
828
        //! But first set some camera parameters
829
        camera->SetAGC(false);
830
        camera->SetAEC(false);
831
        camera->SetExposure(200);
832
        camera->SetIntensity(4);
833
        camera->SetFrameRate(30);
834
        camera->SetIRFilter(true);
835
        camera->SetContinuousIR(false);
836
        camera->SetHighPowerMode(false);
837
838
        int number_samples = 0;
839
        int imagesToSample = 80;
840
841
        std::vector<std::vector<Point2f> > imagePoints;
        std::vector<Point2f> pointBuf;
842
843
        bool found;
844
        Size boardSize(9, 6);
        Size imageSize(cameraWidth, cameraHeight);
845
        Mat Rvec(3, 1, DataType<double>::type);
Mat Tvec(3, 1, DataType<double>::type);
846
847
848
```

```
//! the user has to provide the size of one square in mm
850
               int qsquareSize = QInputDialog::getInt(nullptr, "Chessboard size in mm", "Chessboard size in mm", 23, 1
851
           , 60, 1, &ok);
               float squareSize = 23;
852
853
854
               if (ok)
855
856
                       squareSize = qsquareSize;
857
858
859
               QPixmap QPFrame;
860
               commObj.progressUpdate(0);
861
               while (number_samples < imagesToSample)</pre>
862
863
                       //! Fetch a new frame from the camera ===---
864
                      cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV 8UC1);
865
866
                      //! which is why we also set this constant to 8
                      const int BACKBUFFER_BITSPERPIXEL = 8;
867
868
869
                       //! later on, when we get the frame as usual:
870
                      CameraLibrary::Frame * frame = camera->GetFrame();
871
872
                       if (frame)
873
                               //! Lets have the Camera Library raster the camera's
874
875
                              //! image into our texture.
876
                              {\tt frame-} \\ {\tt Rasterize} ({\tt cameraWidth, cameraHeight, matFrame.step, BACKBUFFER\_BITSPERPIXEL, matFrame.step)} \\ {\tt raterize} ({\tt cameraWidth, cameraHeight, matFrame.step)} \\ {\tt raterize} ({\tt cameraWidth, camera
877
           data):
878
                              QPFrame = Mat2QPixmap(matFrame);
                              commObj.changeImage(QPFrame);
found = findChessboardCorners(matFrame, boardSize, pointBuf, CV_CALIB_CB_ADAPTIVE_THRESH |
879
880
           CV_CALIB_CB_FAST_CHECK | CV_CALIB_CB_NORMALIZE_IMAGE);
881
882
                               if (found)
                                                                               //!< If done with success,
883
884
                                      //! improve the found corners' coordinate accuracy for chessboard
885
                                      cornerSubPix(matFrame, pointBuf, Size(11, 11), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS +
             CV_TERMCRIT_ITER, 30, 0.1));
886
887
                                      imagePoints.push_back(pointBuf);
888
                                      number_samples += 1;
                                      commObj.addLog(QString::fromStdString(ss.str()));
890
                                      QCoreApplication::processEvents();
891
                              frame->Release();
ss.str("");
892
893
                              ss << "Samples found = " << number_samples;
894
                              commObj.progressUpdate(number_samples * 100 / imagesToSample);
895
896
897
                       Sleep(2);
898
               }
899
900
               std::vector<std::vector<Point3f> > objectPoints(1);
               calcBoardCornerPositions(boardSize, squareSize, objectPoints[0]);
901
902
               objectPoints.resize(imagePoints.size(), objectPoints[0]);
903
904
               double rms = calibrateCamera(objectPoints, imagePoints, imageSize,
           cameraMatrix, distCoeffs, Rvec, Tvec);
905
               commObj.progressUpdate(0);
906
               //! Release camera ==--
907
               camera->Release();
908
               //! Save the obtained calibration coefficients in a file for later use
QString fileName = QFileDialog::getSaveFileName(nullptr, "Save calibration file", "", "Calibration File
909
910
              (*.xml);; All Files (*)");
911
               FileStorage fs(fileName.toUtf8().constData(), FileStorage::WRITE);
912
               fs << "CameraMatrix" << cameraMatrix;</pre>
913
               fs << "DistCoeff" << distCoeffs;</pre>
914
               fs << "RMS" << rms;
               strBuf = fs.releaseAndGetString();
915
               commObj.changeStatus(QString::fromStdString(strBuf));
916
               commObj.addLog("Saved calibration!");
917
918
               return 0;
919 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.3 calibrateGround()

int calibrateGround ()

Get the pose of the camera w.r.t the ground calibration frame. This frame sets the navigation frame for later results. The pose is averaged over 200 samples and then saved in the file referenceData.xml. This routine is basically the same as setReference.

Definition at line 1563 of file main.cpp.

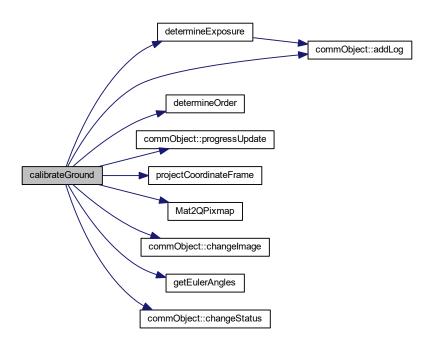
```
1564 {
          //! initialize the variables with starting values
1565
1566
         gotOrder = false;
         posRef = 0:
1567
1568
         eulerRef = 0:
         RmatRef = 0;
1569
1570
         Rvec = RvecOriginal;
1571
         Tvec = TvecOriginal;
1572
1573
         determineExposure();
1574
1575
         ss.str("");
1576
         commObj.addLog("Started ground calibration");
1577
1578
         CameraLibrary_EnableDevelopment();
1579
          //! Initialize Camera SDK =
1580
         CameraLibrary::CameraManager::X();
1581
1582
         //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
1583
         //! them on it's own.
1584
1585
          //! Get a connected camera =========
1586
         CameraManager::X().WaitForInitialization();
1587
         Camera *camera = CameraManager::X().GetCamera();
1588
1589
         //! If no device connected, pop a message box and exit ==--
1590
         if (camera == 0)
1591
1592
              commObj.addLog("No camera found!");
1593
              return 1;
1594
1595
1596
         //! Determine camera resolution to size application window ==----
         int cameraWidth = camera->Width();
int cameraHeight = camera->Height();
1597
1598
         camera->GetDistortionModel(distModel);
1599
1600
         cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
1601
1602
          //! Set camera mode to precision mode, it directly provides marker coordinates
1603
         camera->SetVideoType(Core::PrecisionMode);
1604
1605
          //! Start camera output ==--
1606
         camera->Start();
1607
          /! Turn on some overlay text so it's clear things are
1608
1609
         //! working even if there is nothing in the camera's view. ===---
1610
          //! Set some other parameters as well of the camera
1611
         camera->SetTextOverlay(true);
         camera->SetFrameRate(intFrameRate);
1612
1613
         camera->SetIntensity(intIntensity);
1614
         camera->SetIRFilter(true);
1615
         camera->SetContinuousIR(false);
1616
         camera->SetHighPowerMode(false);
1617
         //! sample some frames and calculate the position and attitude. then average those values and use that
1618
       as zero position
1619
        int numberSamples = 0;
1620
          int numberToSample = 200;
1621
         double projectionError = 0;
1622
1623
         while (numberSamples < numberToSample)</pre>
1624
         {
1625
              //! Fetch a new frame from the camera ===---
             Frame *frame = camera->GetFrame();
1626
1627
1628
              if (frame)
1629
                  //! Ok, we've received a new frame, lets do something
1630
1631
                  //! with it.
1632
                  if (frame->ObjectCount() == numberMarkers)
1633
1634
                      //!for(int i=0; i<frame->ObjectCount(); i++)
1635
                      for (int i = 0; i < numberMarkers; i++)</pre>
1636
                          cObject *obj = frame->Object(i);
1637
                          list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
1638
1639
1640
1641
                      if (gotOrder == false)
1642
1643
                          determineOrder();
1644
1645
1646
                      //! sort the 2d points with the correct indices as found in the preceeding order
       {\tt determination}\ {\tt algorithm}
1647
                      for (int w = 0; w < numberMarkers; w++)</pre>
1648
```

```
1649
                          list_points2d[w] = list_points2dUnsorted[
      pointOrderIndices[w]];
1650
1651
                      list_points2dOld = list_points2dUnsorted;
1652
1653
                      //!Compute the pose from the 3D-2D corresponses
                      solvePnP(list_points3d, list_points2d,
1654
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
1655
                      //! project the marker 3d points with the solution into the camera image CoSy and calculate
1656
       difference to true camera image
1657
                      projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2dProjected);
1658
                      projectionError = norm(list_points2dProjected,
      list_points2d);
1659
1660
                      if (projectionError > 3)
1661
1662
                          commObj.addLog("Reprojection error is bigger than 3 pixel. Correct marker
       configuration loaded?\nMarker position measured precisely?");
1663
                          frame->Release();
1664
                          return 1;
1665
1666
1667
                      double maxValue = 0;
1668
                      double minValue = 0;
1669
                      minMaxLoc(Tvec.at<double>(2), &minValue, &maxValue);
1670
1671
                      if (maxValue > 10000 || minValue < 0)</pre>
1672
1673
1674
1675
                          commObj.addLog("Negative z distance, thats not possible. Start the set
       zero routine again and check marker configurations.");
1676
                          frame->Release();
1677
                          return 1;
1678
1679
                      if (norm(positionOld) - norm(Tvec) < 0.05) //!<Iterative Method needs time</pre>
1680
       to converge to solution
1681
                      {
                          add(posRef, Tvec, posRef);
1682
                          add(eulerRef, Rvec, eulerRef); //! That are not the values of yaw,
1683
       roll and pitch yet! Rodriguez has to be called first.

numberSamples++; //!<-- one sample more :D
1684
1685
                          commObj.progressUpdate(numberSamples * 100 / numberToSample);
1686
                      positionOld = Tvec;
1687
1688
1689
                      Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
1690
                      for (int i = 0; i < numberMarkers; i++)</pre>
1691
1692
                          circle(cFrame, Point(list_points2d[i].x,
      list_points2d[i].y), 6, Scalar(0, 225, 0), 3);
1693
1694
                      projectCoordinateFrame(cFrame);
1695
                      projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2d);
1696
                      for (int i = 0; i < numberMarkers; i++)</pre>
1697
                          circle(cFrame, Point(list_points2d[i].x,
1698
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
1699
1700
1701
                      QPixmap QPFrame;
1702
                      QPFrame = Mat2QPixmap(cFrame);
                      commObj.changeImage(QPFrame);
1703
1704
                      OCoreApplication::processEvents();
1705
1706
1707
                  frame->Release();
1708
             }
1709
         //! Release camera ==--
1710
1711
         camera->Release();
1712
1713
          \ensuremath{//!}\ensuremath{\text{Divide}} by the number of samples to get the mean of the reference position
1714
         divide(posRef, numberToSample, posRef);
         divide(eulerRef, numberToSample, eulerRef); //!< eulerRef is here in Axis Angle
1715
       notation
1716
1717
         Rodrigues(eulerRef, RmatRef);
                                                         //!< axis angle to rotation matrix
1718
         getEulerAngles(RmatRef, eulerRef); //!< rotation matrix to euler
ss.str("");</pre>
1719
1720
1721
         ss << "RmatRef is:\n";
```

```
ss << RmatRef << "\n";
              ss << "Reference Position is:\n";
ss << posRef << "[mm] \n";
ss << "Reference Euler angles are:\n";
ss << eulerRef << "[deg] \n";
1723
1724
1725
1726
1727
         //! Save the obtained calibration coefficients in a file for later use
  QString fileName = QFileDialog::getSaveFileName(nullptr, "Save ground calibration file", "
referenceData.xml", "Calibration File (*.xml);; All Files (*)");
1728
1729
              FileStorage fs(fileName.toUtf8().constData(), FileStorage::WRITE);
fs << "M_NC" << RmatRef;
fs << "eulerRef" << eulerRef;
1730
1731
1732
              strBuf = fs.releaseAndGetString();
1733
1734
              commObj.changeStatus(QString::fromStdString(strBuf));
1735
              commObj.addLog("Saved ground calibration!");
1736
1737
              commObj.progressUpdate(0);
              return 0;
1738 }
```

Here is the call graph for this function:





6.5.2.4 closeUDP()

```
void closeUDP ( )
```

Close the UDP ports again to release network interfaces etc. If this is not done the network resources are still occupied and the program can't exit properly.

Definition at line 1173 of file main.cpp.

```
1174 {
         //! check if the socket is open and if yes close it
if (udpSocketObject->isOpen())
1175
1176
1177
         -{
              udpSocketObject->close();
1179
1180
1181
         if (udpSocketSafety->isOpen())
1182
              udpSocketSafety->close();
1183
1184
1185
1186
         if (udpSocketSafety2->isOpen())
1187
1188
              udpSocketSafety2->close();
1189
1190
          commObj.addLog("Closed all UDP ports.");
1191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.5 determineExposure()

```
int determineExposure ( )
```

Get the optimal exposure for the camera. For that find the minimum and maximum exposure were the right number of markers are detected. Then the mean of those two values is used as exposure.

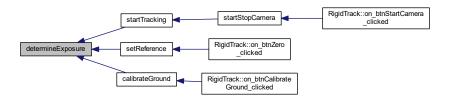
Definition at line 1362 of file main.cpp.

```
1363 {
          //! For OptiTrack Ethernet cameras, it's important to enable development mode if you
1364
1365
          //! want to stop execution for an extended time while debugging without disconnecting
1366
          //! the Ethernet devices. Lets do that now:
1367
1368
          CameraLibrary EnableDevelopment():
1369
1370
          //! Initialize Camera SDK ==--
1371
          CameraLibrary::CameraManager::X();
1372
1373
          //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
1374
          //! them on it's own.
1375
1376
          //! Get a connected camera =========
1377
          CameraManager::X().WaitForInitialization();
1378
          Camera *camera = CameraManager::X().GetCamera();
1379
1380
          //! If no device connected, pop a message box and exit ==--
1381
          if (camera == 0)
1382
          {
              commObj.addLog("No camera found!");
1383
1384
              return 1;
1385
          }
1386
1387
          //! Determine camera resolution to size application window ==----
1388
          int cameraWidth = camera->Width();
1389
          int cameraHeight = camera->Height();
1390
          1391
       {\tt greyscale} \ {\tt imformation} \ {\tt for} \ {\tt marker} \ {\tt property} \ {\tt calculations}
1392
1393
                                                          //! Start camera output ==
1394
1395
          //! Turn on some overlay text so it's clear things are ===---/! working even if there is nothing in the camera's view. ===---
1396
1397
1398
          camera->SetTextOverlay(true);
1399
          camera->SetExposure(intExposure);
                                                   //! set the camera exposure
          camera->SetIntensity(intIntensity); //! set the camera infrared LED intensity camera->SetFrameRate(intFrameRate); //! set the camera framerate to 100 Hz
1400
1401
1402
          camera->SetIRFilter(true); //! enable the filter that blocks visible light and only passes infrared
       light
         camera->SetHighPowerMode(true); //! enable high power mode of the leds
camera->SetContinuousIR(false); //! enable continuous LED light
1403
1404
          camera->SetThreshold(intThreshold); //! set threshold for marker detection
1405
1406
1407
          //!set exposure such that num markers are visible
1408
         int numberObjects = 0; //! Number of objects (markers) found in the current picture with the given
       exposure
          int minExposure = 1; //! exposure when objects detected the first time is numberMarkers int maxExposure = 480; //! exposure when objects detected is first time numberMarkers+1
1409
         int minExposure = 1;
1410
          intExposure = minExposure; //! set the exposure to the smallest value possible
1411
                                   //! if the markers arent found after numberTries then there might be no markers
1412
          int numberTries = 0;
       at all in the real world
1413
1414
                                    //! Determine minimum exposure, hence when are numberMarkers objects detected
          camera->SetExposure(intExposure);
1415
1416
          while (numberObjects != numberMarkers && numberTries < 48)</pre>
1417
              //! get a new camera frame
Frame *frame = camera->GetFrame();
1418
1419
              if (frame) //! frame received
1420
1421
                   numberObjects = frame->ObjectCount();  //! how many objects are detected in the image
1422
1423
                   if (numberObjects == numberMarkers) { minExposure =
      intExposure; frame->Release(); break; } //! if the right amount if markers is found, exit while
1424
                   //! not the right amount of markers was found so increase the exposure and try again
1425
                   numberTries++;
1426
                   intExposure += 10;
1427
                   camera->SetExposure(intExposure);
                   ss.str("");
1428
                  ss << "Exposure: " << intExposure << "\t";
ss << "Objects found: " << numberObjects;</pre>
1429
1430
1431
                   commObj.addLog(OString::fromStdString(ss.str()));
1432
                   frame->Release();
1433
1434
1435
          //! Now determine maximum exposure, hence when are numberMarkers+1 objects detected
1436
          numberTries = 0;
                              //! if the markers arent found after numberTries then there might be no markers at
1437
       all in the real world
1438
          intExposure = maxExposure;
1439
          camera->SetExposure(intExposure);
1440
          numberObjects = 0;
          while (numberObjects != numberMarkers && numberTries < 48)</pre>
1441
1442
```

```
1443
              Frame *frame = camera->GetFrame();
1444
1445
                   numberObjects = frame->ObjectCount(); //! how many objects are detected in the image
1446
      if (numberObjects == numberMarkers) { maxExposure =
intExposure; frame->Release(); break; } //! if the right amount if markers is found, exit while
1447
1448
1449
                   //! not the right amount of markers was found so decrease the exposure and try again
1450
                   intExposure -= 10;
1451
                   numberTries++;
                   camera->SetExposure(intExposure);
1452
                   ss.str("");
1453
                   ss.del( ),
ss << "Exposure: " << intExposure << "\t";
ss << "Objects found: " << numberObjects;</pre>
1454
1455
1456
                   commObj.addLog(QString::fromStdString(ss.str()));
1457
                   frame->Release();
1458
              }
1459
         }
1460
1461
          //! set the exposure to the mean of min and max exposure determined
1462
          camera->SetExposure((minExposure + maxExposure) / 2.0);
1463
1464
          //! and now check if the correct amount of markers is detected with that new value
1465
          while (1)
1466
         {
1467
              Frame *frame = camera->GetFrame();
1468
              if (frame)
1469
              {
                   numberObjects = frame->ObjectCount(); //! how many objects are detected in the image
if (numberObjects != numberMarkers) //! are all markers and not more or less
1470
1471
       detected in the image
1472
1473
                        frame->Release();
1474
                        commObj.addLog("Was not able to detect the right amount of markers.");
1475
                        //! Release camera ==-
1476
                       camera->Release();
1477
                        return 1;
1478
1479
                   else //! all markers and not more or less are found
1480
1481
                        frame->Release();
                        intExposure = (minExposure + maxExposure) / 2.0;
1482
                       commObj.addLog("Exposure set to:");
1483
1484
1485
                        commObj.addLog(QString::number(intExposure));
1486
                       break;
1487
                   }
              }
1488
1489
         }
1490
1491
          camera->Release();
1492
          return 0;
1493
1494 }
```



Here is the caller graph for this function:



6.5.2.6 determineOrder()

```
void determineOrder ( )
```

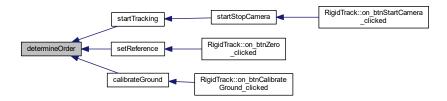
Compute the order of the marker points in 2D so they are the same as in the 3D array. Hence marker 1 must be in first place for both, list_points2d and list_points3d.

Definition at line 1498 of file main.cpp.

```
1499 {
1500
         //! determine the 3D-2D correspondences that are crucial for the PnP algorithm
1501
         //! Try every possible correspondence and solve PnP
1502
         //! Then project the 3D marker points into the 2D camera image and check the difference
1503
         //! between projected points and points as seen by the camera
1504
         //! the corresponce with the smallest difference is probably the correct one
1505
1506
             //! the difference between true 2D points and projected points is super big
1507
         minPointDistance = 5000;
1508
         std::sort(pointOrderIndices, pointOrderIndices + 4);
1509
1510
         //! now try every possible permutation of correspondence
1511
             //! reset the starting values for solvePnP
1512
             Rvec = RvecOriginal;
1513
1514
             Tvec = TvecOriginal;
1515
1516
             //! sort the 2d points with the current permutation
1517
             for (int m = 0; m < numberMarkers; m++)</pre>
1518
1519
                 list_points2d[m] = list_points2dUnsorted[
     pointOrderIndices[m]];
1520
             }
1521
1522
             //! Call solve PNP with P3P since its more robust and sufficient for start value determination
1523
             solvePnP(list_points3d, list_points2d,
     cameraMatrix, distCoeffs, Rvec, Tvec, useGuess, SOLVEPNP P3P):
1524
1525
             //! set the current difference of all point correspondences to zero
1526
             currentPointDistance = 0;
1527
1528
             //! project the 3D points with the solvePnP solution onto 2D \,
1529
             projectPoints(list_points3d, Rvec, Tvec,
     cameraMatrix, distCoeffs, list_points2dProjected);
1530
1531
             //! now compute the absolute difference (error)
1532
             for (int n = 0; n < numberMarkers; n++)</pre>
1533
             {
                 currentPointDistance += norm(list_points2d[n] -
1534
     list_points2dProjected[n]);
1535
1536
1537
             //! if the difference with the current permutation is smaller than the smallest value till now
1538
             //! it is probably the more correct permutation
1539
             if (currentPointDistance < minPointDistance)</pre>
1540
1541
                 minPointDistance = currentPointDistance;
                                                              //! < set the
```

```
smallest value of difference to the current one
1542
                  for (int b = 0; b < numberMarkers; b++)</pre>
                                                                 //!< now safe the better permutation
1543
                       pointOrderIndicesNew[b] = pointOrderIndices[b];
1544
1545
1546
              }
1547
1548
1549
         //! try every permutation
while (std::next_permutation(pointOrderIndices,
1550
1551
      pointOrderIndices + 4));
1552
1553
          //! now that the correct order is found assign it to the indices array
1554
          for (int w = 0; w < numberMarkers; w++)</pre>
1555
1556
              pointOrderIndices[w] = pointOrderIndicesNew[w];
1557
1558
         gotOrder = true;
1559 }
```

Here is the caller graph for this function:



6.5.2.7 drawPositionText()

Draw the position, attitude and reprojection error in the picture.

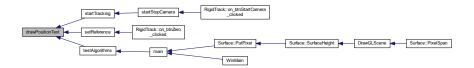
Parameters

in	Picture	is the camera image in OpenCV matrix format.	
in	Position	is the position of the tracked object in navigation CoSy.	
in	Euler	are the Euler angles with respect to the navigation frame.	
in	error	is the reprojection error of the pose estimation.	

Definition at line 1315 of file main.cpp.

```
1320
          ss.str("");
ss << "Y: " << Position[1] << " m";</pre>
1321
1322
          putText(Picture, ss.str(), cv::Point(200, 455), 1, 1, cv::Scalar(255, 255, 255));
1323
1324
1325
         ss.str("");
ss << "Z: " << Position[2] << " m";</pre>
1326
1327
          putText(Picture, ss.str(), cv::Point(200, 470), 1, 1, cv::Scalar(255, 255, 255));
1328
1329
         ss << "Heading: " << Euler[2] << " deg";
putText(Picture, ss.str(), cv::Point(350, 440), 1, 1, cv::Scalar(255, 255, 255));
1330
1331
1332
1333
          ss << "Pitch: " << Euler[1] << " deg";
1334
1335
          putText(Picture, ss.str(), cv::Point(350, 455), 1, 1, cv::Scalar(255, 255, 255));
1336
1337
          ss.str("");
1338
          ss << "Roll: " << Euler[0] << " deg";
1339
         putText(Picture, ss.str(), cv::Point(350, 470), 1, 1, cv::Scalar(255, 255, 255));
1340
1341
          ss << "Error: " << error << " px";
1342
          putText(Picture, ss.str(), cv::Point(10, 470), 1, 1, cv::Scalar(255, 255, 255));
1343
1344 }
```

Here is the caller graph for this function:



6.5.2.8 getEulerAngles()

Get the euler angles from a rotation matrix

Parameters

in	rotCamerMatrix	is a projection matrix, here normally only the extrinsic values.
out	eulerAngles	contains the Euler angles that result in the same rotation matrix as rotCamerMatrix.

Definition at line 241 of file main.cpp.

```
241
242
243
         Mat cameraMatrix, rotMatrix, transVect, rotMatrixX, rotMatrixY, rotMatrixZ;
244
         double* _r = rotCamerMatrix.ptr<double>();
         double projMatrix[12] = { _r[0],_r[1],_r[2],0,
    _r[3],_r[4],_r[5],0,
245
246
2.47
             _r[6],_r[7],_r[8],0 };
248
249
         decomposeProjectionMatrix(Mat(3, 4, CV_64FC1, projMatrix),
250
             cameraMatrix,
```

```
251 rotMatrix,

252 transVect,

253 rotMatrixX,

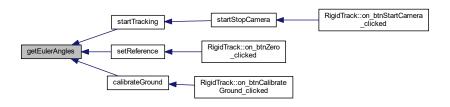
254 rotMatrixY,

255 rotMatrixZ,

256 eulerAngles);

257 }
```

Here is the caller graph for this function:



6.5.2.9 loadCalibration()

Load a previously saved camera calibration from a file.

Parameters

in method whether or not load the camera calibration from calibration.xml. If ==0 then yes, if != 0 then let the user select a different file.

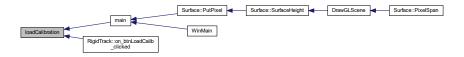
Definition at line 923 of file main.cpp.

```
923
                                                                                                                                                                                        {
924
925
                                      QString fileName;
926
                                       if (method == 0)
927
928
                                                           fileName = "calibration.xml";
929
930
                                      else
931
                                                          \verb|fileName| = QFileDialog::getOpenFileName(nullptr, "Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", "", " | Choose a previous saved calibration file", " | Choose a
932
                             Calibration Files (*.xml);;All Files (*)");
933
                                                         if (fileName.length() == 0)
934
935
                                                                              fileName = "calibration.xml";
936
937
                                      FileStorage fs;
938
                                     fs.open(fileName.toUtf8().constData(), FileStorage::READ);
fs["CameraMatrix"] >> cameraMatrix;
fs["DistCoeff"] >> distCoeffs;
commObj.addLog("Loaded calibration from file:");
939
940
941
942
                                     commobj.addLog(fileName);
ss.str("");
ss << "\nCamera Matrix is" << "\n" << cameraMatrix << "\n";
ss << "\nDistortion Coefficients are" << "\n" << distCoeffs << "\n";</pre>
943
944
945
946
947
                                      commObj.addLog(QString::fromStdString(ss.str()));
948 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.10 loadCameraPosition()

```
void loadCameraPosition ( )
```

Load the rotation matrix from camera CoSy to ground CoSy It is determined during calibrateGround() and stays the same once the camera is mounted and fixed.

Definition at line 1348 of file main.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.11 loadMarkerConfig()

Load a marker configuration from file. This file has to be created by hand, use the standard marker configuration file as template.

Parameters

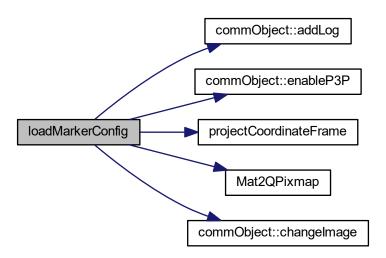
in method whether or not load the configuration from the markerStandard.xml. If ==0 load it, if != 0 let the user select a different file.

Definition at line 1195 of file main.cpp.

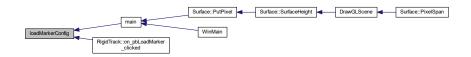
```
1196 {
1197
         OString fileName:
1198
         //! during start up of the programm load the standard marker configuration
1199
           (method == 0)
1200
1201
             //! open the standard marker configuration file
1202
             FileStorage fs;
             fs.open("markerStandard.xml", FileStorage::READ);
1203
1204
1205
             //! copy the values to the respective variables
1206
             fs["numberMarkers"] >> numberMarkers;
1207
1208
             //! inizialise vectors with correct length depending on the number of markers
1209
             list_points3d = std::vector<Point3d>(numberMarkers);
1210
             list_points2d = std::vector<Point2d>(numberMarkers);
1211
             list_points2dOld = std::vector<Point2d>(numberMarkers);
             list_points2dDifference = std::vector<double>(
1212
      numberMarkers);
1213
             list_points2dProjected = std::vector<Point2d>(
      numberMarkers):
1214
             list_points2dUnsorted = std::vector<Point2d>(
      numberMarkers);
1215
1216
             //! save the marker locations in the points3d vector
1217
             fs["list_points3d"] >> list_points3d;
1218
             fs.release();
             commObj.addLog("Loaded marker configuration from file:");
1219
1220
             commObj.addLog(fileName);
1221
1222
1223
1224
         else
1225
1226
1227
             //! if the load marker configuration button was clicked show a open file dialog
1228
             fileName = QFileDialog::getOpenFileName(nullptr, "Choose a previous saved marker configuration file
      ", "", "marker configuratio files (*.xml);;All Files (*)");
1229
1230
             //! was cancel or abort clicked
1231
             if (fileName.length() == 0)
1232
```

```
//! if yes load the standard marker configuration
1234
                  fileName = "markerStandard.xml";
1235
1236
1237
              //! open the selected marker configuration file
1238
              FileStorage fs;
1239
              fs.open(fileName.toUtf8().constData(), FileStorage::READ);
1240
1241
              //! copy the values to the respective variables
1242
              fs["numberMarkers"] >> numberMarkers;
1243
              //! inizialise vectors with correct length depending on the number of markers
1244
              list_points3d = std::vector<Point3d>(numberMarkers);
list_points2d = std::vector<Point2d>(numberMarkers);
1245
1246
1247
              list_points2dOld = std::vector<Point2d>(numberMarkers);
              list_points2dDifference = std::vector<double>(numberMarkers);
list_points2dProjected = std::vector<Point2d>(numberMarkers);
1248
1249
              list_points2dUnsorted = std::vector<Point2d>(numberMarkers);
1250
1251
1252
              //! save the marker locations in the points3d vector
1253
              fs["list_points3d"] >> list_points3d;
1254
              fs.release();
              commObj.addLog("Loaded marker configuration from file:");
1255
              commObj.addLog(fileName);
1256
1257
1258
1259
1260
          //! Print out the number of markers and their position to the GUI
1261
         ss.str("");
         ss << "Number of Markers: " << numberMarkers << "\n"; ss << "Marker 3D Points X,Y and Z [mm]: \n";
1262
1263
1264
          for (int i = 0; i < numberMarkers; i++)</pre>
1265
1266
              list_points3d[i].z << "\n";
1267
1268
         commObj.addLog(OString::fromStdString(ss.str()));
1269
1270
          //! check if P3P algorithm can be enabled, it needs exactly 4 marker points to work
1271
          if (numberMarkers == 4)
1272
1273
              //! if P3P is possible, let the user choose which algorithm he wants but keep iterative active
              methodPNP = 0:
1274
1275
              commObj.enableP3P(true);
1276
1277
         else
1278
1279
              //! More (or less) marker than 4 loaded, P3P is not possible, hence user cant select P3P in GUI
              methodPNP = 0;
1280
1281
              commObj.enableP3P(false);
1282
              commObj.addLog("P3P algorithm disabled, only works with 4 markers.");
1283
1284
         //! now display the marker configuration in the camera view Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
1285
1286
1287
1288
          //! Set the camera pose parallel to the marker coordinate system
1289
          Tvec.at<double>(0) = 0;
1290
         Tvec.at < double > (1) = 0;
1291
         Tvec.at<double>(2) = 4500;
         Rvec.at<double>(0) = 0 * 3.141592653589 / 180.0;
Rvec.at<double>(1) = 0 * 3.141592653589 / 180.0;
1292
1293
1294
         Rvec.at<double>(2) = -90. * 3.141592653589 / 180.0;
1295
1296
         projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
1297
         for (int i = 0; i < numberMarkers; i++)</pre>
1298
              circle(cFrame, Point(list_points2dProjected[i].x, list_points2dProjected[i].y), 3, Scalar(255, 0, 0
1299
      ), 3);
1300
1301
1302
          projectCoordinateFrame(cFrame);
         QPixmap QPFrame;
1303
1304
         QPFrame = Mat2QPixmap(cFrame);
1305
          commObj.changeImage(QPFrame);
1306
         QCoreApplication::processEvents();
1307
1308 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.12 main()

```
int main (
                int argc,
                 char * argv[] )
```

main initialises the GUI and values for the marker position etc

First the GUI is set up with Signals and Slots, see Qt docu for how that works. Then some variables are initialized with arbitrary values. At last calibration and marker configuration etc. are loaded from xml files.

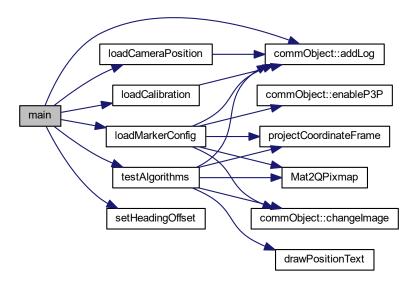
Parameters

in	argc	is not used.
in	argv	is also not used.

Definition at line 156 of file main.cpp.

```
157 {
158
         QApplication a(argc, argv);
         RigidTrack w;
w.show(); //!< show the GUI
159
160
         //! connect the Qt slots and signals for event handling
161
         QObject::connect(&commObj, SIGNAL(statusChanged(QString)), &w, SLOT(setStatus(QString)),
162
       Qt::DirectConnection);
163
         QObject::connect(&commObj, SIGNAL(imageChanged(QPixmap)), &w, SLOT(setImage(QPixmap)),
       Qt::DirectConnection);
164
         Object::connect(&commObj, SIGNAL(logAdded(OString)), &w, SLOT(setLog(OString)),
       Ot::DirectConnection);
         Object::connect(&commObj, SIGNAL(logCleared()), &w, SLOT(clearLog(OString)),
165
       Ot::DirectConnection);
         QObject::connect(&commObj, SIGNAL(P3Penabled(bool)), &w, SLOT(enableP3P(bool)),
166
       Qt::DirectConnection);
167
         QObject::connect(&commObj, SIGNAL(progressUpdated(int)), &w, SLOT(progressUpdate(int)),
       Ot::DirectConnection);
168
169
         commObj.addLog("RigidTrack Version:");
170
         commObj.addLog(QString::number(_MSC_FULL_VER));
171
         commObj.addLog("Built on:");
172
         commObj.addLog(QString(__DATE_
173
174
         //! initial guesses for position and rotation, important for Iterative Method!
175
         Tvec.at < double > (0) = 45;
176
         Tvec.at < double > (1) = 45;
177
         Tvec.at < double > (2) = 4500;
         Rvec.at<double>(0) = 0 * 3.141592653589 / 180.0;
Rvec.at<double>(1) = 0 * 3.141592653589 / 180.0;
178
179
         Rvec.at<double>(2) = -45 * 3.141592653589 / 180.0;
180
181
182
         //! Points that make up the marker CoSy axis system, hence one line in each axis direction
183
         coordinateFrame = std::vector<Point3d>(4);
184
         coordinateFrameProjected = std::vector<Point2d>(4);
         coordinateFrame[0] = cv::Point3d(0, 0, 0);
coordinateFrame[1] = cv::Point3d(300, 0, 0);
coordinateFrame[2] = cv::Point3d(0, 300, 0);
coordinateFrame[3] = cv::Point3d(0, 0, 300);
185
186
187
188
189
190
         position[0] = 1.1234;
                                        //! < \, {\rm set} \, \, {\rm position} \, \, {\rm initial} \, \, {\rm values}
191
         position[1] = 1.2345;
                                        //!< set position initial values
         position[2] = 1.3456;
                                        //! < \, {\rm set} \, {\rm position} \, {\rm initial} \, {\rm values}
192
193
194
         velocity[0] = 0.123;
                                      //!< set velocity initial values
         velocity[1] = 0.234;
                                      //!< set velocity initial values
195
196
         velocity[2] = 0.345;
                                      //!< set velocity initial values
197
198
         \frac{\texttt{eulerAngles}[0]}{\texttt{eulerAngles}[0]} = 1.002; \hspace{0.2cm} //! < \text{set initial euler angles to arbitrary values for testing}
         eulerAngles[1] = 1.003; //!< set initial euler angles to arbitrary values for testing eulerAngles[2] = 1.004; //!< set initial euler angles to arbitrary values for testing
199
200
201
202
         setHeadingOffset(0.0); //! < set the heading offset to 0
203
204
         ss.precision(4); //!< outputs in the log etc are limited to 3 decimal values
205
206
         loadCameraPosition(); //!< load the rotation matrix from camera CoSy to ground CoSy</pre>
         loadCalibration(0); //! < load the calibration file with the camera intrinsics
207
208
         loadMarkerConfig(0); //!< load the standard marker configuration</pre>
209
         testAlgorithms(); //!< test the algorithms and their accuracy
210
211
         return a.exec();
212 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.13 Mat2QPixmap()

Convert an opency matrix that represents a picture to a Qt Pixmap object for the GUI.

Parameters

```
in src is the camera image represented as OpenCV matrix.
```

Definition at line 216 of file main.cpp.

```
217 {
218 QImage dest((const uchar *)src.data, src.cols, src.rows, src.step, QImage::Format_RGB888);
```

```
dest.bits(); //! enforce deep copy, see documentation

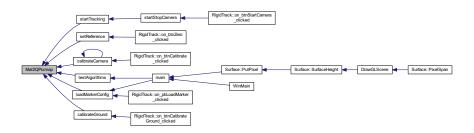
//! of QImage::QImage ( const uchar * data, int width, int height, Format format )

QPixmap pixmapDest = QPixmap::fromImage(dest);

return pixmapDest;

233 }
```

Here is the caller graph for this function:



6.5.2.14 projectCoordinateFrame()

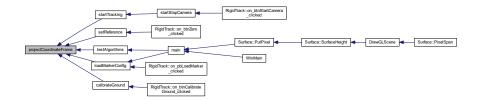
Project the coordinate CoSy origin and axis direction of the marker CoSy with the rotation and translation of the object for visualization.

Parameters

ir	pictureFrame	the image in which the CoSy frame should be pasted.	Ì
----	--------------	---	---

Definition at line 1081 of file main.cpp.

Here is the caller graph for this function:



6.5.2.15 sendDataUDP()

Send the position and attitude over UDP to every receiver, the safety receiver is handled on its own in the start

Tracking function because its send rate is less than 100 Hz.

Definition at line 1154 of file main.cpp.

```
1155 {
1156
           datagram.clear();
1157
           QDataStream out(&datagram, QIODevice::WriteOnly);
          out.setVersion(QDataStream::Qt_4_3);
out << (float)Position[0] << (float)Position[1] << (float)Position[2];
out << (float)Euler[0] << (float)Euler[1] << (float)Euler[2]; //! Roll Pitch Heading</pre>
1158
1159
1160
1161
           udpSocketObject->writeDatagram(datagram,
       IPAdressObject, portObject);
1162
           //! if second receiver is activated send it also the tracking data
1163
1164
          if (safety2Enable)
1165
          {
                udpSocketSafety2->writeDatagram(datagram,
       IPAdressSafety2, portSafety2);
1167
1168
1169 }
```

Here is the caller graph for this function:



6.5.2.16 setHeadingOffset()

```
void setHeadingOffset ( double d )
```

Add a heading offset to the attitude for the case it is wanted by the user.

Parameters

```
in d denotes heading offset in degrees.
```

Definition at line 1122 of file main.cpp.

```
//! Calculate rotation about x axis
           Mat R_x = (Mat_<double>(3, 3) << 1, 0, 0,
1128
1129
1130
                0, 1, 0,
1131
                0, 0, 1
1132
                );
1133
1134
           //! Calculate rotation about y axis
           Mat R_y = (Mat_<double>(3, 3) <<
    1, 0, 0,</pre>
1135
1136
1137
                0, 1, 0,
1138
                0, 0, 1
1139
                );
1140
1141
           //! Calculate rotation about z axis
           Mat R_z = (Mat_<double>(3, 3) <<
    cos(d), -sin(d), 0,
    sin(d), cos(d), 0,
    0, 0, 1);</pre>
1142
1143
1144
1145
1146
1147
1148
           //! Combined rotation matrix
1149
           M_HeadingOffset = R_z * R_y * R_x;
1150 }
```

Here is the caller graph for this function:



6.5.2.17 setReference()

```
int setReference ( )
```

Determine the initial position of the object that serves as reference point or as ground frame origin. Computes the pose 200 times and then averages it. The position and attitude are from now on used as navigation CoSy.

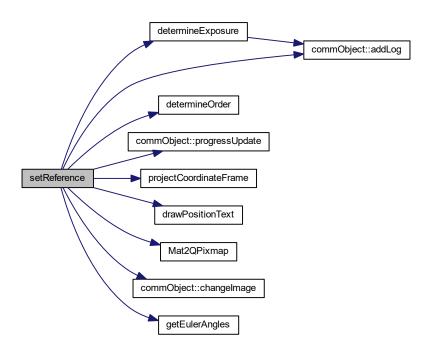
Definition at line 595 of file main.cpp.

```
596 {
        //! initialize the variables with starting values
597
598
       gotOrder = false;
       posRef = 0;
599
        eulerRef = 0;
600
601
        RmatRef = 0;
602
        Rvec = RvecOriginal;
       Tvec = TvecOriginal;
603
604
605
       determineExposure();
606
607
608
        commObj.addLog("Started reference coordinate determination.");
609
610
       CameraLibrary_EnableDevelopment();
611
        //! Initialize Camera SDK =
612
       CameraLibrary::CameraManager::X();
613
614
        //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
615
        //! them on it's own.
616
617
        //! Get a connected camera =========
618
       CameraManager::X().WaitForInitialization();
619
       Camera *camera = CameraManager::X().GetCamera();
```

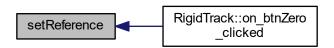
```
620
621
        //! If no device connected, pop a message box and exit ==--
622
        if (camera == 0)
623
        {
62.4
            commObj.addLog("No camera found!");
625
            return 1:
626
627
628
        //! Determine camera resolution to size application window ==----
629
        int cameraWidth = camera->Width();
        int cameraHeight = camera->Height();
630
631
        camera->GetDistortionModel(distModel);
        cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
632
633
634
        //! Set camera mode to precision mode, it directly provides marker coordinates
635
        camera->SetVideoType(Core::PrecisionMode);
636
637
        //! Start camera output ==--
638
        camera->Start();
639
640
        //! Turn on some overlay text so it's clear things are
641
        //! working even if there is nothing in the camera's view. ===---
642
        //! Set some other parameters as well of the camera
643
        camera->SetTextOverlay(true);
644
        camera->SetFrameRate(intFrameRate);
645
        camera->SetIntensity(intIntensity);
646
        camera->SetIRFilter(true);
647
        camera->SetContinuousIR(false);
648
        camera->SetHighPowerMode(false);
649
        //! sample some frames and calculate the position and attitude. then average those values and use that
650
       as zero position
651
       int numberSamples = 0;
652
        int numberToSample = 200;
653
        {\tt double\ projectionError\ =\ 0;\ //!<\ difference\ between\ the\ marker\ points\ as\ seen\ by\ the\ camera\ and\ the}
       projected marker points with Rvec and Tvec
654
655
        while (numberSamples < numberToSample)</pre>
656
657
            //! Fetch a new frame from the camera ===---
658
            Frame *frame = camera->GetFrame();
659
660
            if (frame)
661
662
                //! Ok, we've received a new frame, lets do something
663
                //! with it.
664
                if (frame->ObjectCount() == numberMarkers)
665
                     //!for(int i=0; i<frame->ObjectCount(); i++)
666
                     for (int i = 0; i < numberMarkers; i++)</pre>
667
668
669
                         cObject *obj = frame->Object(i);
670
                         list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
671
                     }
672
673
                     if (gotOrder == false)
674
675
                         determineOrder();
676
677
                    //! sort the 2d points with the correct indices as found in the preceeding order
678
       determination algorithm
679
                    for (int w = 0; w < numberMarkers; w++)</pre>
680
681
                         list_points2d[w] = list_points2dUnsorted[
      pointOrderIndices[w]];
682
                     list points2dOld = list points2dUnsorted;
683
684
685
                    //!Compute the pose from the 3D-2D corresponses
                    solvePnP(list_points3d, list_points2d,
686
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
687
                    //! project the marker 3d points with the solution into the camera image CoSy and calculate
688
       difference to true camera image
689
                    projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2dProjected);
690
                    projectionError = norm(list_points2dProjected,
      list points2d):
691
692
                    double maxValue = 0;
693
                    double minValue = 0;
694
                    minMaxLoc(Tvec.at<double>(2), &minValue, &maxValue);
695
696
                     if (maxValue > 10000 || minValue < 0)
697
```

```
ss.str("");
698
                         ss << "Negative z distance, thats not possible. Start the set zero routine again or
699
       restart Programm.";
700
                         commObj.addLog(QString::fromStdString(ss.str()));
                         frame->Release();
701
702
                         return 1:
703
                    }
704
705
                     if (projectionError > 3)
706
707
                         commObj.addLog("Reprojection error is bigger than 3 pixel. Correct marker
       configuration loaded?\nMarker position measured precisely?");
708
                         frame->Release();
709
710
711
                    if (norm(positionOld) - norm(Tvec) < 0.05) //!<Iterative Method needs time</pre>
712
       to converge to solution
713
                         add(posRef, Tvec, posRef);
715
                         add(eulerRef, Rvec, eulerRef); //!< That are not the values of yaw,
       roll and pitch yet! Rodriguez has to be called first.
716
                                           //!< one sample more :D
                         numberSamples++;
                         commObj.progressUpdate(numberSamples * 100 / numberToSample);
717
718
719
                    positionOld = Tvec;
720
                    Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
for (int i = 0; i < numberMarkers; i++)</pre>
721
722
723
                         circle(cFrame, Point(list_points2d[i].x,
724
      list_points2d[i].y), 6, Scalar(0, 225, 0), 3);
725
726
                    projectCoordinateFrame(cFrame);
727
                    projectPoints(list_points3d, Rvec, Tvec,
      728
729
730
                         circle(cFrame, Point(list_points2d[i].x,
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
731
                    drawPositionText(cFrame, position,
732
      eulerAngles, projectionError);
733
734
                    QPixmap QPFrame;
735
                    QPFrame = Mat2QPixmap(cFrame);
736
                     commObj.changeImage(QPFrame);
737
                    QCoreApplication::processEvents();
738
739
740
                frame->Release();
741
            }
742
743
        //! Release camera ==--
744
        camera->Release();
745
746
        //!Divide by the number of samples to get the mean of the reference position
747
        divide(posRef, numberToSample, posRef);
748
        divide(eulerRef, numberToSample, eulerRef); //!< eulerRef is here in Axis Angle
       notation
749
750
        Rodrigues(eulerRef, RmatRef);
                                                       //!< axis angle to rotation matrix
751
        //!-- Euler Angles, finally
752
        getEulerAngles(RmatRef, eulerRef); //!< rotation matrix to euler</pre>
753
        ss.str("");
        ss << "RmatRef is:\n";
ss << RmatRef << "\n";
754
755
        ss << "Reference Position is:\n";
756
        ss << posRef << "[mm] \n";
757
        ss << "Reference Euler Angles are:\n";
758
759
        ss << eulerRef << "[deg] \n";
760
761
        //! compute the difference between last obtained TVec and the average Value
        ^{\prime\prime}! When it is large the iterative method has not converged properly so it is advised to start the
762
       setReference() function once again
763
        double error = norm(posRef) - norm(Tvec);
764
        if (error > 5.0)
765
766
            ss << "Caution, distance between reference position and last position is: " << error << "\n Start
       the set zero routine once again.";
767
768
        commObj.addLog(QString::fromStdString(ss.str()));
769
        commObj.progressUpdate(0);
770
        return 0;
771 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.18 setUpUDP()

void setUpUDP ()

Open the UDP ports for communication.

Definition at line 1090 of file main.cpp.

```
1091 {
1092
          //! Initialise the QDataStream that stores the data to be send
1093
         QDataStream out(&datagram, QIODevice::WriteOnly);
1094
         out.setVersion(QDataStream::Qt_4_3);
1095
1096
         //! Create UDP slots
         commObj.addLog("Opening UDP ports.");
udpSocketObject = new QUdpSocket(0);
1097
1098
1099
         udpSocketObject->connectToHost(IPAdressObject,
      portObject);
1100
         commObj.addLog("Opened first receiver UDP port.");
1101
         udpSocketSafety = new QUdpSocket(0);
udpSocketSafety2 = new QUdpSocket(0);
1102
1103
1104
1105
          //! if the safety feature is activated open the udp port
1106
         if (safetyEnable)
1107
         {
1108
              udpSocketSafety->connectToHost(IPAdressSafety,
      portSafety);
1109
              commObj.addLog("Opened safety UDP port.");
1110
1111
         //! if the second receiver feature is activated open the udp port
1112
1113
         if (safety2Enable)
1114
1115
              udpSocketSafety2->connectToHost(IPAdressSafety2,
      portSafety2);
1116
             commObj.addLog("Opened second receiver UDP port.");
1117
1118 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.19 startStopCamera()

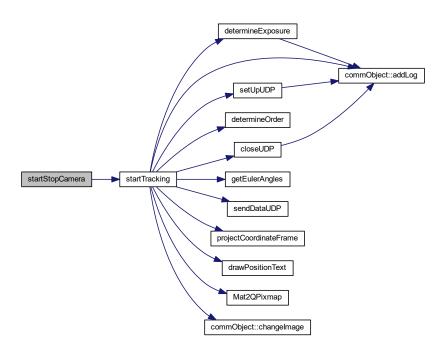
```
void startStopCamera ( )
```

Start or stop the tracking depending on if the camera is currently running or not.

Definition at line 579 of file main.cpp.

```
580 {
581
         //! tracking is not running so start it
582
        if (exitRequested)
583
            exitRequested = false;
startTracking();
584
585
586
587
        else //!< tracking is currently running, set exitRequest to true so the while loop in startTracking()</pre>
588
589
             exitRequested = true;
590
591 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.20 startTracking()

```
int startTracking ( )
```

Start the loop that fetches frames, computes the position etc and sends it to other computers. This function is the core of this program, hence the pose estimation is done here.

Definition at line 261 of file main.cpp.

```
261
                                          {
262
263
              gotOrder = false; //! The order of points, hence which entry in list_points3d corresponds to
264
            which in list_points2d is not calculated yet
265
              Rvec = RvecOriginal; //! Use the value of Rvec that was set in main() as starting value
            for the solvePnP algorithm
   Tvec = TvecOriginal; //! Use the value of Tvec that was set in main() as starting value
266
            for the solvePnP algorithm
267
              GetLocalTime(&logDate); //! Get the current date and time to name the log file
268
269
              //! Concat the log file name as followed. The file is saved in the folder /logs in the Rigid Track
            installation folder
          logFileName = "./logs/positionLog_" + QString::number(logDate.wDay) + "_" +
QString::number(logDate.wMonth) + "_" + QString::number(logDate.wYear);
logFileName += "_" + QString::number(logDate.wHour) + "_" + 
270
271
272
              logName = logFileName.toStdString(); //! Convert the QString to a standard string
273
274
              determineExposure(); //! Get the exposure where the right amount of markers is
            detected
275
276
                /! For OptiTrack Ethernet cameras, it's important to enable development mode if you
277
              //! want to stop execution for an extended time while debugging without disconnecting
278
              //! the Ethernet devices. Lets do that now:
279
280
              CameraLibrary_EnableDevelopment();
281
              CameraLibrary::CameraManager::X(); //! Initialize Camera SDK
282
283
              //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
284
              //! them on it's own
285
286
              //! Get a connected camera
287
              CameraManager::X().WaitForInitialization();
288
              Camera *camera = CameraManager::X().GetCamera();
289
290
              //! If no camera can be found, inform user in message log and exit function
291
              if (camera == 0)
292
              {
293
                     commObj.addLog("No camera found!");
294
                     return 1;
295
296
297
              //! Determine camera resolution to size application window
298
              int cameraWidth = camera->Width();
299
              int cameraHeight = camera->Height();
300
              camera->SetVideoType(Core::PrecisionMode); //! Set the camera mode to precision mode, it used
301
            greyscale imformation for marker property calculations
302
303
              camera->Start(); //! Start camera output
304
305
              //! Turn on some overlay text so it's clear things are
306
              //! working even if there is nothing in the camera's view
307
              camera->SetTextOverlay(true);
308
              camera->SetExposure(intExposure);
                                                                              //! Set the camera exposure
              camera->SetIntensity(intIntensity); //! Set the camera infrared LED intensity
camera->SetFrameRate(intFrameRate); //! Set the camera framerate to 100 Hz
309
310
              camera->SetIRFilter(true); //! Enable the filter that blocks visible light and only passes infrared
311
             light
312
              camera->SetHighPowerMode(true); //! Enable high power mode of the LEDs
camera->SetContinuousIR(false); //! Disable continuous LED light
313
314
              camera->SetThreshold(intThreshold); //! Set threshold for marker detection
315
316
              //! Create a new matrix that stores the grayscale picture from the camera
317
              Mat matFrame = Mat::zeros(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
318
              QPixmap QPFrame; //! QPixmap is the corresponding Qt class that saves images
319
              //! Matrix that stores the colored picture, hence marker points, coordinate frame and reprojected
            points
320
              Mat cFrame(480, 640, CV 8UC3, Scalar(0, 0, 0));
321
322
              int v = 0; //! Helper variable used to kick safety switch
```

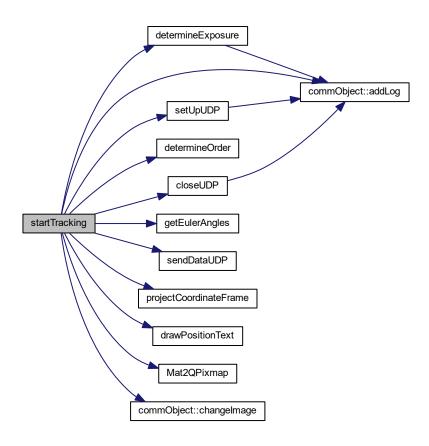
```
323
                  //! Variables for the min and max values that are needed for sanity checks
324
                  double maxValue = 0;
325
                  double minValue = 0;
                  int framesDropped = 0; //! Ff a marker is not visible or accuracy is bad increase this counter
326
327
                  double projectionError = 0; //! Equals the quality of the tracking
328
329
                  setUpUDP(); //! Open sockets and ports for UDP communication
330
331
                  if (safetyEnable) //! If the safety feature is enabled send the starting message
332
333
                           //! Send enable message, hence send a 9 and then a 1
334
                           data.setNum((int)(9));
335
                           udpSocketSafety->write(data);
336
                           data.setNum((int)(1));
337
                           udpSocketSafety->write(data);
338
339
340
                  //! Fetch a new frame from the camera
341
                 bool gotTime = false; //! Get the timestamp of the first frame. This time is subtracted from every
                subseeding frame so the time starts at 0 in the logs
342
                 while (!gotTime) //! While no new frame is received loop
343
344
                           Frame *frame = camera->GetFrame(); //! Get a new camera frame
                           if (frame) //! There is actually a new frame
345
346
                           {
347
                                    timeFirstFrame = frame->TimeStamp(); //! Get the time stamp for the first frame.
                It is subtracted for the following frames
3/18
                                    frame->Release(); //! Release the frame so the camera can continue
                                    gotTime = true; //! Exit the while loop
349
350
                           }
351
                  }
352
353
                  //! Now enter the main loop that processes each frame and computes the pose, sends it and logs stuff
354
                  while (!exitRequested) //! Check if the user has not pressed "Stop Tracking" yet
355
356
357
                          Frame *frame = camera->GetFrame(); //! Fetch a new frame from the camera
358
359
                           if (frame) //! Did we got a new frame or does the camera still need more time
360
361
                                    frames Dropped ++; \ //! \ Increase \ by \ one, \ if \ everything \ is \ okay \ it \ is \ decreased \ at \ the \ end \ of \ the
               loop again
362
363
                                     //! Only use this frame it the right number of markers is found in the picture
                                     if (frame->ObjectCount() == numberMarkers)
364
365
366
                                              //! Get the marker points in 2D in the camera image frame and store them in the
               {\tt list\_points2dUnsorted\ vector}
                                             //! The order of points that come from the camera corresponds to the Y coordinate
367
                                              for (int i = 0; i < numberMarkers; i++)</pre>
368
369
                                              {
370
                                                       cObject *obj = frame->Object(i);
371
                                                       list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
372
                                              }
373
374
                                             if (gotOrder == false) //! Was the order already determined? This is false for the
                first frame and from then on true
375
                                             {
376
                                                      determineOrder(); //! Now compute the order
377
                                             }
378
                                             //! Sort the 2d points with the correct indices as found in the preceeding order % \left( 1\right) =\left( 1\right) \left( 1\right) 
379
               determination algorithm
380
                                              for (int w = 0; w < numberMarkers; w++)</pre>
381
382
                                                      list_points2d[w] = list_points2dUnsorted[
              pointOrderIndices[w]]; //! pointOrderIndices was calculated in determineOrder()
383
                                             list_points2dOld = list_points2dUnsorted;
384
385
386
                                              //! The first time the 2D-3D corresspondence was determined with gotOrder was okay.
387
                                              //! But this order can change as the object moves and the marker objects appear in a
388
                                              //! different order in the frame->Object() array.
                                              //! The solution is that: When a marker point (in the camera image, hence in 2D) was at
389
                                              //! a position then it wont move that much from one frame to the other.
390
                                              //! So for the new frame we take a marker object and check which marker was closest this
391
               point
392
                                              //! in the old image frame? This is probably the same (true) marker. And we do that for
                every other marker as well.
393
                                              //! When tracking is good and no frames are dropped because of missing markers this should
                work every frame.
394
                                              for (int j = 0; j < numberMarkers; j++)</pre>
395
396
                                                      minPointDistance = 5000; //! The sum of point distances is set to
                something unrealistic large
                                                      for (int k = 0; k < numberMarkers; k++)</pre>
397
398
```

```
399
                           //! Calculate N_2 norm of unsorted points minus old points
                           currentPointDistance = norm(
400
      list_points2dUnsorted[pointOrderIndices[j]] -
      list_points2dOld[k]);
401
                           //! If the norm is smaller than minPointDistance the correspondence is more likely
      to be correct
402
                           if (currentPointDistance <</pre>
      minPointDistance)
403
404
                               //! Update the array that saves the new point order
405
                               minPointDistance =
      currentPointDistance:
406
                               pointOrderIndicesNew[i] = k;
407
408
409
                   }
410
411
                   //! Now the new order is found, set the point order to the new value
                   for (int k = 0; k < numberMarkers; k++)</pre>
412
413
                       pointOrderIndices[k] = pointOrderIndicesNew[k];
414
415
                       list_points2d[k] = list_points2dUnsorted[
      pointOrderIndices[k]];
416
                   }
417
418
                   //! Save the unsorted position of the marker points for the next loop
419
                   list_points2dOld = list_points2dUnsorted;
420
421
                   //!Compute the object pose from the 3D-2D corresponses
422
                   solvePnP(list_points3d, list_points2d,
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
423
424
                   //! Project the marker 3d points with the solution into the camera image CoSy and calculate
      difference to true camera image
                   projectPoints(list_points3d, Rvec, Tvec,
425
     cameraMatrix, distCoeffs, list_points2dProjected);
    projectionError = norm(list_points2dProjected,
426
      list_points2d); //! Difference of true pose and found pose
427
428
                   //! Increase the framesDropped variable if accuracy of tracking is too bad
429
                   if (projectionError > 5)
430
431
                       framesDropped++;
432
433
                   else
434
                       framesDropped = 0; //! Set number of subsequent frames dropped to zero because error
435
      is small enough and no marker was missing
436
437
438
                   //! Get the min and max values from TVec for sanity check
439
                   minMaxLoc(Tvec.at<double>(2), &minValue, &maxValue);
440
                   //! Sanity check of values. negative z means the marker CoSy is behind the camera, that's
441
      not possible.
442
                   if (minValue < 0)
443
                       commObj.addLog("Negative z distance, that is not possible. Start the set
444
       zero routine again or restart Program.");
frame->Release(); //! Release the frame so the camera can move on
445
                       camera->Release(); //! Release the camera
446
447
                       closeUDP(); //! Close all UDP connections so the programm can be closed later
      on and no resources are locked
448
                       return 1; //! Exit the function
449
450
                   //! Next step is the transformation from camera CoSy to navigation CoSy
451
                   //! Compute the relative object position from the reference position to the current one
452
                    //! given in the camera CoSy: f T_C^{NM} = Tvec - Tvec_{Ref} f
453
454
                   subtract(Tvec, posRef, position);
455
456
                   //! Transform the position from the camera CoSy to the navigation CoSy with INS alligned
      457
                   Mat V = 0.001 * M_HeadingOffset * M_CN.t() * (Mat)
458
      position;
                   459
460
      hence height above ground is considered
461
462
                    //! Realtive angle between reference orientation and current orientation
                   Rodrigues (Rvec, Rmat); //! Convert axis angle respresentation to ordinary rotation
463
       matrix
464
                   465
466
```

```
467
                   Rmat = RmatRef.t() *Rmat;
468
469
                    //! Euler Angles, finally
470
                   getEulerAngles(Rmat, eulerAngles); //! Get the euler angles
       from the rotation matrix
471
                   eulerAngles[2] += headingOffset; //! Add the heading offset to the
      heading angle
472
473
                    //! Compute the velocity with finite differences. Only use is the log file. It is done here
      because the more precise time stamp can be used
474
                   frameTime = frame->TimeStamp() - timeOld; //! Time between the old frame
       and the current frame
475
                   timeOld = frame->TimeStamp();
                                                    //! Set the old frame time to the current one
                    velocity[0] = (position[0] - positionOld[0]) /
476
      frameTime; //! Calculate the x velocity with finite differences
477
                    velocity[1] = (position[1] - positionOld[1]) /
      478
      frameTime; //! Calculate the z velocity with finite differences
479
                   positionOld = position; //! Set the old position to the current one for
      next frame velocity calcuation
480
481
                    //! Send position and Euler angles over WiFi with 100 Hz
482
                    sendDataUDP (position, eulerAngles);
483
484
                    //! Save the values in a log file, values are:
485
                    //! Time sinc tracking started Position
                                                              Euler Angles
                                                                              Velocity
486
                   logfile.open(logName, std::ios::app); //! Open the log file, the folder is
      RigidTrackInstallationFolder/logs
487
                   logfile << frame->TimeStamp() - timeFirstFrame << ";" <</pre>
     488
489
                    logfile << velocity[0] << ";" << velocity[1] << ";" <<
      velocity[2] << "\n";</pre>
                   logfile.close(); //! Close the file to save values
490
491
                }
492
493
                //! Check if the position and euler angles are below the allowed value, if yes send OKAY signal
       (1), if not send shutdown signal (0)
494
                //! Absolute x, y and z position in navigation CoSy must be smaller than the allowed distance
495
                if (safetyEnable)
496
                {
497
                    if ((abs(position[0]) < safetyBoxLength && abs(position[1]) <</pre>
      safetyBoxLength && abs(position[2]) < safetyBoxLength))</pre>
498
499
                        //! Absolute Euler angles must be smaller than allowed value. Heading is not considered
500
                       if ((abs(eulerAngles[0]) < safetyAngle && abs(eulerAngles[1]) <</pre>
      safetvAngle))
501
                       {
502
                            //! Send the OKAY signal to the desired computer every 5th time
                            if (v == 5) {
503
504
                               data.setNum((int)(1));
                               udpSocketSafety->write(data); //! Send the 1
505
506
                               v = 0; //! reset the counter that is needed for decimation to every 5th time
      step
507
508
509
                       //! The euler angles of the object exceeded the allowed euler angles, send the shutdown
       signal (0)
510
                       else
511
512
                           data.setNum((int)(0)); //! Send the shutdown signal, a 0
513
                           udpSocketSafety->write(data);
                           commObj.addLog("Object exceeded allowed Euler angles, shutdown signal
514
      sent."); //! Inform the user
515
516
517
518
                    //! The position of the object exceeded the allowed position, shut the object down
519
520
                       data.setNum((int)(0)); //! Send the shutdown signal, a 0
521
                       udpSocketSafety->write(data);
522
                       commObj.addLog("Object left allowed area, shutdown signal sent."); //!
       Inform the user
524
525
                   }
526
               1
527
528
                //! Inform the user if tracking system is disturbed (marker lost or so) or error was too big
                if (framesDropped > 10)
529
530
531
                    if (safetyEnable) //! Also send the shutdown signal
532
                       data.setNum((int)(0)); //! Send the shutdown signal, a 0
533
```

```
534
                         udpSocketSafety->write(data);
535
536
                     commObj.addLog("Lost marker points or precision was bad!"); //! Inform the
       user
537
                     framesDropped = 0;
538
                }
539
540
                 //! Rasterize the frame so it can be shown in the GUI
541
                frame->Rasterize(cameraWidth, cameraHeight, matFrame.step,
      BACKBUFFER_BITSPERPIXEL, matFrame.data);
542
                //! Convert the frame from greyscale as it comes from the camera to rgb color cvtColor(matFrame, cFrame, COLOR_GRAY2RGB);
543
544
545
546
                //! Project (draw) the marker CoSy origin into 2D and save it in the cFrame image
547
                projectCoordinateFrame(cFrame);
548
                //! Project the marker points from 3D to the camera image frame (2d) with the computed pose
549
                projectPoints(list_points3d, Rvec, Tvec,
550
      cameraMatrix, distCoeffs, list_points2d);
551
                for (int i = 0; i < numberMarkers; i++)</pre>
552
                     //! Draw a circle around the projected points so the result can be better compared to the
553
       real marker position
554
                     //! In the resulting picture those are the red dots
                     circle(cFrame, Point(list_points2d[i].x,
555
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
556
557
                //! Write the current position, attitude and error values as text in the frame
558
559
                drawPositionText(cFrame, position, eulerAngles, projectionError);
560
561
                 //! Send the new camera picture to the GUI and call the GUI processing routine
562
                QPixmap QPFrame;
                QPFrame = Mat2QPixmap(cFrame);
563
                 commObj.changeImage(QPFrame); //! Update the picture in the GUI
564
                QCoreApplication::processEvents(); //! Give Qt time to handle everything
565
566
567
                 //! Release the camera frame to fetch the new one
568
                frame->Release();
569
            }
570
        }
571
572
        //! User choose to stop the tracking, clean things up
573
        closeUDP(); //! Close the UDP connections so resources are deallocated
574
        camera->Release(); //! Release camera
575
        return 0:
576 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.2.21 testAlgorithms()

void testAlgorithms ()

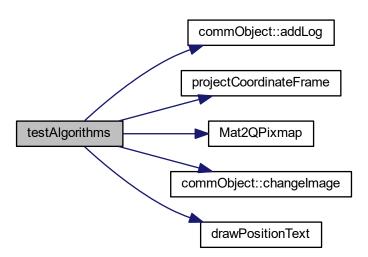
Project some points from 3D to 2D and then check the accuracy of the algorithms. Mainly to generate something that can be shown in the camera view so the user knows everything loaded correctly.

Definition at line 952 of file main.cpp.

```
953 {
954
955
        int _methodPNP;
956
957
        std::vector<Point2d> noise(numberMarkers);
958
959
        RvecOriginal = Rvec;
960
        TvecOriginal = Tvec;
961
962
        projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
963
964
        ss.str("");
965
        ss << "Unsorted Points 2D Projected \n";
966
        ss << list_points2dProjected << "\n";
967
        commObj.addLog(QString::fromStdString(ss.str()));
968
       Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
for (int i = 0; i < numberMarkers; i++)</pre>
969
970
971
972
            circle(cFrame, Point(list_points2dProjected[i].x, list_points2dProjected[i].y), 6, Scalar(0, 255, 0
      ), 3);
973
        }
974
975
        projectCoordinateFrame(cFrame);
976
977
        ss << "======\n";
978
        ss << "======= Projected Points =======\n";
979
        ss << list_points2dProjected << "\n";
980
981
982
        randn(noise, 0, 0.5);
983
        add(list_points2dProjected, noise, list_points2dProjected);
984
        985
986
987
        commObj.addLog(QString::fromStdString(ss.str()));
988
989
990
        bool useGuess = true;
        _methodPNP = 0; //!< 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //!< not used
991
992
        solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
993
      distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
994
        ss.str("");
995
        ss << "-----\n";
ss << "-----\n";
996
997
        ss << "rvec: " << "\n";
998
        ss << Rvec << "\n";
ss << "tvec: " << "\n";
999
1000
        ss << Tvec << "\n";
1001
1002
1003
        commObj.addLog(QString::fromStdString(ss.str()));
1004
1005
         _methodPNP = 1; //! < 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP UPnP not used
         Rvec = cv::Mat::zeros(3, 1, CV_64F);
Tvec = cv::Mat::zeros(3, 1, CV_64F);
1006
1007
1008
         solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
      distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1009
1010
        ss.str("");
1011
               "========\n";
        ss <<
        ss << "====== EPNP
1012
         ss << "rvec: " << "\n";
1013
        ss << Rvec << "\n";
ss << "tvec: " << "\n";
1014
1015
        ss << Tvec << "\n";
1016
1017
1018
        projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
1019
        for (int i = 0; i < numberMarkers; i++)</pre>
1020
              \mbox{circle(cFrame, Point(list\_points2dProjected[i].x, list\_points2dProjected[i].y), 3, Scalar(255, 0, 0) } \\
1021
     ), 3);
1022
1023
         QPixmap QPFrame;
1024
         QPFrame = Mat2QPixmap(cFrame);
1025
         commObj.changeImage(QPFrame);
1026
         OCoreApplication::processEvents();
1027
        commObj.addLog(QString::fromStdString(ss.str()));
1028
         if (numberMarkers == 4)
1029
1030
              methodPNP = 2; //! < 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //! < not used
             Rvec = cv::Mat::zeros(3, 1, CV_64F);
Tvec = cv::Mat::zeros(3, 1, CV_64F);
solvePnP(list_points3d, list_points2dProjected,
1031
1032
1033
```

```
cameraMatrix, distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1034
1035
            ss.str("");
            ss << "======\n";
1036
            ss << "=======\n";
1037
            ss << "rvec: " << "\n";
1038
           ss << Rvec << "\n";
ss << "tvec: " << "\n";
1039
1040
           ss << Tvec << "\n";
1041
1042
            projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
1043
     distCoeffs, list_points2dProjected);
       for (int i = 0; i < numberMarkers; i++)</pre>
1044
1045
1046
                circle(cFrame, Point(list_points2dProjected[i].x, list_points2dProjected[i].y), 3, Scalar(255,
     0, 0), 3);
1047
            double projectionError = norm(list_points2dProjected, list_points2d);
putText(cFrame, "Testing Algorithms Finished", cv::Point(5, 420), 1, 1, cv::Scalar(255, 255, 255));
1048
1049
            drawPositionText(cFrame, position, eulerAngles, projectionError)
1050
1051
1052
            QPixmap QPFrame;
1053
            OPFrame = Mat2OPixmap(cFrame);
1054
            commObj.changeImage(QPFrame);
1055
            QCoreApplication::processEvents();
1056
            commObj.addLog(QString::fromStdString(ss.str()));
1057
1058
1059
         _methodPNP = 4; //! < 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //! < not used
        Rvec = cv::Mat::zeros(3, 1, CV_64F);
1060
        Tvec = cv::Mat::zeros(3, 1, CV_64F);
solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
1061
     distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1063
1064
        ss << "-----\n";
1065
        ss << "========\\n";
1066
        ss << "rvec: " << "\n";
        ss << Rvec << "\n";
ss << "tvec: " << "\n";
1068
1069
        ss << Tvec << "\n";
1070
1071
1072
        commObj.addLog(QString::fromStdString(ss.str()));
1073
1074
        Rvec = RvecOriginal;
1075
        Tvec = TvecOriginal;
1076
1077 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.5.3 Variable Documentation

6.5.3.1 BACKBUFFER_BITSPERPIXEL

```
const int BACKBUFFER_BITSPERPIXEL = 8
```

8 bit per pixel and greyscale image from camera

Definition at line 140 of file main.cpp.

6.5.3.2 camera_started

```
bool camera_started = false
```

variable thats needed to exit the main while loop

Definition at line 122 of file main.cpp.

6.5.3.3 cameraMatrix

Mat cameraMatrix

camera matrix of the camera

Definition at line 124 of file main.cpp.

6.5.3.4 commObj

commObject commObj

class that handles the communication from main.cpp to the GUI

Now declare variables that are used across the main.cpp file. Basically almost every variable used is declared here.

Definition at line 68 of file main.cpp.

6.5.3.5 coordinateFrame

std::vector<Point3d> coordinateFrame

coordinate visualisazion of marker CoSy

Definition at line 113 of file main.cpp.

6.5.3.6 coordinateFrameProjected

std::vector<Point2d> coordinateFrameProjected

marker CoSy projected from 3D to 2D camera image CoSy

Definition at line 114 of file main.cpp.

6.5.3.7 currentMinIndex

```
int currentMinIndex = 0
```

helper variable set to the point order that holds the current minimum point distance

Definition at line 119 of file main.cpp.

6.5.3.8 currentPointDistance

```
double currentPointDistance = 5000
```

distance from the projected 3D points (hence in 2d) to the real 2d marker positions in camera image CoSy

Definition at line 117 of file main.cpp.

6.5.3.9 data

QByteArray data

data package that's sent to the safety receiver

Definition at line 138 of file main.cpp.

6.5.3.10 datagram

QByteArray datagram

data package that is sent to receiver 1 and 2

Definition at line 137 of file main.cpp.

6.5.3.11 distCoeffs

Mat distCoeffs

distortion coefficients of the camera

Definition at line 125 of file main.cpp.

6.5.3.12 distModel

Core::DistortionModel distModel

distortion model of the camera

Definition at line 126 of file main.cpp.

6.5.3.13 eulerAngles

Vec3d eulerAngles = Vec3d()

Roll Pitch Heading in this order, units in degrees.

Definition at line 82 of file main.cpp.

6.5.3.14 eulerRef

Vec3d eulerRef = Vec3d()

initial euler angle of object respectivley to camera CoSy

Definition at line 86 of file main.cpp.

6.5.3.15 exitRequested

```
bool exitRequested = true
```

variable if tracking loop should be exited

Definition at line 74 of file main.cpp.

6.5.3.16 frameTime

```
double frameTime = 0.01
```

100 Hz CoSy rate, is later on replaced with the hardware timestamp delivered by the camera

Definition at line 77 of file main.cpp.

6.5.3.17 gotOrder

```
bool gotOrder = false
```

order of the list_points3d and list_points3d already tetermined or not, has to be done once

Definition at line 120 of file main.cpp.

6.5.3.18 headingOffset

```
double headingOffset = 0
```

heading offset variable for aligning INS heading with tracking heading

Definition at line 87 of file main.cpp.

6.5.3.19 intExposure

```
int intExposure = 1
```

max is 480 increase if markers are badly visible but should be determined automatically during setReference()

Definition at line 90 of file main.cpp.

6.5.3.20 intFrameRate

```
int intFrameRate = 100
```

CoSy rate of camera, maximum is 100 fps.

Definition at line 91 of file main.cpp.

6.5.3.21 intIntensity

```
int intIntensity = 15
```

max infrared spot light intensity is 15 1-6 is strobe 7-15 is continuous 13 and 14 are meaningless

Definition at line 89 of file main.cpp.

6.5.3.22 intThreshold

```
int intThreshold = 200
```

threshold value for marker detection. If markers are badly visible lower this value but should not be necessary

Definition at line 92 of file main.cpp.

6.5.3.23 invertZ

```
int invertZ = 1
```

dummy variable to invert Z direction on request

Definition at line 75 of file main.cpp.

6.5.3.24 IPAdressObject

```
QHostAddress IPAdressObject = QHostAddress("127.0.0.1")
```

IPv4 adress of receiver 1.

Definition at line 131 of file main.cpp.

6.5.3.25 IPAdressSafety

```
QHostAddress IPAdressSafety = QHostAddress("192.168.4.1")
```

IPv4 adress of safety receiver.

Definition at line 132 of file main.cpp.

6.5.3.26 IPAdressSafety2

```
QHostAddress IPAdressSafety2 = QHostAddress("192.168.4.4")
```

IPv4 adress of receiver 2.

Definition at line 133 of file main.cpp.

6.5.3.27 list_points2d

```
std::vector<Point2d> list_points2d
```

marker positions projected in 2D in camera image CoSy

Definition at line 108 of file main.cpp.

6.5.3.28 list_points2dDifference

```
std::vector<double> list_points2dDifference
```

difference of the old and new 2D marker position to determine the order of the points

Definition at line 110 of file main.cpp.

6.5.3.29 list_points2dOld

```
std::vector<Point2d> list_points2d0ld
```

marker positions in previous picture in 2D in camera image CoSy

Definition at line 109 of file main.cpp.

6.5.3.30 list_points2dProjected

std::vector<Point2d> list_points2dProjected

3D marker points projected to 2D in camera image CoSy with the algorithm projectPoints

Definition at line 111 of file main.cpp.

6.5.3.31 list_points2dUnsorted

std::vector<Point2d> list_points2dUnsorted

marker points in 2D camera image CoSy, sorted with increasing x (camera image CoSy) but not sorted to correspond with list_points3d

Definition at line 112 of file main.cpp.

6.5.3.32 list_points3d

std::vector<Point3d> list_points3d

marker positions in marker CoSy

Definition at line 107 of file main.cpp.

6.5.3.33 logDate

SYSTEMTIME logDate

Systemtime struct that saves the current date and time thats needed for the log file name creation.

Definition at line 145 of file main.cpp.

6.5.3.34 logfile

std::ofstream logfile

file handler for writing the log file

Definition at line 146 of file main.cpp.

6.5.3.35 logFileName

QString logFileName

Filename for the logfiles.

Definition at line 143 of file main.cpp.

6.5.3.36 logName

```
std::string logName
```

Filename for the logfiles as standard string.

Definition at line 144 of file main.cpp.

6.5.3.37 M_CN

```
Mat M_CN = cv::Mat_<double>(3, 3)
```

rotation matrix from camera to ground, fixed for given camera position

Definition at line 97 of file main.cpp.

6.5.3.38 M_HeadingOffset

```
Mat M_HeadingOffset = cv::Mat_<double>(3, 3)
```

rotation matrix that turns the ground system to the INS magnetic heading for alignment

Definition at line 98 of file main.cpp.

6.5.3.39 methodPNP

```
int methodPNP = 0
```

solvePNP algorithm 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //! < 4 and 1 are the same and not implemented correctly by OpenCV

Definition at line 105 of file main.cpp.

6.5.3.40 minPointDistance

```
double minPointDistance = 5000
```

minimum distance from the projected 3D points (hence in 2d) to the real 2d marker positions in camera image CoSy Definition at line 118 of file main.cpp.

6.5.3.41 numberMarkers

```
int numberMarkers = 4
```

number of markers. Is loaded during start up from the marker configuration file

Definition at line 106 of file main.cpp.

6.5.3.42 pointOrderIndices

```
int pointOrderIndices[] = { 0, 1, 2, 3 }
```

old correspondence from list_points3d and list_points_2d

Definition at line 115 of file main.cpp.

6.5.3.43 pointOrderIndicesNew

```
int pointOrderIndicesNew[] = { 0, 1, 2, 3 }
```

new correspondence from list_points3d and list_points_2d

Definition at line 116 of file main.cpp.

6.5.3.44 portObject

```
int portObject = 9155
```

Port of receiver 1.

Definition at line 134 of file main.cpp.

6.5.3.45 portSafety

```
int portSafety = 9155
```

Port of the safety receiver.

Definition at line 135 of file main.cpp.

6.5.3.46 portSafety2

```
int portSafety2 = 9155
```

Port of receiver 2.

Definition at line 136 of file main.cpp.

6.5.3.47 position

```
Vec3d position = Vec3d()
```

position vector x,y,z for object position in O-CoSy, unit is meter

Definition at line 81 of file main.cpp.

6.5.3.48 positionOld

```
Vec3d positionOld = Vec3d()
```

old position in O-CoSy for finite differences velocity calculation

Definition at line 83 of file main.cpp.

6.5.3.49 posRef

```
Vec3d posRef = Vec3d()
```

initial position of object in camera CoSy

Definition at line 85 of file main.cpp.

6.5.3.50 Rmat

```
Mat Rmat = (cv::Mat_<double>(3, 1) << 0.0, 0.0, 0.0)
```

Rotation, translation etc. matrix for PnP results.

rotation matrix from camera CoSy to marker CoSy

Definition at line 95 of file main.cpp.

6.5.3.51 RmatRef

```
Mat RmatRef = (cv::Mat_<double>(3, 3) << 1., 0., 0., 0., 1., 0., 0., 1.)
```

reference rotation matrix from camera CoSy to marker CoSy

Definition at line 96 of file main.cpp.

6.5.3.52 Rvec

```
Mat Rvec = (cv::Mat_<double>(3, 1) << 0.0, 0.0, 0.0)
```

rotation vector (axis-angle notation) from camera CoSy to marker CoSy

Definition at line 99 of file main.cpp.

6.5.3.53 RvecOriginal

```
Mat RvecOriginal
```

initial values as start values for algorithms and algorithm tests

Definition at line 101 of file main.cpp.

6.5.3.54 safety2Enable

```
bool safety2Enable = false
```

is the second receiver enabled

Definition at line 71 of file main.cpp.

```
6.5.3.55 safetyAngle
```

```
int safetyAngle = 30
```

bank and pitch angle protection in degrees

Definition at line 73 of file main.cpp.

6.5.3.56 safetyBoxLength

```
double safetyBoxLength = 1.5
```

length of the safety area cube in meters

Definition at line 72 of file main.cpp.

6.5.3.57 safetyEnable

```
bool safetyEnable = false
```

is the safety feature enabled

Definition at line 70 of file main.cpp.

6.5.3.58 ss

```
\operatorname{std}::\operatorname{stringstream}\ \operatorname{ss}
```

stream that sends the strBuf buffer to the Qt GUI

Definition at line 142 of file main.cpp.

6.5.3.59 strBuf

```
std::string strBuf
```

buffer that holds the strings that are sent to the Qt GUI

Definition at line 141 of file main.cpp.

6.5.3.60 timeFirstFrame

```
double timeFirstFrame = 0
```

Time stamp of the first frame. This value is then subtracted for every other frame so the time in the log start at zero.

Definition at line 79 of file main.cpp.

6.5.3.61 timeOld

```
double timeOld = 0.0
```

old time for finite differences velocity calculation. Is later on replaced with the hardware timestamp delivered by the camera

Definition at line 78 of file main.cpp.

6.5.3.62 Tvec

```
Mat Tvec = (cv::Mat_<double>(3, 1) << 0.0, 0.0, 0.0)
```

translation vector from camera CoSy to marker CoSy in camera CoSy

Definition at line 100 of file main.cpp.

6.5.3.63 TvecOriginal

```
Mat TvecOriginal
```

initial values as start values for algorithms and algorithm tests

Definition at line 102 of file main.cpp.

6.5.3.64 udpSocketObject

```
QUdpSocket* udpSocketObject
```

socket for the communication with receiver 1

Definition at line 128 of file main.cpp.

6.5.3.65 udpSocketSafety QUdpSocket* udpSocketSafety socket for the communication with safety receiver Definition at line 129 of file main.cpp. 6.5.3.66 udpSocketSafety2 QUdpSocket* udpSocketSafety2 socket for the communication with receiver 3 Definition at line 130 of file main.cpp. 6.5.3.67 useGuess bool useGuess = true set to true and the algorithm uses the last result as starting value Definition at line 104 of file main.cpp. 6.5.3.68 velocity $Vec3d\ velocity = Vec3d()$

Generated by Doxygen

Definition at line 84 of file main.cpp.

velocity vector of object in o-CoSy in respect to o-CoSy

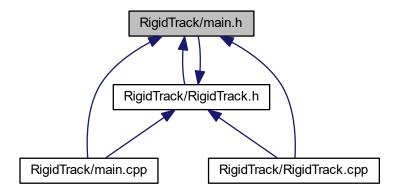
6.6 RigidTrack/main.h File Reference

Header file for main.cpp.

```
#include <fstream>
#include <windows.h>
#include <conio.h>
#include <tchar.h>
#include <stdio.h>
#include <iostream>
#include <stdarg.h>
#include <ctype.h>
#include <stdlib.h>
#include <gl/glu.h>
#include <sstream>
#include <thread>
#include <future>
#include <atomic>
#include "communication.h"
#include "RigidTrack.h"
#include <QtWidgets/QApplication>
#include <QUdpSocket>
#include "cameralibrary.h"
#include "modulevector.h"
#include "modulevectorprocessing.h"
#include "coremath.h"
#include <opencv\cv.h>
#include "opencv2\core.hpp"
#include "opencv2\calib3d.hpp"
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2\video\tracking.hpp>
Include dependency graph for main.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- int startTracking ()
- void startStopCamera ()

Start or stop the tracking depending on if the camera is currently running or not.

- int setReference ()
- int calibrateCamera ()

Start the camera calibration routine that computes the camera matrix and distortion coefficients.

- void loadCalibration (int method)
- · void testAlgorithms ()
- void projectCoordinateFrame (Mat pictureFrame)
- void setUpUDP ()

Open the UDP ports for communication.

- void setHeadingOffset (double d)
- void sendDataUDP (cv::Vec3d &Position, cv::Vec3d &Euler)
- void closeUDP ()
- void loadMarkerConfig (int method)
- void drawPositionText (cv::Mat &Picture, cv::Vec3d &Position, cv::Vec3d &Euler, double error)
- void loadCameraPosition ()
- int determineExposure ()
- void determineOrder ()
- int calibrateGround ()

Variables

• int methodPNP

solvePNP algorithm $0 = iterative \ 1 = EPNP \ 2 = P3P \ 4 = UPNP //! < 4$ and 1 are the same and not implemented correctly by OpenCV

· bool safetyEnable

is the safety feature enabled

• bool safety2Enable

is the second receiver enabled

• double safetyBoxLength

length of the safety area cube in meters

· int safetyAngle

bank and pitch angle protection in degrees

QHostAddress IPAdressObject

IPv4 adress of receiver 1.

· QHostAddress IPAdressSafety

IPv4 adress of safety receiver.

QHostAddress IPAdressSafety2

IPv4 adress of receiver 2.

int portObject

Port of receiver 1.

int portSafety

Port of the safety receiver.

int portSafety2

Port of receiver 2.

int invertZ

dummy variable to invert Z direction on request

commObject commObj

class that handles the communication from main.cpp to the GUI

6.6.1 Detailed Description

Header file for main.cpp.

Author

Florian J.T. Wachter

Version

1.0

Date

April, 8th 2017

6.6.2 Function Documentation

6.6.2.1 calibrateCamera()

```
int calibrateCamera ( )
```

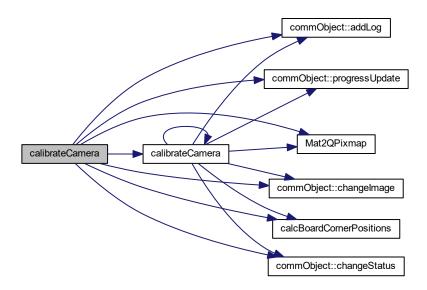
Start the camera calibration routine that computes the camera matrix and distortion coefficients.

Definition at line 774 of file main.cpp.

```
775 {
776
        commObj.addLog("Started camera calibration. 80 pictures are going to be captured.");
777
        CameraLibrary_EnableDevelopment();
778
779
        //! Initialize Camera SDK ==--
780
        CameraLibrary::CameraManager::X();
781
782
        //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
783
        //! them on it's own.
784
785
        //! Get a connected camera ======
786
        CameraManager::X().WaitForInitialization();
787
788
        Camera *camera = CameraManager::X().GetCamera();
789
        if (camera == 0)
790
        {
791
            commObj.addLog("No camera found!");
792
            return 1;
793
794
        //! Determine camera resolution
795
796
        int cameraWidth = camera->Width();
        int cameraHeight = camera->Height();
797
798
799
        //! Set Video Mode ==--
800
801
        //! We set the camera to Segment Mode here. This mode is support by all of our products.
        //! Depending on what device you have connected you might want to consider a different
802
        //! video mode to achieve the best possible tracking quality. All devices that support a
803
        //! mode that will achieve a better quality output with a mode other than Segment Mode are
805
            listed here along with what mode you should use if you're looking for the best head
806
        //! tracking:
807
                V100:R1/R2
808
                              Precision Mode
                              Bit-Packed Precision Mode
809
                TrackIR 5
810
                               Precision Mode
811
                              Precision Mode
812
                S250e
                               Precision Mode
813
        //! If you have questions about a new device that might be conspicuously missing here or
814
        //! have any questions about head tracking, email support or participate in our forums.
815
816
817
        camera->SetVideoType(Core::GrayscaleMode);
818
819
        //! Start camera output ==--
820
        camera->Start();
821
822
        //! Camera Matrix creation
823
        cameraMatrix = Mat::eye(3, 3, CV_64F);
        distCoeffs = Mat::zeros(8, 1, CV_64F);
824
825
826
        //! Ok, start main loop. This loop fetches and displays
827
        //! camera frames.
828
        //! But first set some camera parameters
829
        camera->SetAGC(false);
830
        camera->SetAEC(false);
831
        camera->SetExposure(200);
832
        camera->SetIntensity(4);
833
        camera->SetFrameRate(30);
834
        camera->SetIRFilter(true);
835
        camera->SetContinuousIR(false);
836
        camera->SetHighPowerMode(false);
837
838
        int number_samples = 0;
839
        int imagesToSample = 80;
840
841
        std::vector<std::vector<Point2f> > imagePoints;
        std::vector<Point2f> pointBuf;
842
843
        bool found;
844
        Size boardSize(9, 6);
        Size imageSize(cameraWidth, cameraHeight);
845
        Mat Rvec(3, 1, DataType<double>::type);
Mat Tvec(3, 1, DataType<double>::type);
846
847
848
```

```
//! the user has to provide the size of one square in mm
850
               int qsquareSize = QInputDialog::getInt(nullptr, "Chessboard size in mm", "Chessboard size in mm", 23, 1
851
           , 60, 1, &ok);
               float squareSize = 23;
852
853
854
               if (ok)
855
856
                       squareSize = qsquareSize;
857
858
859
               QPixmap QPFrame;
860
               commObj.progressUpdate(0);
861
               while (number_samples < imagesToSample)</pre>
862
863
                       //! Fetch a new frame from the camera ===---
864
                      cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV 8UC1);
865
866
                      //! which is why we also set this constant to 8
                      const int BACKBUFFER_BITSPERPIXEL = 8;
867
868
869
                       //! later on, when we get the frame as usual:
870
                      CameraLibrary::Frame * frame = camera->GetFrame();
871
872
                       if (frame)
873
                               //! Lets have the Camera Library raster the camera's
874
875
                              //! image into our texture.
876
                              {\tt frame-} \\ {\tt Rasterize} ({\tt cameraWidth, cameraHeight, matFrame.step, BACKBUFFER\_BITSPERPIXEL, matFrame.step)} \\ {\tt raterize} ({\tt cameraWidth, cameraHeight, matFrame.step)} \\ {\tt raterize} ({\tt cameraWidth, camera
877
           data):
878
                              QPFrame = Mat2QPixmap(matFrame);
                              commObj.changeImage(QPFrame);
found = findChessboardCorners(matFrame, boardSize, pointBuf, CV_CALIB_CB_ADAPTIVE_THRESH |
879
880
           CV_CALIB_CB_FAST_CHECK | CV_CALIB_CB_NORMALIZE_IMAGE);
881
882
                               if (found)
                                                                               //!< If done with success,
883
884
                                      //! improve the found corners' coordinate accuracy for chessboard
885
                                      cornerSubPix(matFrame, pointBuf, Size(11, 11), Size(-1, -1), TermCriteria(CV_TERMCRIT_EPS +
             CV_TERMCRIT_ITER, 30, 0.1));
886
887
                                      imagePoints.push_back(pointBuf);
888
                                      number_samples += 1;
                                      commObj.addLog(QString::fromStdString(ss.str()));
890
                                      QCoreApplication::processEvents();
891
                              frame->Release();
ss.str("");
892
893
                              ss << "Samples found = " << number_samples;
894
                              commObj.progressUpdate(number_samples * 100 / imagesToSample);
895
896
897
                       Sleep(2);
898
               }
899
900
               std::vector<std::vector<Point3f> > objectPoints(1);
               calcBoardCornerPositions(boardSize, squareSize, objectPoints[0]);
901
902
               objectPoints.resize(imagePoints.size(), objectPoints[0]);
903
904
               double rms = calibrateCamera(objectPoints, imagePoints, imageSize,
           cameraMatrix, distCoeffs, Rvec, Tvec);
905
               commObj.progressUpdate(0);
906
               //! Release camera ==--
907
               camera->Release();
908
               //! Save the obtained calibration coefficients in a file for later use
QString fileName = QFileDialog::getSaveFileName(nullptr, "Save calibration file", "", "Calibration File
909
910
              (*.xml);; All Files (*)");
911
               FileStorage fs(fileName.toUtf8().constData(), FileStorage::WRITE);
912
               fs << "CameraMatrix" << cameraMatrix;</pre>
913
               fs << "DistCoeff" << distCoeffs;</pre>
914
               fs << "RMS" << rms;
               strBuf = fs.releaseAndGetString();
915
               commObj.changeStatus(QString::fromStdString(strBuf));
916
               commObj.addLog("Saved calibration!");
917
918
               return 0;
919 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.2 calibrateGround()

int calibrateGround ()

Get the pose of the camera w.r.t the ground calibration frame. This frame sets the navigation frame for later results. The pose is averaged over 200 samples and then saved in the file referenceData.xml. This routine is basically the same as setReference.

Definition at line 1563 of file main.cpp.

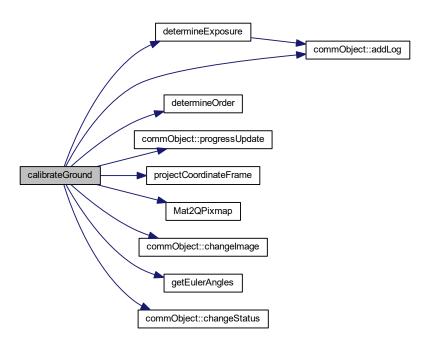
```
1564 {
          //! initialize the variables with starting values
1565
1566
         gotOrder = false;
         posRef = 0:
1567
1568
         eulerRef = 0:
         RmatRef = 0;
1569
1570
         Rvec = RvecOriginal;
1571
         Tvec = TvecOriginal;
1572
1573
         determineExposure();
1574
1575
         ss.str("");
1576
         commObj.addLog("Started ground calibration");
1577
1578
         CameraLibrary_EnableDevelopment();
1579
          //! Initialize Camera SDK =
1580
         CameraLibrary::CameraManager::X();
1581
1582
         //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
1583
         //! them on it's own.
1584
1585
          //! Get a connected camera =========
1586
         CameraManager::X().WaitForInitialization();
1587
         Camera *camera = CameraManager::X().GetCamera();
1588
1589
         //! If no device connected, pop a message box and exit ==--
1590
         if (camera == 0)
1591
1592
              commObj.addLog("No camera found!");
1593
              return 1;
1594
1595
1596
         //! Determine camera resolution to size application window ==----
         int cameraWidth = camera->Width();
int cameraHeight = camera->Height();
1597
1598
         camera->GetDistortionModel(distModel);
1599
1600
         cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
1601
1602
          //! Set camera mode to precision mode, it directly provides marker coordinates
1603
         camera->SetVideoType(Core::PrecisionMode);
1604
1605
          //! Start camera output ==--
1606
         camera->Start();
1607
          /! Turn on some overlay text so it's clear things are
1608
1609
         //! working even if there is nothing in the camera's view. ===---
1610
          //! Set some other parameters as well of the camera
1611
         camera->SetTextOverlay(true);
         camera->SetFrameRate(intFrameRate);
1612
1613
         camera->SetIntensity(intIntensity);
1614
         camera->SetIRFilter(true);
1615
         camera->SetContinuousIR(false);
1616
         camera->SetHighPowerMode(false);
1617
         //! sample some frames and calculate the position and attitude. then average those values and use that
1618
       as zero position
1619
        int numberSamples = 0;
1620
          int numberToSample = 200;
1621
         double projectionError = 0;
1622
1623
         while (numberSamples < numberToSample)</pre>
1624
         {
1625
              //! Fetch a new frame from the camera ===---
             Frame *frame = camera->GetFrame();
1626
1627
1628
              if (frame)
1629
                  //! Ok, we've received a new frame, lets do something
1630
1631
                  //! with it.
1632
                  if (frame->ObjectCount() == numberMarkers)
1633
1634
                      //!for(int i=0; i<frame->ObjectCount(); i++)
1635
                      for (int i = 0; i < numberMarkers; i++)</pre>
1636
                          cObject *obj = frame->Object(i);
1637
                          list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
1638
1639
1640
1641
                      if (gotOrder == false)
1642
1643
                          determineOrder();
1644
1645
1646
                      //! sort the 2d points with the correct indices as found in the preceeding order
       {\tt determination}\ {\tt algorithm}
1647
                      for (int w = 0; w < numberMarkers; w++)</pre>
1648
```

```
1649
                          list_points2d[w] = list_points2dUnsorted[
      pointOrderIndices[w]];
1650
1651
                      list_points2dOld = list_points2dUnsorted;
1652
1653
                      //!Compute the pose from the 3D-2D corresponses
                      solvePnP(list_points3d, list_points2d,
1654
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
1655
                      //! project the marker 3d points with the solution into the camera image CoSy and calculate
1656
       difference to true camera image
1657
                      projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2dProjected);
1658
                      projectionError = norm(list_points2dProjected,
      list_points2d);
1659
1660
                      if (projectionError > 3)
1661
1662
                          commObj.addLog("Reprojection error is bigger than 3 pixel. Correct marker
       configuration loaded?\nMarker position measured precisely?");
1663
                          frame->Release();
1664
                          return 1;
1665
1666
1667
                      double maxValue = 0;
1668
                      double minValue = 0;
1669
                      minMaxLoc(Tvec.at<double>(2), &minValue, &maxValue);
1670
1671
                      if (maxValue > 10000 || minValue < 0)</pre>
1672
1673
1674
1675
                          commObj.addLog("Negative z distance, thats not possible. Start the set
       zero routine again and check marker configurations.");
1676
                          frame->Release();
1677
                          return 1;
1678
1679
                      if (norm(positionOld) - norm(Tvec) < 0.05) //!<Iterative Method needs time</pre>
1680
       to converge to solution
1681
                      {
                          add(posRef, Tvec, posRef);
1682
                          add(eulerRef, Rvec, eulerRef); //! That are not the values of yaw,
1683
       roll and pitch yet! Rodriguez has to be called first.

numberSamples++; //!<-- one sample more :D
1684
1685
                          commObj.progressUpdate(numberSamples * 100 / numberToSample);
1686
                      positionOld = Tvec;
1687
1688
1689
                      Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
1690
                      for (int i = 0; i < numberMarkers; i++)</pre>
1691
1692
                          circle(cFrame, Point(list_points2d[i].x,
      list_points2d[i].y), 6, Scalar(0, 225, 0), 3);
1693
1694
                      projectCoordinateFrame(cFrame);
1695
                      projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2d);
1696
                      for (int i = 0; i < numberMarkers; i++)</pre>
1697
                          circle(cFrame, Point(list_points2d[i].x,
1698
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
1699
1700
1701
                      QPixmap QPFrame;
1702
                      QPFrame = Mat2QPixmap(cFrame);
                      commObj.changeImage(QPFrame);
1703
1704
                      OCoreApplication::processEvents();
1705
1706
1707
                  frame->Release();
1708
             }
1709
         //! Release camera ==--
1710
1711
         camera->Release();
1712
1713
          \ensuremath{//!}\ensuremath{\text{Divide}} by the number of samples to get the mean of the reference position
1714
         divide(posRef, numberToSample, posRef);
         divide(eulerRef, numberToSample, eulerRef); //!< eulerRef is here in Axis Angle
1715
       notation
1716
1717
         Rodrigues(eulerRef, RmatRef);
                                                         //!< axis angle to rotation matrix
1718
         getEulerAngles(RmatRef, eulerRef); //!< rotation matrix to euler
ss.str("");</pre>
1719
1720
1721
         ss << "RmatRef is:\n";
```

```
ss << RmatRef << "\n";
              ss << "Reference Position is:\n";
ss << posRef << "[mm] \n";
ss << "Reference Euler angles are:\n";
ss << eulerRef << "[deg] \n";
1723
1724
1725
1726
1727
         //! Save the obtained calibration coefficients in a file for later use
  QString fileName = QFileDialog::getSaveFileName(nullptr, "Save ground calibration file", "
referenceData.xml", "Calibration File (*.xml);; All Files (*)");
1728
1729
              FileStorage fs(fileName.toUtf8().constData(), FileStorage::WRITE);
fs << "M_NC" << RmatRef;
fs << "eulerRef" << eulerRef;
1730
1731
1732
              strBuf = fs.releaseAndGetString();
1733
1734
              commObj.changeStatus(QString::fromStdString(strBuf));
1735
              commObj.addLog("Saved ground calibration!");
1736
1737
              commObj.progressUpdate(0);
              return 0;
1738 }
```

Here is the call graph for this function:





6.6.2.3 closeUDP()

```
void closeUDP ( )
```

Close the UDP ports again to release network interfaces etc. If this is not done the network resources are still occupied and the program can't exit properly.

Definition at line 1173 of file main.cpp.

```
1174 {
         //! check if the socket is open and if yes close it
if (udpSocketObject->isOpen())
1175
1176
1177
         {
              udpSocketObject->close();
1179
1180
1181
         if (udpSocketSafety->isOpen())
1182
              udpSocketSafety->close();
1183
1185
1186
         if (udpSocketSafety2->isOpen())
1187
              udpSocketSafety2->close();
1188
1189
1190
         commObj.addLog("Closed all UDP ports.");
1191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.4 determineExposure()

```
int determineExposure ( )
```

Get the optimal exposure for the camera. For that find the minimum and maximum exposure were the right number of markers are detected. Then the mean of those two values is used as exposure.

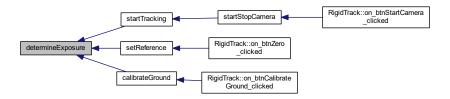
Definition at line 1362 of file main.cpp.

```
1363 {
          //! For OptiTrack Ethernet cameras, it's important to enable development mode if you
1364
1365
          //! want to stop execution for an extended time while debugging without disconnecting
1366
          //! the Ethernet devices. Lets do that now:
1367
1368
          CameraLibrary EnableDevelopment():
1369
1370
          //! Initialize Camera SDK ==--
1371
          CameraLibrary::CameraManager::X();
1372
1373
          //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
1374
          //! them on it's own.
1375
1376
          //! Get a connected camera =========
1377
          CameraManager::X().WaitForInitialization();
1378
          Camera *camera = CameraManager::X().GetCamera();
1379
1380
          //! If no device connected, pop a message box and exit ==--
1381
          if (camera == 0)
1382
          {
              commObj.addLog("No camera found!");
1383
1384
              return 1;
1385
          }
1386
1387
          //! Determine camera resolution to size application window ==----
1388
          int cameraWidth = camera->Width();
1389
          int cameraHeight = camera->Height();
1390
          1391
       {\tt greyscale} \ {\tt imformation} \ {\tt for} \ {\tt marker} \ {\tt property} \ {\tt calculations}
1392
1393
                                                          //! Start camera output ==
1394
1395
          //! Turn on some overlay text so it's clear things are ===---/! working even if there is nothing in the camera's view. ===---
1396
1397
1398
          camera->SetTextOverlay(true);
1399
          camera->SetExposure(intExposure);
                                                   //! set the camera exposure
          camera->SetIntensity(intIntensity); //! set the camera infrared LED intensity camera->SetFrameRate(intFrameRate); //! set the camera framerate to 100 Hz
1400
1401
1402
          camera->SetIRFilter(true); //! enable the filter that blocks visible light and only passes infrared
       light
         camera->SetHighPowerMode(true); //! enable high power mode of the leds
camera->SetContinuousIR(false); //! enable continuous LED light
1403
1404
          camera->SetThreshold(intThreshold); //! set threshold for marker detection
1405
1406
1407
          //!set exposure such that num markers are visible
1408
         int numberObjects = 0; //! Number of objects (markers) found in the current picture with the given
       exposure
          int minExposure = 1; //! exposure when objects detected the first time is numberMarkers int maxExposure = 480; //! exposure when objects detected is first time numberMarkers+1
1409
         int minExposure = 1;
1410
          intExposure = minExposure; //! set the exposure to the smallest value possible
1411
                                   //! if the markers arent found after numberTries then there might be no markers
1412
          int numberTries = 0;
       at all in the real world
1413
1414
                                    //! Determine minimum exposure, hence when are numberMarkers objects detected
          camera->SetExposure(intExposure);
1415
1416
          while (numberObjects != numberMarkers && numberTries < 48)</pre>
1417
              //! get a new camera frame
Frame *frame = camera->GetFrame();
1418
1419
              if (frame) //! frame received
1420
1421
                   numberObjects = frame->ObjectCount();  //! how many objects are detected in the image
1422
1423
                   if (numberObjects == numberMarkers) { minExposure =
      intExposure; frame->Release(); break; } //! if the right amount if markers is found, exit while
1424
                   //! not the right amount of markers was found so increase the exposure and try again
1425
                   numberTries++;
1426
                   intExposure += 10;
1427
                   camera->SetExposure(intExposure);
                   ss.str("");
1428
                  ss << "Exposure: " << intExposure << "\t";
ss << "Objects found: " << numberObjects;</pre>
1429
1430
1431
                   commObj.addLog(OString::fromStdString(ss.str()));
1432
                   frame->Release();
1433
1434
1435
          //! Now determine maximum exposure, hence when are numberMarkers+1 objects detected
1436
          numberTries = 0;
                              //! if the markers arent found after numberTries then there might be no markers at
1437
       all in the real world
1438
          intExposure = maxExposure;
1439
          camera->SetExposure(intExposure);
1440
          numberObjects = 0;
          while (numberObjects != numberMarkers && numberTries < 48)</pre>
1441
1442
```

```
1443
              Frame *frame = camera->GetFrame();
1444
1445
                   numberObjects = frame->ObjectCount(); //! how many objects are detected in the image
1446
      if (numberObjects == numberMarkers) { maxExposure =
intExposure; frame->Release(); break; } //! if the right amount if markers is found, exit while
1447
1448
1449
                   //! not the right amount of markers was found so decrease the exposure and try again
1450
                   intExposure -= 10;
1451
                   numberTries++;
                   camera->SetExposure(intExposure);
1452
                   ss.str("");
1453
                   ss.del( ),
ss << "Exposure: " << intExposure << "\t";
ss << "Objects found: " << numberObjects;</pre>
1454
1455
1456
                   commObj.addLog(QString::fromStdString(ss.str()));
1457
                   frame->Release();
1458
              }
1459
         }
1460
1461
          //! set the exposure to the mean of min and max exposure determined
1462
          camera->SetExposure((minExposure + maxExposure) / 2.0);
1463
1464
          //! and now check if the correct amount of markers is detected with that new value
1465
          while (1)
1466
         {
1467
              Frame *frame = camera->GetFrame();
1468
              if (frame)
1469
              {
                   numberObjects = frame->ObjectCount(); //! how many objects are detected in the image
if (numberObjects != numberMarkers) //! are all markers and not more or less
1470
1471
       detected in the image
1472
1473
                        frame->Release();
1474
                        commObj.addLog("Was not able to detect the right amount of markers.");
1475
                        //! Release camera ==-
1476
                       camera->Release();
1477
                        return 1;
1478
1479
                   else //! all markers and not more or less are found
1480
1481
                        frame->Release();
                        intExposure = (minExposure + maxExposure) / 2.0;
1482
                       commobj.addLog("Exposure set to:");
1483
1484
1485
                        commObj.addLog(QString::number(intExposure));
1486
                       break;
1487
                   }
              }
1488
1489
         }
1490
1491
          camera->Release();
1492
          return 0;
1493
1494 }
```



Here is the caller graph for this function:



6.6.2.5 determineOrder()

```
void determineOrder ( )
```

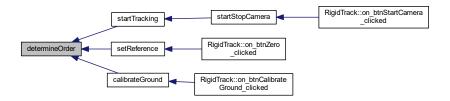
Compute the order of the marker points in 2D so they are the same as in the 3D array. Hence marker 1 must be in first place for both, list_points2d and list_points3d.

Definition at line 1498 of file main.cpp.

```
1499 {
1500
         //! determine the 3D-2D correspondences that are crucial for the PnP algorithm
1501
         //! Try every possible correspondence and solve PnP
1502
         //! Then project the 3D marker points into the 2D camera image and check the difference
1503
         //! between projected points and points as seen by the camera
1504
         //! the corresponce with the smallest difference is probably the correct one
1505
1506
             //! the difference between true 2D points and projected points is super big
1507
         minPointDistance = 5000;
1508
         std::sort(pointOrderIndices, pointOrderIndices + 4);
1509
1510
         //! now try every possible permutation of correspondence
1511
             //! reset the starting values for solvePnP
1512
             Rvec = RvecOriginal;
1513
1514
             Tvec = TvecOriginal;
1515
1516
             //! sort the 2d points with the current permutation
1517
             for (int m = 0; m < numberMarkers; m++)</pre>
1518
1519
                 list_points2d[m] = list_points2dUnsorted[
     pointOrderIndices[m]];
1520
             }
1521
1522
             //! Call solve PNP with P3P since its more robust and sufficient for start value determination
1523
             solvePnP(list_points3d, list_points2d,
     cameraMatrix, distCoeffs, Rvec, Tvec, useGuess, SOLVEPNP P3P):
1524
1525
             //! set the current difference of all point correspondences to zero
1526
             currentPointDistance = 0;
1527
1528
             //! project the 3D points with the solvePnP solution onto 2D \,
1529
             projectPoints(list_points3d, Rvec, Tvec,
     cameraMatrix, distCoeffs, list_points2dProjected);
1530
1531
             //! now compute the absolute difference (error)
1532
             for (int n = 0; n < numberMarkers; n++)</pre>
1533
             {
                 currentPointDistance += norm(list_points2d[n] -
1534
     list_points2dProjected[n]);
1535
1536
1537
             //! if the difference with the current permutation is smaller than the smallest value till now
1538
             //! it is probably the more correct permutation
1539
             if (currentPointDistance < minPointDistance)</pre>
1540
1541
                 minPointDistance = currentPointDistance;
                                                              //!< set the
```

```
smallest value of difference to the current one
1542
                  for (int b = 0; b < numberMarkers; b++)</pre>
                                                                 //!< now safe the better permutation
1543
                       pointOrderIndicesNew[b] = pointOrderIndices[b];
1544
1545
1546
              }
1547
1548
1549
         //! try every permutation
while (std::next_permutation(pointOrderIndices,
1550
1551
      pointOrderIndices + 4));
1552
1553
          //! now that the correct order is found assign it to the indices array
1554
          for (int w = 0; w < numberMarkers; w++)</pre>
1555
1556
              pointOrderIndices[w] = pointOrderIndicesNew[w];
1557
1558
         gotOrder = true;
1559 }
```

Here is the caller graph for this function:



6.6.2.6 drawPositionText()

Draw the position, attitude and reprojection error in the picture.

Parameters

in	Picture	is the camera image in OpenCV matrix format.
in	Position	is the position of the tracked object in navigation CoSy.
in	Euler	are the Euler angles with respect to the navigation frame.
in	error	is the reprojection error of the pose estimation.

Definition at line 1315 of file main.cpp.

```
1320
          ss.str("");
ss << "Y: " << Position[1] << " m";</pre>
1321
1322
          putText(Picture, ss.str(), cv::Point(200, 455), 1, 1, cv::Scalar(255, 255, 255));
1323
1324
1325
         ss.str("");
ss << "Z: " << Position[2] << " m";</pre>
1326
1327
         putText(Picture, ss.str(), cv::Point(200, 470), 1, 1, cv::Scalar(255, 255, 255));
1328
1329
         ss << "Heading: " << Euler[2] << " deg";
putText(Picture, ss.str(), cv::Point(350, 440), 1, 1, cv::Scalar(255, 255, 255));
1330
1331
1332
1333
          ss << "Pitch: " << Euler[1] << " deg";
1334
1335
          putText(Picture, ss.str(), cv::Point(350, 455), 1, 1, cv::Scalar(255, 255, 255));
1336
1337
          ss.str("");
1338
          ss << "Roll: " << Euler[0] << " deg";
1339
         putText(Picture, ss.str(), cv::Point(350, 470), 1, 1, cv::Scalar(255, 255, 255));
1340
1341
          ss << "Error: " << error << " px";
1342
          putText(Picture, ss.str(), cv::Point(10, 470), 1, 1, cv::Scalar(255, 255, 255));
1343
1344 }
```

Here is the caller graph for this function:



6.6.2.7 loadCalibration()

Load a previously saved camera calibration from a file.

Parameters

in	method	whether or not load the camera calibration from calibration.xml. If ==0 then yes, if != 0 then let	1
		the user select a different file.	

Definition at line 923 of file main.cpp.

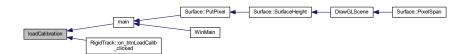
```
923
924
925
         QString fileName;
926
         if (method == 0)
927
928
              fileName = "calibration.xml";
929
930
931
       fileName = QFileDialog::getOpenFileName(nullptr, "Choose a previous saved calibration file", "", "Calibration Files (\star.xml);;All Files (\star)");
932
933
              if (fileName.length() == 0)
934
```

```
935
                      fileName = "calibration.xml";
936
937
938
          FileStorage fs;
          fs.open(fileName.toUtf8().constData(), FileStorage::READ);
939
          fs["CameraMatrix"] >> cameraMatrix;
fs["DistCoeff"] >> distCoeffs;
940
941
942
          commObj.addLog("Loaded calibration from file:");
          commobj.addLog(fileName);
ss.str("");
ss << "\nCamera Matrix is" << "\n" << cameraMatrix << "\n";
ss << "\nDistortion Coefficients are" << "\n" << distCoeffs << "\n";</pre>
943
944
945
946
947
          commObj.addLog(QString::fromStdString(ss.str()));
948 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.8 loadCameraPosition()

```
void loadCameraPosition ( ) \,
```

Load the rotation matrix from camera CoSy to ground CoSy It is determined during calibrateGround() and stays the same once the camera is mounted and fixed.

Definition at line 1348 of file main.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.9 loadMarkerConfig()

Load a marker configuration from file. This file has to be created by hand, use the standard marker configuration file as template.

Parameters

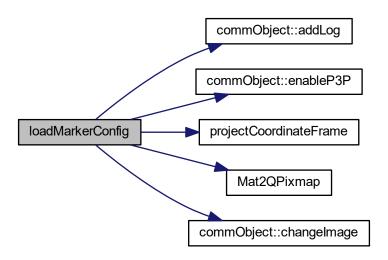
in	method	whether or not load the configuration from the markerStandard.xml. If ==0 load it, if != 0 let the	
		user select a different file.	

Definition at line 1195 of file main.cpp.

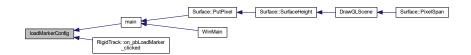
```
1196 {
1197
          QString fileName;
1198
          //! during start up of the programm load the standard marker configuration
1199
             (method == 0)
1200
1201
               //! open the standard marker configuration file
1202
              FileStorage fs;
1203
              fs.open("markerStandard.xml", FileStorage::READ);
1204
              //! copy the values to the respective variables
fs["numberMarkers"] >> numberMarkers;
1205
1206
1207
               //! inizialise vectors with correct length depending on the number of markers
1208
              list_points3d = std::vector<Point3d>(numberMarkers);
list_points2d = std::vector<Point2d>(numberMarkers);
1209
1210
               list_points2dOld = std::vector<Point2d>(numberMarkers);
1212
              list_points2dDifference = std::vector<double>(
      numberMarkers);
1213
              list_points2dProjected = std::vector<Point2d>(
      numberMarkers);
1214
              list_points2dUnsorted = std::vector<Point2d>(
```

```
numberMarkers);
1215
1216
               //! save the marker locations in the points3d vector
1217
               fs["list_points3d"] >> list_points3d;
1218
              fs.release();
              commObj.addLog("Loaded marker configuration from file:");
1219
1220
              commObj.addLog(fileName);
1221
1222
1223
1224
          else
1225
1226
               //! if the load marker configuration button was clicked show a open file dialog
1227
1228
               fileName = QFileDialog::getOpenFileName(nullptr, "Choose a previous saved marker configuration file
      ", "", "marker configuratio files (*.xml);;All Files (*)");
1229
1230
              //! was cancel or abort clicked
              if (fileName.length() == 0)
1231
1232
              {
                   //! if yes load the standard marker configuration
fileName = "markerStandard.xml";
1233
1234
1235
              }
1236
1237
               //! open the selected marker configuration file
1238
              FileStorage fs;
1239
               fs.open(fileName.toUtf8().constData(), FileStorage::READ);
1240
              //! copy the values to the respective variables
fs["numberMarkers"] >> numberMarkers;
1241
1242
1243
1244
               //! inizialise vectors with correct length depending on the number of markers
              list_points2d = std::vector<Point2d>(numberMarkers);
list_points2d = std::vector<Point2d>(numberMarkers);
1245
1246
1247
               list_points2dOld = std::vector<Point2d>(numberMarkers);
              list_points2dDifference = std::vector<double>(numberMarkers);
list_points2dProjected = std::vector<Point2d>(numberMarkers);
1248
1249
               list_points2dUnsorted = std::vector<Point2d>(numberMarkers);
1250
1251
1252
               //! save the marker locations in the points3d vector
1253
               fs["list_points3d"] >> list_points3d;
1254
               fs.release();
              commObj.addLog("Loaded marker configuration from file:");
1255
1256
              commObj.addLog(fileName);
1257
1258
1259
          //! Print out the number of markers and their position to the GUI
1260
          ss.str("");
1261
1262
          ss << "Number of Markers: " << numberMarkers << "\n";
          ss << "Marker 3D Points X,Y and Z [mm]: \n";
1263
1264
          for (int i = 0; i < numberMarkers; i++)</pre>
1265
      ss << "Marker " << i + 1 << ":\t" << list_points3d[i].x << "\t" << list_points3d[i].y << "\t" << list_points3d[i].z << "\n";
1266
1267
1268
          commObj.addLog(QString::fromStdString(ss.str()));
1269
1270
          //! check if P3P algorithm can be enabled, it needs exactly 4 marker points to work
1271
          if (numberMarkers == 4)
1272
1273
               //! if P3P is possible, let the user choose which algorithm he wants but keep iterative active
1274
              methodPNP = 0;
1275
              commObj.enableP3P(true);
1276
          else
1277
1278
1279
              //! More (or less) marker than 4 loaded, P3P is not possible, hence user cant select P3P in GUI
1280
              methodPNP = 0;
1281
              commObj.enableP3P(false);
1282
              commObj.addLog("P3P algorithm disabled, only works with 4 markers.");
1283
1284
          //! now display the marker configuration in the camera view
1285
          Mat cFrame (480, 640, CV_8UC3, Scalar(0, 0, 0));
1286
1287
1288
          //! Set the camera pose parallel to the marker coordinate system
1289
          Tvec.at<double>(0) = 0;
Tvec.at<double>(1) = 0;
1290
          Tvec.at<double>(2) = 4500:
1291
          Rvec.at<double>(1) = 0 * 3.141592653589 / 180.0;
Rvec.at<double>(1) = 0 * 3.141592653589 / 180.0;
1292
1293
1294
          Rvec.at<double>(2) = -90. * 3.141592653589 / 180.0;
1295
1296
          projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
  for (int i = 0; i < numberMarkers; i++)</pre>
1297
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.10 projectCoordinateFrame()

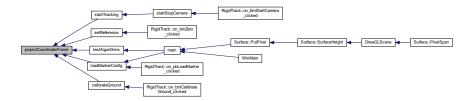
Project the coordinate CoSy origin and axis direction of the marker CoSy with the rotation and translation of the object for visualization.

Parameters

d.

Definition at line 1081 of file main.cpp.

Here is the caller graph for this function:



6.6.2.11 sendDataUDP()

Send the position and attitude over UDP to every receiver, the safety receiver is handled on its own in the start

Tracking function because its send rate is less than 100 Hz.

Definition at line 1154 of file main.cpp.

```
1155 {
1156
         datagram.clear();
1157
         QDataStream out(&datagram, QIODevice::WriteOnly);
         out.setVersion(QDataStream::Qt_4_3);
1158
         out << (float)Position[0] << (float)Position[1] << (float)Position[2];</pre>
1160
         out << (float)Euler[0] << (float)Euler[1] << (float)Euler[2]; //! Roll Pitch Heading</pre>
1161
         udpSocketObject->writeDatagram(datagram,
      IPAdressObject, portObject);
1162
1163
         //! if second receiver is activated send it also the tracking data
1164
         if (safety2Enable)
1166
             udpSocketSafety2->writeDatagram(datagram,
      IPAdressSafety2, portSafety2);
1167
1168
1169 }
```

Here is the caller graph for this function:



6.6.2.12 setHeadingOffset()

```
void setHeadingOffset ( double d )
```

Add a heading offset to the attitude for the case it is wanted by the user.

Parameters

in	d	denotes heading offset in degrees.	1
----	---	------------------------------------	---

Definition at line 1122 of file main.cpp.

```
1123 {
1124
         headingOffset = d;
1125
         d = d * 3.141592653589 / 180.0; //! Convert heading offset from degrees to rad
1126
         //! Calculate rotation about x axis
1127
1128
         Mat R_x = (Mat_{double})(3, 3) <<
             1, 0, 0,
1130
             0, 1, 0,
1131
             0, 0, 1
1132
1133
         //! Calculate rotation about y axis
1134
1135
         Mat R_y = (Mat_<double>(3, 3) <<
             1, 0, 0,
0, 1, 0,
1137
1138
             0, 0, 1
1139
             );
1140
         //! Calculate rotation about z axis
1141
1142
         Mat R_z = (Mat_{double})(3, 3) <<
1143
             cos(d), -sin(d), 0,
             sin(d), cos(d), 0, 0, 0, 1);
1144
1145
1146
1147
         //! Combined rotation matrix
1149
         M_HeadingOffset = R_z * R_y * R_x;
1150 }
```

```
Surface::PutPixel Surface::SurfaceHeight DrawGLScene Surface::PixelSpan

Surface::PixelSpan

WinMain

RigidTrack::on_sbHeading
Offset_valueChanged
```

6.6.2.13 setReference()

```
int setReference ( )
```

Determine the initial position of the object that serves as reference point or as ground frame origin. Computes the pose 200 times and then averages it. The position and attitude are from now on used as navigation CoSy.

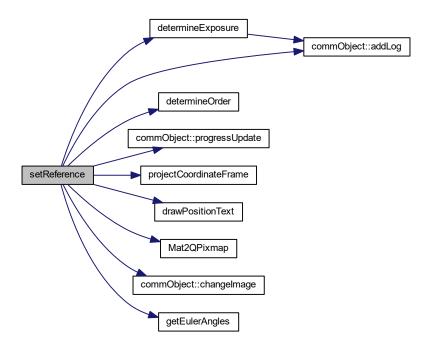
Definition at line 595 of file main.cpp.

```
596 {
597
        //! initialize the variables with starting values
598
        gotOrder = false;
599
        posRef = 0;
600
        eulerRef = 0;
601
        RmatRef = 0;
        Rvec = RvecOriginal;
602
603
        Tvec = TvecOriginal;
604
605
        determineExposure();
606
607
        ss.str("");
        commObj.addLog("Started reference coordinate determination.");
608
609
610
        CameraLibrary EnableDevelopment();
611
        //! Initialize Camera SDK
612
        CameraLibrary::CameraManager::X();
613
        //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
614
615
        //! them on it's own.
616
        //! Get a connected camera ==
618
        CameraManager::X().WaitForInitialization();
619
        Camera *camera = CameraManager::X().GetCamera();
620
621
        //! If no device connected, pop a message box and exit ==--
622
        if (camera == 0)
623
        {
624
            commObj.addLog("No camera found!");
625
            return 1;
626
627
628
        //! Determine camera resolution to size application window ==----
629
        int cameraWidth = camera->Width();
630
        int cameraHeight = camera->Height();
631
        camera->GetDistortionModel(distModel);
632
        cv::Mat matFrame(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
633
        //! Set camera mode to precision mode, it directly provides marker coordinates
634
635
        camera->SetVideoType(Core::PrecisionMode);
636
637
        //! Start camera output ==--
638
        camera->Start();
639
        //! Turn on some overlay text so it's clear things are
640
641
        //! working even if there is nothing in the camera's view. ===---
        //! Set some other parameters as well of the camera
642
643
        camera->SetTextOverlay(true);
644
        camera->SetFrameRate(intFrameRate);
645
        camera->SetIntensity(intIntensity);
646
        camera->SetIRFilter(true);
647
        camera->SetContinuousIR(false);
648
        camera->SetHighPowerMode(false);
649
650
        //! sample some frames and calculate the position and attitude. then average those values and use that
       as zero position
651
        int numberSamples = 0;
        int numberToSample = 200;
652
        double projectionError = 0; //! < difference between the marker points as seen by the camera and the
653
       projected marker points with Rvec and Tvec
654
655
        while (numberSamples < numberToSample)</pre>
656
            //! Fetch a new frame from the camera ===---
657
658
            Frame *frame = camera->GetFrame();
659
660
            if (frame)
661
                //! Ok, we've received a new frame, lets do something
662
                //! with it.
663
664
                if (frame->ObjectCount() == numberMarkers)
665
```

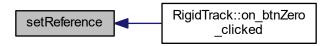
```
666
                     //!for(int i=0; i<frame->ObjectCount(); i++)
                     for (int i = 0; i < numberMarkers; i++)</pre>
667
668
669
                         cObject *obj = frame->Object(i);
670
                         list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
671
                     }
672
673
                     if (gotOrder == false)
674
675
                         determineOrder();
676
677
678
                     //! sort the 2d points with the correct indices as found in the preceeding order
       determination algorithm
679
                     for (int w = 0; w < numberMarkers; w++)</pre>
680
                         list_points2d[w] = list_points2dUnsorted[
681
      pointOrderIndices[w]];
682
683
                     list_points2dOld = list_points2dUnsorted;
684
685
                     //!Compute the pose from the 3D-2D corresponses
686
                     solvePnP(list_points3d, list_points2d,
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
687
                    //! project the marker 3d points with the solution into the camera image CoSy and calculate
688
       difference to true camera image
689
                    projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2dProjected);
690
                    projectionError = norm(list_points2dProjected,
      list_points2d);
691
692
                    double maxValue = 0;
                    double minValue = 0;
693
                    minMaxLoc(Tvec.at < double > (2), & minValue, & maxValue);
694
695
696
                     if (maxValue > 10000 || minValue < 0)</pre>
697
698
                         ss.str("");
699
                         ss << "Negative z distance, thats not possible. Start the set zero routine again or
       restart Programm.";
700
                         commObj.addLog(OString::fromStdString(ss.str()));
701
                         frame->Release();
702
                         return 1;
703
                     }
704
705
                     if (projectionError > 3)
706
                         commObj.addLog("Reprojection error is bigger than 3 pixel. Correct marker
707
       configuration loaded?\nMarker position measured precisely?");
708
                         frame->Release();
709
                         return 1;
710
                     }
711
                     if (norm(positionOld) - norm(Tvec) < 0.05) //!<Iterative Method needs time</pre>
712
       to converge to solution
713
714
                         add(posRef, Tvec, posRef);
       add(eulerRef, Rvec, eulerRef); //!< That are not the values of yaw, roll and pitch yet! Rodriguez has to be called first.

numberSamples++; //!< one sample more :D
715
716
717
                         commObj.progressUpdate(numberSamples * 100 / numberToSample);
718
                    positionOld = Tvec;
719
720
                    Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
721
722
                     for (int i = 0; i < numberMarkers; i++)</pre>
723
724
                         circle(cFrame, Point(list_points2d[i].x,
      list_points2d[i].y), 6, Scalar(0, 225, 0), 3);
725
                     projectCoordinateFrame(cFrame);
726
      727
728
729
                     {
730
                         circle(cFrame, Point(list_points2d[i].x,
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
731
                    drawPositionText(cFrame, position,
732
      eulerAngles, projectionError);
733
734
                    QPixmap QPFrame;
735
                    QPFrame = Mat2QPixmap(cFrame);
                     commObj.changeImage(QPFrame);
736
737
                    QCoreApplication::processEvents();
```

```
738
739
740
                  frame->Release();
741
             }
742
743
         //! Release camera ==--
744
        camera->Release();
745
746
         \protect\//\protect\/ Divide by the number of samples to get the mean of the reference position
747
        divide(posRef, numberToSample, posRef);
748
        divide(eulerRef, numberToSample, eulerRef); //!< eulerRef is here in Axis Angle
       notation
749
750
        Rodrigues(eulerRef, RmatRef);
                                                           //!< axis angle to rotation matrix
751
         //!-- Euler Angles, finally
        getEulerAngles(RmatRef, eulerRef); //!< rotation matrix to euler
ss.str("");</pre>
752
753
        ss.stf( );
ss << "RmatRef is:\n";
ss << RmatRef << "\n";
754
755
756
        ss << "Reference Position is:\n";
        ss << posRef << "[mm] \n";
ss << "Reference Euler Angles are:\n";
ss << eulerRef << "[deg] \n";
757
758
759
760
761
        //! compute the difference between last obtained TVec and the average Value
        //! When it is large the iterative method has not converged properly so it is advised to start the
762
       setReference() function once again
763
       double error = norm(posRef) - norm(Tvec);
764
         if (error > 5.0)
765
             ss << "Caution, distance between reference position and last position is: " << error << "\n Start
766
       the set zero routine once again.";
767
768
         commObj.addLog(QString::fromStdString(ss.str()));
769
        commObj.progressUpdate(0);
770
         return 0;
771 }
```



Here is the caller graph for this function:



6.6.2.14 setUpUDP()

```
void setUpUDP ( )
```

Open the UDP ports for communication.

Definition at line 1090 of file main.cpp.

```
1091 {
1092
          //! Initialise the QDataStream that stores the data to be send
          QDataStream out(&datagram, QIODevice::WriteOnly);
1094
          out.setVersion(QDataStream::Qt_4_3);
1095
         //! Create UDP slots
commObj.addLog("Opening UDP ports.");
udpSocketObject = new QUdpSocket(0);
1096
1097
1098
          udpSocketObject->connectToHost(IPAdressObject,
1099
1100
         commObj.addLog("Opened first receiver UDP port.");
1101
         udpSocketSafety = new QUdpSocket(0);
udpSocketSafety2 = new QUdpSocket(0);
1102
1103
1104
1105
          //! if the safety feature is activated open the udp port
1106
1107
              udpSocketSafety->connectToHost(IPAdressSafety,
1108
      portSafety);
1109
              commObj.addLog("Opened safety UDP port.");
1110
1111
1112
          //! if the second receiver feature is activated open the udp port
1113
         if (safety2Enable)
1114
              udpSocketSafety2->connectToHost(IPAdressSafety2,
1115
      portSafety2);
1116
              commObj.addLog("Opened second receiver UDP port.");
1117
1118 }
```



Here is the caller graph for this function:



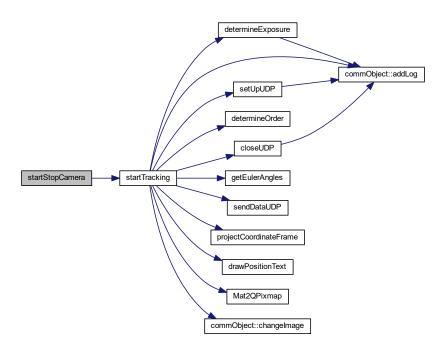
6.6.2.15 startStopCamera()

```
void startStopCamera ( )
```

Start or stop the tracking depending on if the camera is currently running or not.

Definition at line 579 of file main.cpp.

```
580 {
581
        //! tracking is not running so start it
582
        if (exitRequested)
583
        {
584
            exitRequested = false;
585
            startTracking();
586
587
        else //!< tracking is currently running, set exitRequest to true so the while loop in startTracking()</pre>
       exits
588
589
            exitRequested = true;
591 }
```



Here is the caller graph for this function:



6.6.2.16 startTracking()

```
int startTracking ( )
```

Start the loop that fetches frames, computes the position etc and sends it to other computers. This function is the core of this program, hence the pose estimation is done here.

Definition at line 261 of file main.cpp.

```
261
262
264
        gotOrder = false; //! The order of points, hence which entry in list_points3d corresponds to
       which in list_points2d is not calculated yet

Rvec = RvecOriginal; //! Use the value of Rvec that was set in main() as starting value
265
        for the solvePnP algorithm
266
         Tvec = TvecOriginal; //! Use the value of Tvec that was set in main() as starting value
        for the solvePnP algorithm
2.67
        GetLocalTime(&logDate);
                                    //! Get the current date and time to name the log file
268
         //! Concat the log file name as followed. The file is saved in the folder /logs in the Rigid Track
269
       installation folder
      logFileName = "./logs/positionLog_" + QString::number(logDate.wDay) + "_" +
QString::number(logDate.wMonth) + "_" + QString::number(logDate.wYear);
270
      logFileName += "_" + QString::number(logDate.wHour) + "_" + QString::number(logDate.wMinute) + "_" + QString::number(logDate.wSecond) + ".txt";
271
272
        logName = logFileName.toStdString(); //! Convert the QString to a standard string
273
274
        determineExposure(); //! Get the exposure where the right amount of markers is
275
276
         //! For OptiTrack Ethernet cameras, it's important to enable development mode if you
277
         //! want to stop execution for an extended time while debugging without disconnecting
278
         //! the Ethernet devices. Lets do that now:
279
280
         CameraLibrary_EnableDevelopment();
281
         CameraLibrary::CameraManager::X(); //! Initialize Camera SDK
282
283
         //! At this point the Camera SDK is actively looking for all connected cameras and will initialize
        //! them on it's own
284
285
286
         //! Get a connected camera
287
         CameraManager::X().WaitForInitialization();
288
         Camera *camera = CameraManager::X().GetCamera();
289
290
         //! If no camera can be found, inform user in message log and exit function
291
         if (camera == 0)
292
293
             commObj.addLog("No camera found!");
294
295
296
297
         //! Determine camera resolution to size application window
298
         int cameraWidth = camera->Width();
299
        int cameraHeight = camera->Height();
```

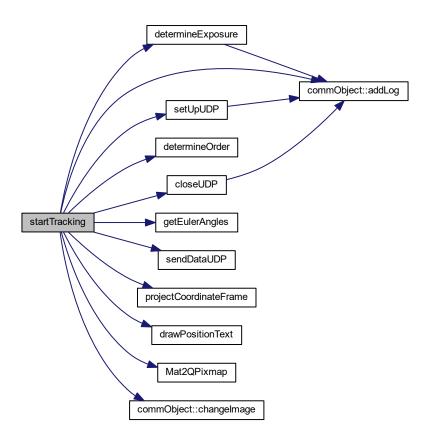
```
300
             \verb|camera->SetVideoType(Core::PrecisionMode); | //! Set the camera mode to precision mode, it used to precision mode in the camera mode mode 
301
            greyscale imformation for marker property calculations
302
303
             camera->Start(); //! Start camera output
304
305
             //! Turn on some overlay text so it's clear things are
306
             //! working even if there is nothing in the camera's view
307
             camera->SetTextOverlay(true);
308
             camera->SetExposure(intExposure);
                                                                           //! Set the camera exposure
             camera ->SetIntensity(intIntensity); //! Set the camera infrared LED intensity
camera->SetFrameRate(intFrameRate); //! Set the camera framerate to 100 Hz
309
310
             camera->SetIRFilter(true); //! Enable the filter that blocks visible light and only passes infrared
311
            light
312
             camera->SetHighPowerMode(true); //! Enable high power mode of the LEDs
313
             camera->SetContinuousIR(false); //! Disable continuous LED light
             camera->SetThreshold(intThreshold); //! Set threshold for marker detection
314
315
316
             //! Create a new matrix that stores the grayscale picture from the camera
317
             Mat matFrame = Mat::zeros(cv::Size(cameraWidth, cameraHeight), CV_8UC1);
             QPixmap QPFrame; //! QPixmap is the corresponding Qt class that saves images
318
319
             //! Matrix that stores the colored picture, hence marker points, coordinate frame and reprojected
            points
320
             Mat cFrame(480, 640, CV 8UC3, Scalar(0, 0, 0));
321
322
             int v = 0; //! Helper variable used to kick safety switch
             //! Variables for the min and max values that are needed for sanity checks
323
324
             double maxValue = 0;
325
             double minValue = 0;
             int framesDropped = 0; //! Ff a marker is not visible or accuracy is bad increase this counter double projectionError = 0; //! Equals the quality of the tracking
326
327
328
329
             \mathtt{setUpUDP}(); //! Open sockets and ports for UDP communication
330
331
             if (safetyEnable) //! If the safety feature is enabled send the starting message
332
333
                    //! Send enable message, hence send a 9 and then a 1
334
                    data.setNum((int)(9));
335
                    udpSocketSafety->write(data);
336
                    data.setNum((int)(1));
337
                   udpSocketSafety->write(data);
338
             }
339
340
             //! Fetch a new frame from the camera
             bool gotTime = false; //! Get the timestamp of the first frame. This time is subtracted from every
341
            subseeding frame so the time starts at 0 in the logs
342
             while (!gotTime) //! While no new frame is received loop
343
                   Frame *frame = camera->GetFrame(); //! Get a new camera frame
if (frame) //! There is actually a new frame
344
345
346
                    {
                          \label{timeFirstFrame} \verb| = frame -> TimeStamp(); //! Get the time stamp for the first frame.
347
            It is subtracted for the following frames
348
                          frame->Release();
                                                           //! Release the frame so the camera can continue
                          gotTime = true; //! Exit the while loop
349
350
                   }
351
             }
352
             //! Now enter the main loop that processes each frame and computes the pose, sends it and logs stuff while (!exitRequested) //! Check if the user has not pressed "Stop Tracking" yet
353
354
355
356
357
                    Frame *frame = camera->GetFrame(); //! Fetch a new frame from the camera
358
359
                    if (frame) //! Did we got a new frame or does the camera still need more time
360
361
                          frames Dropped ++; \ //! \ Increase \ by \ one, \ if \ everything \ is \ okay \ it \ is \ decreased \ at \ the \ end \ of \ the
            loop again
362
363
                           //! Only use this frame it the right number of markers is found in the picture
364
                           if (frame->ObjectCount() == numberMarkers)
365
366
                                  //! Get the marker points in 2D in the camera image frame and store them in the
            list_points2dUnsorted vector
367
                                 //! The order of points that come from the camera corresponds to the Y coordinate
                                  for (int i = 0; i < numberMarkers; i++)</pre>
368
369
                                 {
370
                                        cObject *obj = frame->Object(i);
371
                                        list_points2dUnsorted[i] = cv::Point2d(obj->X(), obj->Y());
372
                                 }
373
374
                                  if (gotOrder == false) //! Was the order already determined? This is false for the
            first frame and from then on true
375
376
                                        determineOrder(); //! Now compute the order
377
                                  }
378
```

```
379
                     //! Sort the 2d points with the correct indices as found in the preceeding order
       determination algorithm
380
                     for (int w = 0; w < numberMarkers; w++)</pre>
381
      list_points2d[w] = list_points2dUnsorted[
pointOrderIndices[w]]; //! pointOrderIndices was calculated in determineOrder()
382
383
384
                     list_points2dOld = list_points2dUnsorted;
385
386
                     //! The first time the 2D-3D corresspondence was determined with gotOrder was okay.
387
                     //! But this order can change as the object moves and the marker objects appear in a
                     //! different order in the frame->Object() array.
388
389
                     //! The solution is that: When a marker point (in the camera image, hence in 2D) was at
390
                     //! a position then it wont move that much from one frame to the other.
391
                     //! So for the new frame we take a marker object and check which marker was closest this
       point
                     //! in the old image frame? This is probably the same (true) marker. And we do that for
392
       every other marker as well.
393
                     //! When tracking is good and no frames are dropped because of missing markers this should
       work every frame.
394
                     for (int j = 0; j < numberMarkers; j++)</pre>
395
396
                         minPointDistance = 5000; //! The sum of point distances is set to
       something unrealistic large
397
                         for (int k = 0; k < numberMarkers; k++)</pre>
398
399
                              //! Calculate N_2 norm of unsorted points minus old points
400
                             currentPointDistance = norm(
      list_points2dUnsorted[pointOrderIndices[j]] -
      list_points2dOld[k]);
                              //! If the norm is smaller than minPointDistance the correspondence is more likely
401
       to be correct
402
                              if (currentPointDistance <</pre>
      minPointDistance)
403
                                  //! Update the array that saves the new point order
404
405
                                 minPointDistance =
      currentPointDistance;
406
                                 pointOrderIndicesNew[j] = k;
407
408
                         }
409
                     }
410
411
                     //! Now the new order is found, set the point order to the new value
412
                     for (int k = 0; k < numberMarkers; k++)</pre>
413
414
                         pointOrderIndices[k] = pointOrderIndicesNew[k];
415
                         list_points2d[k] = list_points2dUnsorted[
      pointOrderIndices[k]];
416
                    }
417
418
                     //! Save the unsorted position of the marker points for the next loop
419
                     list_points2dOld = list_points2dUnsorted;
420
                     //!Compute the object pose from the 3D-2D corresponses
421
                     solvePnP(list_points3d, list_points2d,
422
      cameraMatrix, distCoeffs, Rvec, Tvec, useGuess,
      methodPNP);
423
424
                     //! Project the marker 3d points with the solution into the camera image CoSy and calculate
       difference to true camera image
425
                     projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2dProjected);
                     projectionError = norm(list_points2dProjected,
426
      list_points2d); //! Difference of true pose and found pose
427
428
                     //! Increase the framesDropped variable if accuracy of tracking is too bad
429
                     if (projectionError > 5)
430
                     {
431
                         framesDropped++;
432
                     else
433
434
                         framesDropped = 0; //! Set number of subsequent frames dropped to zero because error
435
       is small enough and no marker was missing
436
437
438
                     //! Get the min and max values from TVec for sanity check
439
                     minMaxLoc(Tvec.at < double > (2), & minValue, & maxValue);
440
                     //! Sanity check of values. negative z means the marker CoSy is behind the camera, that's
441
       not possible.
442
                     if (minValue < 0)
443
444
                         {\tt commObj.addLog("Negative\ z\ distance,\ that\ is\ not\ possible.\ Start\ the\ set}
       zero routine again or restart Program.");
445
                         frame->Release(); //! Release the frame so the camera can move on
```

```
446
                         camera->Release(); //! Release the camera
                         closeUDP(); //! Close all UDP connections so the programm can be closed later
447
       on and no resources are locked
448
                         return 1; //! Exit the function
449
450
451
                     //! Next step is the transformation from camera CoSy to navigation CoSy
452
                     //! Compute the relative object position from the reference position to the current one
453
                      //! given in the camera CoSy: f T_C^{NM} = Tvec - Tvec_{Ref} \f
454
                     subtract(Tvec, posRef, position);
455
                     //! Transform the position from the camera CoSy to the navigation CoSy with INS alligned
456
       457
458
                     Mat V = 0.001 * M_HeadingOffset * M_CN.t() * (Mat)
      position;
                     position = V;    //! Position is the result of the preceeding calculation
position[2] *= invertZ;    //! Invert Z if check box in GUI is activated,
459
460
       hence height above ground is considered
461
                      //! Realtive angle between reference orientation and current orientation
462
463
                     Rodrigues (Rvec, Rmat); //! Convert axis angle respresentation to ordinary rotation
       matrix
464
465
                     //! The difference of the reference rotation and the current rotation
                     //! \f$ R_{ NM } = M_{ NC } \times R_{ CM } \f$
466
                     Rmat = RmatRef.t() *Rmat;
467
468
469
                     //! Euler Angles, finally
                     getEulerAngles(Rmat, eulerAngles); //! Get the euler angles
470
       from the rotation matrix
471
                     eulerAngles[2] += headingOffset; //! Add the heading offset to the
472
473
                     //! Compute the velocity with finite differences. Only use is the log file. It is done here
       because the more precise time stamp can be used
474
                     frameTime = frame->TimeStamp() - timeOld; //! Time between the old frame
       and the current frame
475
                     timeOld = frame->TimeStamp();
                                                         //! Set the old frame time to the current one
                     velocity[0] = (position[0] - positionOld[0]) /
476
      frameTime; //! Calculate the x velocity with finite differences
                     velocity[1] = (position[1] - positionOld[1]) /
477
      478
      frameTime; //! Calculate the z velocity with finite differences
positionOld = position; //! Set the old position to the current one for
479
       next frame velocity calcuation
480
                     //! Send position and Euler angles over WiFi with 100 Hz sendDataUDP (position, eulerAngles);
481
482
483
484
                     //! Save the values in a log file, values are:
                     //! Time sinc tracking started Position Euler Angles Velocity
logfile.open(logName, std::ios::app); //! Open the log file, the folder is
485
486
       RigidTrackInstallationFolder/logs
                     logfile << frame->TimeStamp() - timeFirstFrame << ";" <</pre>
487
      position[0] << ";" << position[1] << ";" << position[2] << ";";
      logfile << eulerAngles[0] << ";" <<
eulerAngles[1] << ";" << eulerAngles[2] << ";";</pre>
488
                     logfile << velocity[0] << ";" << velocity[1] << ";" <<
489
      velocity[2] << "\n";</pre>
490
                     logfile.close(); //! Close the file to save values
491
492
493
                 //! Check if the position and euler angles are below the allowed value, if yes send OKAY signal
        (1), if not send shutdown signal (0)
494
                 //! Absolute x, y and z position in navigation CoSy must be smaller than the allowed distance
495
                 if (safetyEnable)
496
      if ((abs(position[0]) < safetyBoxLength && abs(position[1]) <
safetyBoxLength && abs(position[2]) < safetyBoxLength))</pre>
497
498
499
                          //! Absolute Euler angles must be smaller than allowed value. Heading is not considered
500
                         if ((abs(eulerAngles[0]) < safetyAngle && abs(eulerAngles[1]) <</pre>
      safetyAngle))
501
502
                              //! Send the OKAY signal to the desired computer every 5th time
503
                              if (v == 5) {
504
                                  data.setNum((int)(1)):
                                  udpSocketSafety->write(data); //! Send the 1
505
                                  v = 0; //! reset the counter that is needed for decimation to every 5th time
506
       step
507
508
509
                          //! The euler angles of the object exceeded the allowed euler angles, send the shutdown
       signal (0)
```

```
510
                         else
511
512
                             data.setNum((int)(0)); //! Send the shutdown signal, a 0
                             udpSocketSafety->write(data);
commObj.addLog("Object exceeded allowed Euler angles, shutdown signal
513
514
       sent."); //! Inform the user
515
516
517
                     ^{\prime}/^{\prime}! The position of the object exceeded the allowed position, shut the object down
518
519
                    else
520
521
                         data.setNum((int)(0)); //! Send the shutdown signal, a 0
522
                         udpSocketSafety->write(data);
523
                         commObj.addLog("Object left allowed area, shutdown signal sent."); //!
       Inform the user
524
525
526
                }
527
528
                 //! Inform the user if tracking system is disturbed (marker lost or so) or error was too big
529
                if (framesDropped > 10)
530
                     if (safetyEnable) //! Also send the shutdown signal
531
532
533
                         data.setNum((int)(0)); //! Send the shutdown signal, a 0
534
                         udpSocketSafety->write(data);
535
536
                     commObj.addLog("Lost marker points or precision was bad!"); //! Inform the
       user
537
                    framesDropped = 0;
538
                }
539
540
                //!\ {\tt Rasterize} the frame so it can be shown in the GUI
541
                frame->Rasterize(cameraWidth, cameraHeight, matFrame.step,
      BACKBUFFER_BITSPERPIXEL, matFrame.data);
542
543
                //! Convert the frame from greyscale as it comes from the camera to rgb color
                cvtColor(matFrame, cFrame, COLOR_GRAY2RGB);
544
545
546
                //! Project (draw) the marker CoSy origin into 2D and save it in the cFrame image
547
                projectCoordinateFrame(cFrame);
548
549
                //! Project the marker points from 3D to the camera image frame (2d) with the computed pose
550
                projectPoints(list_points3d, Rvec, Tvec,
      cameraMatrix, distCoeffs, list_points2d);
551
                for (int i = 0; i < numberMarkers; i++)</pre>
552
                {
                    //! Draw a circle around the projected points so the result can be better compared to the
553
       real marker position
554
                    //! In the resulting picture those are the red dots
                     circle(cFrame, Point(list_points2d[i].x,
555
      list_points2d[i].y), 3, Scalar(225, 0, 0), 3);
556
               }
557
558
                //! Write the current position, attitude and error values as text in the frame
                drawPositionText(cFrame, position, eulerAngles, projectionError);
559
560
561
                 //! Send the new camera picture to the GUI and call the GUI processing routine
562
                QPixmap QPFrame;
563
                OPFrame = Mat2OPixmap(cFrame);
                commObj.changeImage(QPFrame); //! Update the picture in the GUI
564
565
                QCoreApplication::processEvents(); //! Give Qt time to handle everything
567
                 //! Release the camera frame to fetch the new one
568
                frame->Release();
569
            }
570
        }
571
        //! User choose to stop the tracking, clean things up
573
        closeUDP(); //! Close the UDP connections so resources are deallocated
574
        camera->Release(); //! Release camera
575
        return 0;
576 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.2.17 testAlgorithms()

void testAlgorithms ()

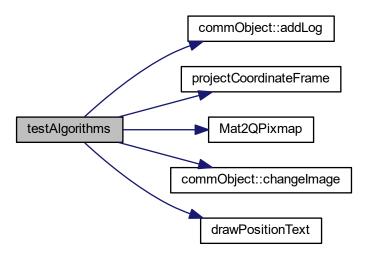
Project some points from 3D to 2D and then check the accuracy of the algorithms. Mainly to generate something that can be shown in the camera view so the user knows everything loaded correctly.

Definition at line 952 of file main.cpp.

```
953 {
954
955
        int _methodPNP;
956
957
        std::vector<Point2d> noise(numberMarkers);
958
959
        RvecOriginal = Rvec;
960
        TvecOriginal = Tvec;
961
962
        projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
963
964
        ss.str("");
965
        ss << "Unsorted Points 2D Projected \n";
966
        ss << list_points2dProjected << "\n";
967
        commObj.addLog(QString::fromStdString(ss.str()));
968
       Mat cFrame(480, 640, CV_8UC3, Scalar(0, 0, 0));
for (int i = 0; i < numberMarkers; i++)</pre>
969
970
971
972
            circle(cFrame, Point(list_points2dProjected[i].x, list_points2dProjected[i].y), 6, Scalar(0, 255, 0
      ), 3);
973
        }
974
975
        projectCoordinateFrame(cFrame);
976
977
        ss << "======\n";
978
        ss << "======= Projected Points =======\n";
979
        ss << list_points2dProjected << "\n";
980
981
982
        randn(noise, 0, 0.5);
983
        add(list_points2dProjected, noise, list_points2dProjected);
984
        985
986
987
        commObj.addLog(QString::fromStdString(ss.str()));
988
989
990
        bool useGuess = true;
        _methodPNP = 0; //!< 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //!< not used
991
992
        solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
993
      distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
994
995
        ss.str("");
        ss << "-----\n";
ss << "-----\n";
996
997
        ss << "rvec: " << "\n";
998
        ss << Rvec << "\n";
ss << "tvec: " << "\n";
999
1000
        ss << Tvec << "\n";
1001
1002
1003
        commObj.addLog(QString::fromStdString(ss.str()));
1004
1005
         _methodPNP = 1; //!< 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP UPnP not used
         Rvec = cv::Mat::zeros(3, 1, CV_64F);
Tvec = cv::Mat::zeros(3, 1, CV_64F);
1006
1007
1008
         solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
      distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1009
1010
        ss.str("");
1011
               "========\n";
        ss <<
        ss << "======= EPNP
1012
         ss << "rvec: " << "\n";
1013
        ss << Rvec << "\n";
ss << "tvec: " << "\n";
1014
1015
        ss << Tvec << "\n";
1016
1017
1018
        projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
      distCoeffs, list_points2dProjected);
1019
        for (int i = 0; i < numberMarkers; i++)</pre>
1020
              \mbox{circle(cFrame, Point(list\_points2dProjected[i].x, list\_points2dProjected[i].y), 3, Scalar(255, 0, 0) } \\
1021
     ), 3);
1022
1023
         QPixmap QPFrame;
1024
         QPFrame = Mat2QPixmap(cFrame);
1025
         commObj.changeImage(QPFrame);
1026
         OCoreApplication::processEvents();
1027
        commObj.addLog(QString::fromStdString(ss.str()));
1028
         if (numberMarkers == 4)
1029
1030
              methodPNP = 2; //! < 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //! < not used
             Rvec = cv::Mat::zeros(3, 1, CV_64F);
Tvec = cv::Mat::zeros(3, 1, CV_64F);
solvePnP(list_points3d, list_points2dProjected,
1031
1032
1033
```

```
cameraMatrix, distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1034
1035
                             ss.str("");
                            ss << "=====\\n";
1036
                            ss << "=======\n";
1037
                            ss << "rvec: " << "\n";
1038
                          ss << Rvec << "\n";
ss << "tvec: " << "\n";
1039
1040
                          ss << Tvec << "\n";
1041
1042
                           projectPoints(list_points3d, Rvec, Tvec, cameraMatrix,
1043
            distCoeffs, list_points2dProjected);
                for (int i = 0; i < numberMarkers; i++)</pre>
1044
1045
1046
                                      \verb|circle(cFrame, Point(list_points2dProjected[i].x, list_points2dProjected[i].y), 3, Scalar(255, and all of the context of t
             0, 0), 3);
1047
                            double projectionError = norm(list_points2dProjected, list_points2d);
putText(cFrame, "Testing Algorithms Finished", cv::Point(5, 420), 1, 1, cv::Scalar(255, 255, 255));
1048
1049
                            drawPositionText(cFrame, position, eulerAngles, projectionError)
1050
1051
1052
                           QPixmap QPFrame;
1053
                            OPFrame = Mat2OPixmap(cFrame);
1054
                             commObj.changeImage(QPFrame);
1055
                             QCoreApplication::processEvents();
1056
                             commObj.addLog(QString::fromStdString(ss.str()));
1057
1058
1059
                    _methodPNP = 4; //! < 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //! < not used
                   Rvec = cv::Mat::zeros(3, 1, CV_64F);
1060
                   Tvec = cv::Mat::zeros(3, 1, CV_64F);
solvePnP(list_points3d, list_points2dProjected, cameraMatrix,
1061
             distCoeffs, Rvec, Tvec, useGuess, _methodPNP);
1063
1064
                   ss << "-----\n";
1065
                   ss << "========\\n";
1066
                   ss << "rvec: " << "\n";
                  ss << Rvec << "\n";
ss << "tvec: " << "\n";
1068
1069
                  ss << Tvec << "\n";
1070
1071
1072
                   commObj.addLog(QString::fromStdString(ss.str()));
1073
1074
                   Rvec = RvecOriginal;
1075
                   Tvec = TvecOriginal;
1076
1077 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.6.3 Variable Documentation

6.6.3.1 commObj

commObject commObj

class that handles the communication from main.cpp to the GUI

Now declare variables that are used across the main.cpp file. Basically almost every variable used is declared here.

Definition at line 68 of file main.cpp.

6.6.3.2 invertZ

int invertZ

dummy variable to invert Z direction on request

Definition at line 75 of file main.cpp.

6.6.3.3 IPAdressObject

QHostAddress IPAdressObject

IPv4 adress of receiver 1.

Definition at line 131 of file main.cpp.

6.6.3.4 IPAdressSafety QHostAddress IPAdressSafety IPv4 adress of safety receiver. Definition at line 132 of file main.cpp. 6.6.3.5 IPAdressSafety2 QHostAddress IPAdressSafety2 IPv4 adress of receiver 2. Definition at line 133 of file main.cpp. 6.6.3.6 methodPNP int methodPNP solvePNP algorithm 0 = iterative 1 = EPNP 2 = P3P 4 = UPNP //!< 4 and 1 are the same and not implemented correctly by OpenCV Definition at line 105 of file main.cpp. 6.6.3.7 portObject int portObject Port of receiver 1. Definition at line 134 of file main.cpp. 6.6.3.8 portSafety int portSafety Port of the safety receiver. Definition at line 135 of file main.cpp.

File Documentation

6.6.3.9 portSafety2

6.6.3.9 portSafety2 int portSafety2 Port of receiver 2. Definition at line 136 of file main.cpp. 6.6.3.10 safety2Enable bool safety2Enable is the second receiver enabled Definition at line 71 of file main.cpp. 6.6.3.11 safetyAngle int safetyAngle bank and pitch angle protection in degrees Definition at line 73 of file main.cpp. 6.6.3.12 safetyBoxLength double safetyBoxLength

length of the safety area cube in meters

Definition at line 72 of file main.cpp.

6.6.3.13 safetyEnable

bool safetyEnable

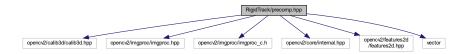
is the safety feature enabled

Definition at line 70 of file main.cpp.

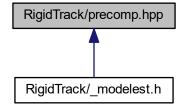
6.7 RigidTrack/precomp.hpp File Reference

```
#include "opencv2/calib3d/calib3d.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/imgproc/imgproc_c.h"
#include "opencv2/core/internal.hpp"
#include "opencv2/features2d/features2d.hpp"
#include <vector>
```

Include dependency graph for precomp.hpp:



This graph shows which files directly or indirectly include this file:



Macros

#define GET_OPTIMIZED(func) (func)

6.7.1 Macro Definition Documentation

6.7.1.1 GET_OPTIMIZED

Definition at line 59 of file precomp.hpp.

6.8 RigidTrack/resource.h File Reference

Macros

#define IDI_ICON1 101
 /<{{NO_DEPENDENCIES}} /< Von Microsoft Visual C++ generierte Includedatei. /< Verwendet durch RigidTrack.rc
 /<

6.8.1 Macro Definition Documentation

6.8.1.1 IDI_ICON1

```
#define IDI_ICON1 101
```

 $/\!\!<\!\!\{\{NO_DEPENDENCIES\}\!\}\ /\!<\ Von\ Microsoft\ Visual\ C++\ generierte\ Includedatei.\ /\!<\ Verwendet\ durch\ Rigid \leftrightarrow Track.rc\ /\!<$

Definition at line 5 of file resource.h.

6.9 RigidTrack/RigidTrack.cpp File Reference

Rigid Track GUI source that contains functions for GUI events.

```
#include "RigidTrack.h"
#include <QProcess>
#include <QdesktopServices>
#include <QDir>
#include <QMessageBox>
#include 'QUrl>
#include "main.h"
#include "communication.h"
#include <exception>
```

Include dependency graph for RigidTrack.cpp:



6.9.1 Detailed Description

Rigid Track GUI source that contains functions for GUI events.

Author

Florian J.T. Wachter

Version

1.0

Date

April, 8th 2017

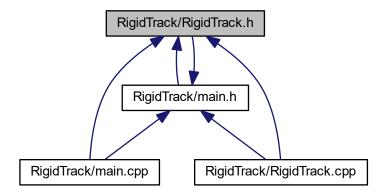
6.10 RigidTrack/RigidTrack.h File Reference

Rigid Track GUI source header with Qt Signals and Slots.

```
#include <QtWidgets/QMainWindow>
#include "ui_RigidTrack.h"
#include <qpixmap.h>
#include "main.h"
#include "communication.h"
Include dependency graph for RigidTrack.h:
```



This graph shows which files directly or indirectly include this file:



Classes

• class RigidTrack

6.10.1 Detailed Description

Rigid Track GUI source header with Qt Signals and Slots.

Author

Florian J.T. Wachter

Version

1.0

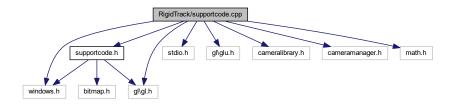
Date

April, 8th 2017

6.11 RigidTrack/supportcode.cpp File Reference

```
#include <windows.h>
#include <stdio.h>
#include <gl\gl.h>
#include <gl\glu.h>
#include "cameralibrary.h"
#include "cameramanager.h"
#include "math.h"
#include "supportcode.h"
```

Include dependency graph for supportcode.cpp:



Functions

- int LoadGLTextures ()
 - /< Permanent Rendering Context
- GLvoid ReSizeGLScene (GLsizei width, GLsizei height)
 - /< Resize And Initialize The GL Window
- int InitGL (GLvoid)
 - /< All Setup For OpenGL Goes Here
- int DrawGLScene (Surface *surf, int width, int height)
- LRESULT CALLBACK WndProc (HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM IParam)
 - /< Additional Message Information
- GLvoid CloseWindow (GLvoid)
 - /< Properly Kill The Window
- BOOL CreateAppWindow (const char *title, int width, int height, int bits, bool fullscreenflag)
- int main (int argc, char *argv[])
 - main initialises the GUI and values for the marker position etc
- int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmd
 — Show)
 - /< Window Show State
- bool FullscreenToggle ()
- bool PumpMessages ()
- LRESULT CALLBACK CBTHookProc (int nCode, WPARAM wParam, LPARAM IParam)
- VOID CALLBACK TimerProc (HWND hWnd, UINT uMsg, UINT idEvent, DWORD dwTime)
- bool PopWaitingDialog ()

Variables

· int gWindowWidth

- · int gWindowHeight
- bool gFullscreen = FALSE

/< Window Active Flag Set To TRUE By Default

bool gActive = TRUE

/< Array Used For Scanning Keyboard

• bool keys [256]

/< Private GDI Device Context

- HDC hDC =NULL
- GLuint texture [1]

/< Fullscreen Flag Set To Fullscreen Mode By Default

• int gSoftwareDecimate = 0

/< Storage For One Texture (NEW)

HWND hWnd =NULL

/< Private GDI Device Context

HINSTANCE hInstance

/< Holds Our Window Handle

HGLRC hRC =NULL

/< Holds The Instance Of The Application

- · int windowWidth
- · int windowHeight
- const char * windowName
- HHOOK hHook = NULL

/< == Code to pop a simple dialog for 'waiting for cameras' using a message box and /< == no resources required for this sample application.

6.11.1 Function Documentation

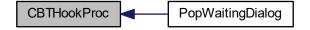
6.11.1.1 CBTHookProc()

```
LRESULT CALLBACK CBTHookProc (
int nCode,
WPARAM wParam,
LPARAM 1Param)
```

Definition at line 575 of file supportcode.cpp.

```
576 {
577
578
        if (nCode < 0)
            return CallNextHookEx(hHook, nCode, wParam, lParam);
579
        if (nCode == HCBT_ACTIVATE)
580
581
582
            HWND hWnd = reinterpret_cast<HWND>(wParam);
583
            SetWindowText(GetDlgItem(hWnd, IDOK), TEXT("Cancel"));
584
            return 0;
585
586
587
        return CallNextHookEx(hHook, nCode, wParam, 1Param);
588 }
```

Here is the caller graph for this function:



6.11.1.2 CloseWindow()

```
GLvoid CloseWindow (
GLvoid )
```

/< Properly Kill The Window

Definition at line 192 of file supportcode.cpp.

```
193 {
194
        if (gFullscreen)
                                                             //!/< Are We In Fullscreen Mode?
195
196
            ChangeDisplaySettings(NULL,0);
                                                             //!/< If So Switch Back To The Desktop
197
            ShowCursor(TRUE);
                                                             //!/< Show Mouse Pointer
198
       }
199
200
        if (hRC)
                                                         //!/< Do We Have A Rendering Context?
201
202
            if (!wglMakeCurrent(NULL, NULL))
                                                            //!/< Are We Able To Release The DC And RC
       Contexts?
203
                MessageBox(NULL,L"Release Of DC And RC Failed.",L"SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
204
205
206
207
            if (!wglDeleteContext(hRC))
                                                             //!/< Are We Able To Delete The RC?
208
            {
209
                MessageBox(NULL, L"Release Rendering Context Failed.", L"SHUTDOWN ERROR", MB_OK |
      MB_ICONINFORMATION);
210
211
            hRC=NULL;
                                                             //!/< Set RC To NULL
212
       }
213
214
        if (hDC && !ReleaseDC(hWnd,hDC))
                                                           //!/< Are We Able To Release The DC
215
216
            MessageBox(NULL, L"Release Device Context Failed.", L"SHUTDOWN ERROR", MB_OK | MB_ICONINFORMATION);
                                                             //!/< Set DC To NULL
217
            hDC=NULL;
218
219
220
        if (hWnd && !DestroyWindow(hWnd))
                                                            //!/< Are We Able To Destroy The Window?
221
            MessageBox(NULL, L"Could Not Release hWnd.", L"SHUTDOWN ERROR", MB_OK | MB_ICONINFORMATION);
222
223
            hWnd=NULL;
                                                            //!/< Set hWnd To NULL
224
225
226
        if (!UnregisterClass(L"OpenGL", hInstance))
                                                           //!/< Are We Able To Unregister Class
227
228
            MessageBox(NULL, L"Could Not Unregister Class.", L"SHUTDOWN ERROR", MB_ICONINFORMATION);
229
            hInstance=NULL:
                                                                //!/< Set hInstance To NULL
230
231 }
```

Here is the caller graph for this function:



6.11.1.3 CreateAppWindow()

```
BOOL CreateAppWindow (

const char * title,

int width,

int height,

int bits,

bool fullscreenflag)
```

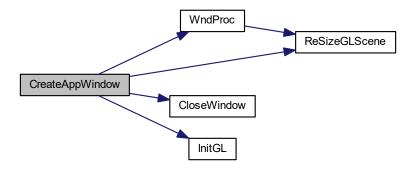
Definition at line 244 of file supportcode.cpp.

```
246
        windowWidth = width;
        windowHeight = height;
2.47
248
        windowName = title;
249
250
        GLuint
                   PixelFormat;
                                            //!/< Holds The Results After Searching For A Match
251
        WNDCLASS
                    wc;
                                            //!/< Windows Class Structure
252
        DWORD
                    dwExStyle;
                                            //!/< Window Extended Style
253
        DWORD
                    dwStyle;
                                            //!/< Window Style
254
                                            //!/< Grabs Rectangle Upper Left / Lower Right Values
        RECT
                   WindowRect;
255
                                            //!/< Set Left Value To 0
        WindowRect.left=(long)0;
256
        WindowRect.right=(long)width;
                                            //!/< Set Right Value To Requested Width
257
        WindowRect.top=(long)0;
                                            //!/< Set Top Value To 0
258
        WindowRect.bottom=(long)height;
                                            //!/< Set Bottom Value To Requested Height
259
260
        gFullscreen=fullscreenflag;
                                             //!/< Set The Global Fullscreen Flag
261
262
        HICON hIcon = (HICON)LoadImage(0,IDI_WINLOGO,IMAGE_ICON,0,0,LR_SHARED);
263
264
                           = GetModuleHandle(NULL);
                                                                    //!/< Grab An Instance For Our
       Window
                           = CS_HREDRAW | CS_VREDRAW | CS_OWNDC; //!/< Redraw On Size, And Own DC For
265
        wc.style
       Window.
266
       wc.lpfnWndProc
                           = (WNDPROC) WndProc;
                                                                  //!/< WndProc Handles Messages
        wc.cbClsExtra
267
                            = 0;
                                                                    //!/< No Extra Window Data
268
        wc.cbWndExtra
                            = 0;
                                                                     //!/< No Extra Window Data
269
        wc.hInstance
                            = hInstance;
                                                                    //!/< Set The Instance
                           = 0;
270
        wc.hIcon
271
                           = LoadCursor(NULL, IDC_ARROW);
        wc.hCursor
                                                                    //!/< Load The Arrow Pointer
272
        wc.hbrBackground
                                                                     //!/< No Background Required For GL
                            = NULL;
273
                            = NULL;
                                                                     //!/< We Don't Want A Menu
        wc.lpszMenuName
                                                                         //!/< Set The Class Name
274
        wc.lpszClassName
                            = L"OpenGL";
275
276
        if (!RegisterClass(&wc))
                                                                     //!/< Attempt To Register The Window Class
277
            MessageBox(NULL, L"Failed To Register The Window Class.", L"ERROR", MB_OK|MB_ICONEXCLAMATION);
278
            return FALSE;
                                                                     //!/< Return FALSE
280
281
282
        if (gFullscreen)
                                                                      //!/< Attempt Fullscreen Mode?
283
284
            DEVMODE dmScreenSettings:
                                                                     //!/< Device Mode
285
            memset(&dmScreenSettings, 0, sizeof(dmScreenSettings));
                                                                     //!/< Makes Sure Memory's Cleared
286
            dmScreenSettings.dmSize=sizeof(dmScreenSettings);
                                                                     //!/< Size Of The Devmode Structure
```

```
287
             dmScreenSettings.dmPelsWidth
                                                                           //!/< Selected Screen Width
                                                                           //!/< Selected Screen Height
//!/< Selected Bits Per Pixel
             dmScreenSettings.dmPelsHeight = height;
dmScreenSettings.dmBitsPerPel = bits;
288
289
             {\tt dmScreenSettings.dmFields=DM\_BITSPERPEL|DM\_PELSWIDTH|DM\_PELSHEIGHT;}
290
291
             //!/< Try To Set Selected Mode And Get Results. NOTE: CDS_FULLSCREEN Gets Rid Of Start Bar.
292
             if (ChangeDisplaySettings(&dmScreenSettings,CDS_FULLSCREEN)!=DISP_CHANGE_SUCCESSFUL)
293
294
295
                  //!/< If The Mode Fails, Offer Two Options. Quit Or Use Windowed Mode.
       if (MessageBox(NULL, L"The Requested Fullscreen Mode Is Not Supported By\nYour Video Card. Use
Windowed Mode Instead?",L"NeHe GL",MB_YESNO|MB_ICONEXCLAMATION)==IDYES)
296
297
298
                                                 //!/< Windowed Mode Selected. Fullscreen = FALSE
                      gFullscreen=FALSE;
299
300
                 else
301
                      //!/< Pop Up A Message Box Letting User Know The Program Is Closing.
302
                      MessageBox(NULL, L"Program Will Now Close.", L" RROR", MB_OK | MB_ICONSTOP);
303
                                                                           //!/< Return FALSE
304
                      return FALSE:
305
                 }
306
             }
307
        }
308
         if (gFullscreen)
                                                                            //!/< Are We Still In
309
       Fullscreen Mode?
310
        {
311
             dwExStyle=WS_EX_APPWINDOW;
                                                                           //!/< Window Extended Style
                                                                           //!/< Windows Style
312
             dwStyle=WS_POPUP;
313
             ShowCursor(FALSE);
                                                                           //!/< Hide Mouse Pointer
314
315
        else
316
        {
317
             dwExStyle=WS_EX_APPWINDOW | WS_EX_WINDOWEDGE;
                                                                           //!/< Window Extended Style
318
             dwStyle=WS_OVERLAPPEDWINDOW;
                                                                           //!/< Windows Style
319
320
321
        AdjustWindowRectEx(&WindowRect, dwStyle, FALSE, dwExStyle);
                                                                              //!/< Adjust Window To True Requested
322
323
         //!/< Create The Window
324
        if (!(hWnd=CreateWindowEx( dwExStyle,
                                                                               //!/< Extended Style For The Window
             L" penGL",
                                                     //!/< Class Name
325
                                       L"title",
                                                                                    //!/< Window Title
326
                                                                                //!/< Defined Window Style
327
                                       dwStyle |
                                        WS_CLIPSIBLINGS |
                                                                                //!/< Required Window Style
328
329
                                        WS_CLIPCHILDREN,
                                                                                //!/< Required Window Style
330
                                        0, 0,
                                                                                //!/< Window Position
                                                                               //!/< Calculate Window Width
//!/< Calculate Window Height
                                       WindowRect.right-WindowRect.left,
331
                                        WindowRect.bottom-WindowRect.top,
332
333
                                                                                //!/< No Parent Window
                                       NULL.
                                                                                //!/< No Menu
334
                                       NULL,
335
                                        hInstance,
                                                                              //!/< Instance
336
                                       NULL)))
                                                                               //!/< Dont Pass Anything To WM_CREATE
337
                                                              //!/< Reset The Display
             CloseWindow();
338
             MessageBox(NULL,L" indow Creation Error.",L"Error", MB_OK|MB_ICONEXCLAMATION);
339
                                                             //!/< Return FALSE
340
             return FALSE:
341
342
        static PIXELFORMATDESCRIPTOR pfd=
343
                                                             //!/< pfd Tells Windows How We Want Things To Be
344
345
             sizeof (PIXELFORMATDESCRIPTOR),
                                                              //!/< Size Of This Pixel Format Descriptor
346
                                                              //!/< Version Number
             PFD_DRAW_TO_WINDOW |
PFD_SUPPORT_OPENGL |
                                                              //!/< Format Must Support Window
347
348
                                                              //!/< Format Must Support OpenGL
349
             PFD_DOUBLEBUFFER,
                                                              //!/< Must Support Double Buffering
                                                              //!/< Request An RGBA Format
//!/< Select Our Color Depth
350
             PFD TYPE RGBA,
351
             bits.
                                                              //!/< Color Bits Ignored
352
             0, 0, 0, 0, 0, 0,
                                                              //!/< No Alpha Buffer
353
             Ο,
354
                                                              //!/< Shift Bit Ignored
             0,
355
             0,
                                                              //!/< No Accumulation Buffer
356
             0, 0, 0, 0,
                                                              //!/< Accumulation Bits Ignored
357
                                                              //!/< 16Bit Z-Buffer (Depth Buffer)
             16.
358
                                                              //!/< No Stencil Buffer
             Ο,
                                                              //!/< No Auxiliary Buffer
359
360
             PFD_MAIN_PLANE,
                                                              //!/< Main Drawing Layer
361
                                                              //!/< Reserved
362
             0, 0, 0
                                                              //!/< Layer Masks Ignored
363
        };
364
365
         if (!(hDC=GetDC(hWnd)))
                                                              //!/< Did We Get A Device Context?
366
         {
             CloseWindow();
367
                                                              //!/< Reset The Display
             MessageBox(NULL,L" an't Create A GL Device Context.",L" RROR",MB_OK|MB_ICONEXCLAMATION);
368
369
             return FALSE:
                                                              //!/< Return FALSE
        }
370
```

```
372
      if (!(PixelFormat=ChoosePixelFormat(hDC,&pfd))) //!/< Did Windows Find A Matching Pixel Format?
373
         374
375
376
377
378
379
      if (!SetPixelFormat (hDC, PixelFormat, &pfd))
                                              //!/< Are We Able To Set The Pixel Format?
380
         381
382
                                              //!/< Return FALSE
383
         return FALSE;
384
385
386
      if (!(hRC=wglCreateContext(hDC)))
                                            //!/< Are We Able To Get A Rendering Context?
387
         388
389
390
         return FALSE;
                                             //!/< Return FALSE
391
392
      if(!wglMakeCurrent(hDC,hRC))
                                            //!/< Try To Activate The Rendering Context
393
394
395
          CloseWindow();
                                               //!/< Reset The Display
396
         MessageBox(NULL, L" an't Activate The GL Rendering Context.", L" RROR", MB_OK|MB_ICONEXCLAMATION);
397
         return FALSE;
                                              //!/< Return FALSE
398
399
                                              //!/< Show The Window
400
      ShowWindow(hWnd,SW SHOW);
      SetForegroundWindow(hWnd);
                                              //!/< Slightly Higher Priority
401
402
      SetFocus(hWnd);
                                               //!/< Sets Keyboard Focus To The Window
403
      ReSizeGLScene(width, height);
                                             //!/< Set Up Our Perspective GL Screen
404
405
      if (!InitGL())
                                                //!/< Initialize Our Newly Created GL Window
406
         CloseWindow();
MessageBox(NULL, L" nitialization Failed.", L" RROR",MB_OK|MB_ICONEXCLAMATION);
407
408
         return FALSE;
409
                                              //!/< Return FALSE
410
411
      return TRUE;
                                              //!/< Success
412
413 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

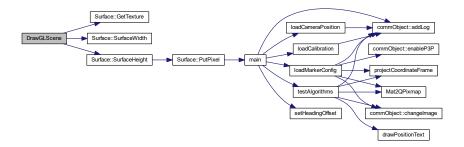


6.11.1.4 DrawGLScene()

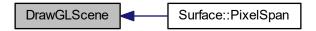
Definition at line 76 of file supportcode.cpp.

```
77 {
78
       if(surf==NULL)
79
           return true;
80
       static bool frameThrottler = true;
81
82
       frameThrottler=!frameThrottler;
83
       if(frameThrottler) //!/<== Only display every other frame in case VSYNC is enabled. Otherwise return true; //!/<== application would get behind
84
85
86
87
       int pixelWidth = width;
       int pixelHeight= height;
89
       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //!/< Clear The Screen And The Depth Buffer
90
91
       glLoadIdentity();
                                                               //!/< Reset The View
92
93
       static GLuint ff = 0;
95
       int tex = surf->GetTexture();
       if(tex==0)
96
97
           tex=ff;
98
       else
           ff=tex;
99
100
        if(tex==0)
101
            return true;
102
103
        glBindTexture(GL_TEXTURE_2D, tex);
104
105
        glBegin (GL QUADS);
            glTexCoord2f(0.0f, 0.0f); glVertex3f(0,
106
                                                         0, 0);
107
             glTexCoord2f((GLfloat)pixelWidth/(GLfloat)surf->SurfaceWidth(), 0.0f); glVertex3f( (
      GLfloat)gWindowWidth, 0, 0);
108
            glTexCoord2f((GLfloat)pixelWidth/(GLfloat)surf->SurfaceWidth(), (GLfloat)pixelHeight/(
      GLfloat)surf->SurfaceHeight()); glVertex3f( (GLfloat)gWindowWidth, (GLfloat)
      gWindowHeight, 0);
            glTexCoord2f(0.0f, (GLfloat)pixelHeight/(GLfloat)surf->SurfaceHeight()); glVertex3f( 0
109
      , (GLfloat) gWindowHeight, 0);
110
111
        SwapBuffers(hDC);
                                               //!/< Swap Buffers (Double Buffering)
112
113
114
                                                                //!/< Keep Going
        return true;
115 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



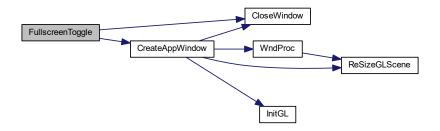
6.11.1.5 FullscreenToggle()

```
bool FullscreenToggle ( )
```

Definition at line 536 of file supportcode.cpp.

```
537 {
538
         keys[VK_F1]=FALSE;
                                                   //!/< If So Make Key FALSE
                                                 //!/< Kill Our Current Window
//!/< Toggle Fullscreen / Windowed Mode
         CloseWindow();
gFullscreen=!gFullscreen;
539
540
541
         if (!CreateAppWindow(windowName, windowWidth,
542
      windowHeight, 32, gFullscreen))
543
544
              MessageBox(0, L"Unable to toggle to full screen", L"Error", MB_OK);
545
546
547
548
         return true;
549 }
```

Here is the call graph for this function:



6.11.1.6 InitGL()

/< All Setup For OpenGL Goes Here

Definition at line 62 of file supportcode.cpp.

```
63 {
         glEnable(GL_TEXTURE_2D);
                                                                               //!/< Enable Texture Mapping ( NEW ) //!/< Enable Smooth Shading
64
65
         {\tt glShadeModel} \; ({\tt GL\_SMOOTH}) \; ; \\
                                                                               //!/< Black Background
//!/< Depth Buffer Setup
        glClearColor(0.0f, 0.0f, 0.0f, 0.5f);
glClearDepth(1.0f);
66
68
         glEnable(GL_DEPTH_TEST);
                                                                               //!/< Enables Depth Testing
         glDepthFunc(GL_LEQUAL);
                                                                               //!/< The Type Of Depth Testing To Do
70
         glBlendFunc(GL_SRC_ALPHA,GL_ONE_MINUS_SRC_ALPHA);
         glEnable(GL_BLEND);
glHint(GL_PERSPECTIVE_CORRECTION_HINT, GL_NICEST);
71
72
                                                                               //!/< Really Nice Perspective Calculations //!/< Initialization Went OK
73
         return TRUE;
```

Here is the caller graph for this function:



6.11.1.7 LoadGLTextures()

```
int LoadGLTextures ( )
```

/< Permanent Rendering Context

/< Load Bitmaps And Convert To Textures

Definition at line 37 of file supportcode.cpp.

6.11.1.8 main()

main initialises the GUI and values for the marker position etc

First the GUI is set up with Signals and Slots, see Qt docu for how that works. Then some variables are initialized with arbitrary values. At last calibration and marker configuration etc. are loaded from xml files.

Parameters

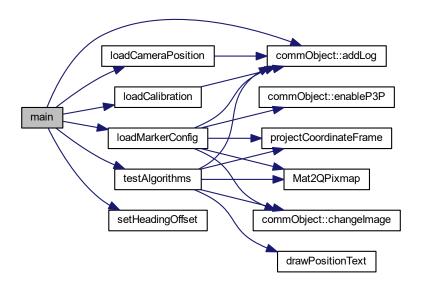
in	argc	is not used.
in	argv	is also not used.

Definition at line 156 of file main.cpp.

```
157 {
158
        QApplication a(argc, argv);
159
                    //!< show the GUI
160
161
        \ensuremath{//!} connect the Qt slots and signals for event handling
        QObject::connect(&commObj, SIGNAL(statusChanged(QString)), &w, SLOT(setStatus(QString)),
162
      Ot::DirectConnection);
163
        QObject::connect(&commObj, SIGNAL(imageChanged(QPixmap)), &w, SLOT(setImage(QPixmap)),
      Qt::DirectConnection);
164
        QObject::connect(&commObj, SIGNAL(logAdded(QString)), &w, SLOT(setLog(QString)),
      Qt::DirectConnection);
165
        QObject::connect(&commObj, SIGNAL(logCleared()), &w, SLOT(clearLog(QString)),
      Ot::DirectConnection);
        QObject::connect(&commObj, SIGNAL(P3Penabled(bool)), &w, SLOT(enableP3P(bool)),
166
      Qt::DirectConnection);
167
        QObject::connect(&commObj, SIGNAL(progressUpdated(int)), &w, SLOT(progressUpdate(int)),
      Qt::DirectConnection);
168
        commObj.addLog("RigidTrack Version:");
169
170
        commObj.addLog(QString::number(_MSC_FULL_VER));
171
        commObj.addLog("Built on:");
172
        commObj.addLog(QString(__DATE__));
173
174
        //!\ \ \text{initial guesses for position and rotation, important for Iterative Method!}
        Tvec.at<double>(0) = 45;
Tvec.at<double>(1) = 45;
175
176
177
        Tvec.at<double>(2) = 4500;
```

```
178
         Rvec.at<double>(0) = 0 * 3.141592653589 / 180.0;
        Rvec.at<double>(1) = 0 * 3.141592653589 / 180.0;
Rvec.at<double>(2) = -45 * 3.141592653589 / 180.0;
179
180
181
182
        //! Points that make up the marker CoSy axis system, hence one line in each axis direction
coordinateFrame = std::vector<Point3d>(4);
183
        coordinateFrameProjected = std::vector<Point2d>(4);
184
185
         coordinateFrame[0] = cv::Point3d(0, 0, 0);
186
         coordinateFrame[1] = cv::Point3d(300, 0, 0);
         coordinateFrame[2] = cv::Point3d(0, 300, 0);
187
        coordinateFrame[3] = cv::Point3d(0, 0, 300);
188
189
        position[0] = 1.1234;
position[1] = 1.2345;
                                     //!< set position initial values
//!< set position initial values</pre>
190
191
192
        position[2] = 1.3456;
                                     //!< set position initial values
193
        velocity[0] = 0.123;
velocity[1] = 0.234;
velocity[2] = 0.345;
                                   //!< set velocity initial values
//!< set velocity initial values
//!< set velocity initial values</pre>
194
195
196
197
        198
199
200
201
202
         setHeadingOffset(0.0); //!< set the heading offset to 0</pre>
203
204
         ss.precision(4); //!< outputs in the log etc are limited to 3 decimal values
205
        206
207
208
209
        testAlgorithms(); //!< test the algorithms and their accuracy
210
211
         return a.exec();
212 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



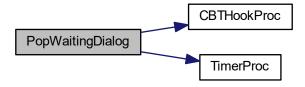
6.11.1.9 PopWaitingDialog()

```
bool PopWaitingDialog ( )
```

Definition at line 615 of file supportcode.cpp.

```
616 {
617
         //!/<== hook in so we can create a message box that has only a 'Cancel' button ==--
618
619
         \verb|hHook| = SetWindowsHookEx(WH\_CBT, reinterpret\_cast<HOOKPROC>(&
       CBTHookProc), NULL, GetCurrentThreadId());
620
         UINT_PTR nTimer = SetTimer(0, 100, 3000,(TIMERPROC) TimerProc);
int iResult = MessageBox( 0, L"waiting for connected cameras...", L"Camera Initialization", MB_OK );
621
622
623
624
          if(iResult == IDOK)
625
              //!/<== user has clicked the cancel button ==--
UnhookWindowsHookEx(hHook);</pre>
626
627
628
              return false;
630
         KillTimer(0, nTimer);
631
632
633
         return true;
634 }
```

Here is the call graph for this function:



6.11.1.10 PumpMessages()

```
bool PumpMessages ( )
```

Definition at line 551 of file supportcode.cpp.

```
552 {
553
        MSG msg;
554
555
        if (PeekMessage(&msg,NULL,0,0,PM_REMOVE))
                                                     //!/< Is There A Message Waiting?
556
557
            if (msg.message==WM_QUIT)
                                                     //!/< Have We Received A Quit Message?
558
                return false;
                                                     //!/< If Not, Deal With Window Messages
559
            else
560
                                                     //!/< Translate The Message
561
                TranslateMessage(&msg);
562
                DispatchMessage(&msg);
                                                     //!/< Dispatch The Message
563
564
        }
565
566
        return true;
567 }
```

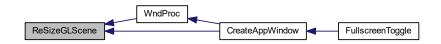
6.11.1.11 ReSizeGLScene()

/< Resize And Initialize The GL Window

Definition at line 42 of file supportcode.cpp.

```
43 {
44
        if (height==0)
                                                                      //!/< Prevent A Divide By Zero By
45
46
             height=1;
                                                                      //!/< Making Height Equal One
47
48
49
        glViewport(0,0,width,height);
                                                                      //!/< Reset The Current Viewport
50
                                                                      //!/< Select The Projection Matrix
//!/< Reset The Projection Matrix
51
        glMatrixMode(GL_PROJECTION);
52
        glLoadIdentity();
53
        gWindowWidth = width;
54
55
        gWindowHeight = height;
57
        glOrtho(0,gWindowWidth,gWindowHeight,0,100,-100);
                                                                      //!/< Select The Modelview Matrix
//!/< Reset The Modelview Matrix
58
        glMatrixMode(GL_MODELVIEW);
59
        glLoadIdentity();
60 }
```

Here is the caller graph for this function:



6.11.1.12 TimerProc()

```
VOID CALLBACK TimerProc (

HWND hWnd,

UINT uMsg,

UINT idEvent,

DWORD dwTime)
```

Definition at line 590 of file supportcode.cpp.

```
591 {
592
        CameraLibrary::CameraList list;
594
        bool found = false;
595
596
        for( int i=0; i<list.Count(); i++ )</pre>
597
598
            if( list[i].State() == CameraLibrary:: Initialized )
599
600
                found = true;
601
        }
602
603
604
        if (found==true)
605
606
            HWND hWndActive = GetActiveWindow();
607
608
            if( hWndActive!=0 )
609
                SendMessage(hWndActive, WM_COMMAND, IDCANCEL, 0);
610
611
        }
613 }
```

Here is the caller graph for this function:



6.11.1.13 WinMain()

/< Window Show State

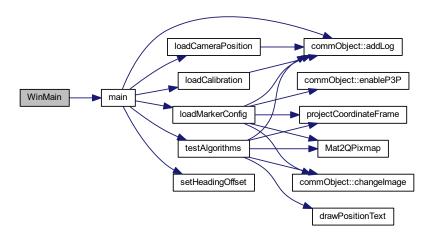
Parameters

hPrevInstance	/< Instance
_lpCmdLine	/< Previous Instance
Generated Syn Boxygen	/< Command Line Parameters

Definition at line 528 of file supportcode.cpp.

```
532 {
533     return main(0,0);
534 }
```

Here is the call graph for this function:



6.11.1.14 WndProc()

```
LRESULT CALLBACK WndProc (

HWND hWnd,

UINT uMsg,

WPARAM wParam,

LPARAM 1Param)
```

/< Additional Message Information

Parameters

uMsg	/< Handle For This Window	
wParam	/< Message For This Window	
IParam	/< Additional Message Information	

Definition at line 118 of file supportcode.cpp.

```
128
129
                    gActive=TRUE;
                                                           //!/< Program Is Active
130
131
                else
132
                {
                    gActive=FALSE;
133
                                                           //!/< Program Is No Longer Active
134
135
136
                return 0;
                                                          //!/< Return To The Message Loop
137
            }
138
            case WM_POWERBROADCAST:
139
140
                if(wParam == PBT_APMSUSPEND)
141
142
                     CameraLibrary::CameraManager::X().PrepareForSuspend();
143
                if(wParam == PBT APMRESUMEAUTOMATIC)
144
145
                {
146
                    CameraLibrary::CameraManager::X().ResumeFromSuspend();
147
148
149
            case WM_SYSCOMMAND:
                                                          //!/< Intercept System Commands
150
151
152
                                                          //!/< Check System Calls
                switch (wParam)
153
154
                     case SC_SCREENSAVE:
                                                          //!/< Screensaver Trying To Start?
155
                    case SC_MONITORPOWER:
                                                          //!/< Monitor Trying To Enter Powersave?
156
                    return 0;
                                                          //!/< Prevent From Happening
157
158
                                                          //!/< Exit
                break:
159
            }
160
161
            case WM_CLOSE:
                                                          //!/< Did We Receive A Close Message?
162
                PostQuitMessage(0);
                                                          //!/< Send A Quit Message
163
                                                          //!/< Jump Back
164
                return 0;
165
            }
166
167
            case WM_KEYDOWN:
                                                          //!/< Is A Key Being Held Down?
168
                                                          //\,!\,/\!< If So, Mark It As TRUE
169
                keys[wParam] = TRUE;
170
                                                          //!/< Jump Back
                return 0;
171
            }
172
173
            case WM_KEYUP:
                                                          //!/< Has A Key Been Released?
174
                keys[wParam] = FALSE;
                                                          //!/< If So, Mark It As FALSE
175
                                                          //!/< Jump Back
176
                return 0;
177
178
            case WM_MOVE:
179
                                                          //!/< Jump Back
180
            case WM_PAINT:
181
                                                          //!/< Resize The OpenGL Window
182
            case WM_SIZE:
183
            {
184
                ReSizeGLScene (LOWORD (1Param), HIWORD (1Param)); //!/< LoWord=Width, HiWord=Height
185
                                                          //!/< Jump Back
                return 0;
186
            }
187
        }
188
        //!/< Pass All Unhandled Messages To DefWindowProc
189
190
        return DefWindowProc(hWnd,uMsg,wParam,lParam);
191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.11.2 Variable Documentation

6.11.2.1 gActive

bool gActive = TRUE

/< Array Used For Scanning Keyboard

Definition at line 26 of file supportcode.cpp.

6.11.2.2 gFullscreen

bool gFullscreen = FALSE

/< Window Active Flag Set To TRUE By Default

Definition at line 27 of file supportcode.cpp.

6.11.2.3 gSoftwareDecimate

int gSoftwareDecimate = 0

/< Storage For One Texture (NEW)

Definition at line 29 of file supportcode.cpp.

6.11.2.4 gWindowHeight

int gWindowHeight

Definition at line 19 of file supportcode.cpp.

6.11.2.5 gWindowWidth

int gWindowWidth

/<-----

- /<== This is boiler-plate code for bringing up the application's window and /<== initializing OpenGL, and an OpenGL surface class for rendering the camera /<== image as a quad using the 3D hardware.

/<-----

_

/< Header File For Windows /< Header File For Standard Input/Output /< Header File For The OpenGL32 Library /< Header File For The GLu32 Library

Definition at line 18 of file supportcode.cpp.

6.11.2.6 hDC

HDC hDC =NULL

Definition at line 24 of file supportcode.cpp.

6.11.2.7 hHook

HHOOK hHook = NULL

/<== Code to pop a simple dialog for 'waiting for cameras' using a message box and /<== no resources required for this sample application.

Definition at line 573 of file supportcode.cpp.

6.11.2.8 hInstance

HINSTANCE hInstance

/< Holds Our Window Handle

Definition at line 34 of file supportcode.cpp.

6.11.2.9 hRC

HGLRC hRC =NULL

/< Holds The Instance Of The Application

Definition at line 35 of file supportcode.cpp.

6.11.2.10 hWnd HWND hWnd =NULL /< Private GDI Device Context Definition at line 33 of file supportcode.cpp. 6.11.2.11 keys bool keys /< Private GDI Device Context Definition at line 25 of file supportcode.cpp. 6.11.2.12 texture GLuint texture[1] /< Fullscreen Flag Set To Fullscreen Mode By Default Definition at line 28 of file supportcode.cpp. 6.11.2.13 windowHeight int windowHeight Definition at line 241 of file supportcode.cpp. 6.11.2.14 windowName const char* windowName Definition at line 242 of file supportcode.cpp. 6.11.2.15 windowWidth

int windowWidth

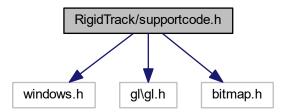
Definition at line 240 of file supportcode.cpp.

Generated by Doxygen

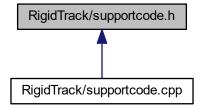
6.12 RigidTrack/supportcode.h File Reference

```
#include <windows.h>
#include <gl\gl.h>
#include "bitmap.h"
```

Include dependency graph for supportcode.h:



This graph shows which files directly or indirectly include this file:



Classes

• class Surface

Macros

- #define WIN32_LEAN_AND_MEAN
- #define BYTESPERPIXEL 4
- #define RGBA(x, y, z, a) ((a<<24)|(x<<16)|(y<<8)|z)

Functions

• LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM)

```
/< Additional Message Information
```

- BOOL CreateAppWindow (const char *title, int width, int height, int bits, bool fullscreenflag)
- GLvoid CloseWindow (GLvoid)

```
/< Properly Kill The Window
```

- bool PumpMessages ()
- bool FullscreenToggle ()
- bool PopWaitingDialog ()
- int DrawGLScene (Surface *surf, int width, int height)

Variables

- HDC hDC
- bool keys [256]
- bool gActive
- bool gFullscreen

6.12.1 Macro Definition Documentation

```
6.12.1.1 BYTESPERPIXEL
```

```
#define BYTESPERPIXEL 4
```

Definition at line 17 of file supportcode.h.

6.12.1.2 RGBA

Definition at line 18 of file supportcode.h.

6.12.1.3 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

Definition at line 3 of file supportcode.h.

6.12.2 Function Documentation

6.12.2.1 CloseWindow()

/< Properly Kill The Window

Definition at line 192 of file supportcode.cpp.

```
193 {
                                                             //!/< Are We In Fullscreen Mode?
194
        if (gFullscreen)
195
196
            ChangeDisplaySettings(NULL,0);
                                                             //!/< If So Switch Back To The Desktop
197
            ShowCursor(TRUE);
                                                             //!/< Show Mouse Pointer
198
199
200
        if (hRC)
                                                          //!/< Do We Have A Rendering Context?
201
            if (!wglMakeCurrent(NULL, NULL))
                                                             //!/< Are We Able To Release The DC And RC
202
       Contexts?
203
           {
204
                MessageBox(NULL,L"Release Of DC And RC Failed.",L"SHUTDOWN ERROR",MB_OK | MB_ICONINFORMATION);
205
206
207
            if (!wglDeleteContext(hRC))
                                                             //!/< Are We Able To Delete The RC?
208
                MessageBox(NULL, L"Release Rendering Context Failed.", L"SHUTDOWN ERROR", MB_OK |
209
      MB_ICONINFORMATION);
210
            hRC=NULL;
211
                                                              //!/< Set RC To NULL
212
       }
213
214
        if (hDC && !ReleaseDC(hWnd,hDC))
                                                           //!/< Are We Able To Release The DC
215
216
            MessageBox(NULL, L"Release Device Context Failed.", L"SHUTDOWN ERROR", MB_OK | MB_ICONINFORMATION);
217
            hDC=NULL;
                                                              //!/< Set DC To NULL
218
219
220
        if (hWnd && !DestroyWindow(hWnd))
                                                            //!/< Are We Able To Destroy The Window?
221
222
            MessageBox(NULL, L"Could Not Release hWnd.", L"SHUTDOWN ERROR", MB_OK | MB_ICONINFORMATION);
223
            hWnd=NULL;
                                                             //!/< Set hWnd To NULL
224
       }
225
226
        if (!UnregisterClass(L"OpenGL", hInstance))
                                                            //!/< Are We Able To Unregister Class
227
228
            MessageBox(NULL, L"Could Not Unregister Class.", L"SHUTDOWN ERROR", MB_OK | MB_ICONINFORMATION);
229
            hInstance=NULL;
                                                                //!/< Set hInstance To NULL
2.30
231 }
```

Here is the caller graph for this function:



6.12.2.2 CreateAppWindow()

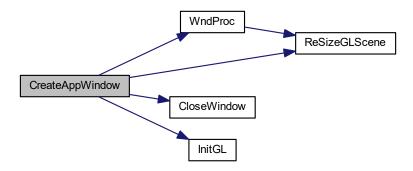
Definition at line 244 of file supportcode.cpp.

```
246
        windowWidth = width;
        windowHeight = height;
windowName = title;
247
248
249
250
                    PixelFormat;
                                              //!/< Holds The Results After Searching For A Match
                                              //!/< Windows Class Structure
251
        WNDCLASS
                     wc;
252
        DWORD
                     dwExStyle;
                                              //!/< Window Extended Style
253
        DWORD
                     dwStyle;
                                              //!/< Window Style
                                              //!/< Grabs Rectangle Upper Left / Lower Right Values
254
        RECT
                     WindowRect:
255
        WindowRect.left=(long)0;
                                              //!/< Set Left Value To 0
256
        WindowRect.right=(long)width;
                                              //!/< Set Right Value To Requested Width
257
        WindowRect.top=(long)0;
                                              //!/< Set Top Value To 0
258
        WindowRect.bottom=(long)height;
                                              //!/< Set Bottom Value To Requested Height
259
        gFullscreen=fullscreenflag;
                                               //!/< Set The Global Fullscreen Flag
260
261
262
        HICON hIcon = (HICON)LoadImage(0,IDI_WINLOGO,IMAGE_ICON,0,0,LR_SHARED);
263
        hInstance
                            = GetModuleHandle(NULL);
                                                                      //!/< Grab An Instance For Our
264
       Window
                             = CS_HREDRAW | CS_VREDRAW | CS_OWNDC; //!/< Redraw On Size, And Own DC For
265
        wc.style
       Window.
266
        wc.lpfnWndProc
                             = (WNDPROC) WndProc;
                                                                    //!/< WndProc Handles Messages
        wc.cbClsExtra
                             = 0;
                                                                       //!/< No Extra Window Data
267
                             = 0;
268
        wc.cbWndExtra
                                                                        //!/< No Extra Window Data
269
        wc.hInstance
                             = hInstance;
                                                                       //!/< Set The Instance
270
        wc.hIcon
                             = 0;
                             = LoadCursor(NULL, IDC_ARROW);
271
        wc.hCursor
                                                                       //!/< Load The Arrow Pointer
272
                             = NULL;
                                                                        //!/< No Background Required For GL
        wc.hbrBackground
                                                                        //!/< We Don't Want A Menu
273
        wc.lpszMenuName
                             = L"OpenGL";
274
        wc.lpszClassName
                                                                            //!/< Set The Class Name
275
276
        if (!RegisterClass(&wc))
                                                                        //!/< Attempt To Register The Window Class
277
278
            MessageBox(NULL, L"Failed To Register The Window Class.", L"ERROR", MB_OK|MB_ICONEXCLAMATION);
279
            return FALSE;
                                                                        //!/< Return FALSE
280
281
282
        if (gFullscreen)
                                                                         //!/< Attempt Fullscreen Mode?
283
284
            DEVMODE dmScreenSettings;
                                                                        //!/< Device Mode
285
            memset(&dmScreenSettings, 0, sizeof(dmScreenSettings));
                                                                        //!/< Makes Sure Memory's Cleared
             dmScreenSettings.dmSize=sizeof(dmScreenSettings);
                                                                        //!/< Size Of The Devmode Structure
286
287
             dmScreenSettings.dmPelsWidth
                                                                        //!/< Selected Screen Width
                                              = width;
                                             = height;
= bits;
288
             dmScreenSettings.dmPelsHeight
                                                                        //!/< Selected Screen Height
289
            dmScreenSettings.dmBitsPerPel
                                                                        //!/< Selected Bits Per Pixel
            dmScreenSettings.dmFields=DM_BITSPERPEL|DM_PELSWIDTH|DM_PELSHEIGHT;
290
291
292
             //!/< Try To Set Selected Mode And Get Results. NOTE: CDS_FULLSCREEN Gets Rid Of Start Bar.
293
             if (ChangeDisplaySettings(&dmScreenSettings,CDS_FULLSCREEN)!=DISP_CHANGE_SUCCESSFUL)
294
                 //!/< If The Mode Fails, Offer Two Options. Quit Or Use Windowed Mode.
if (MessageBox(NULL, L"The Requested Fullscreen Mode Is Not Supported By\nYour Video Card. Use
295
296
       Windowed Mode Instead?", L"NeHe GL", MB_YESNO | MB_ICONEXCLAMATION) ==IDYES)
                 {
298
                                               //!/< Windowed Mode Selected. Fullscreen = FALSE
299
300
                 else
301
                 {
                     //!/< Pop Up A Message Box Letting User Know The Program Is Closing.
302
                     MessageBox(NULL, L"Program Will Now Close.", L" RROR", MB_OK | MB_ICONSTOP);
303
304
                     return FALSE;
305
306
             }
307
        }
308
        if (gFullscreen)
                                                                         //!/< Are We Still In
309
       Fullscreen Mode?
```

```
310
            {
                    dwExStyle=WS_EX_APPWINDOW;
                                                                                                                //!/< Window Extended Style
311
                                                                                                                //!/< Windows Style
312
                   dwStyle=WS_POPUP;
313
                   ShowCursor(FALSE);
                                                                                                                //!/< Hide Mouse Pointer
314
315
            else
316
            {
317
                    dwExStyle=WS_EX_APPWINDOW | WS_EX_WINDOWEDGE;
                                                                                                                //!/< Window Extended Style
318
                   dwStyle=WS_OVERLAPPEDWINDOW;
                                                                                                                //!/< Windows Style
319
320
             AdjustWindowRectEx(&WindowRect, dwStvle, FALSE, dwExStvle):
321
                                                                                                                    //!/< Adjust Window To True Requested
           Size
322
323
             //!/< Create The Window
             if (!(hWnd=CreateWindowEx( dwExStyle,
    L" penGL",
324
                                                                                                                      //!/< Extended Style For The Window
                                                                              //!/< Class Name
325
326
                                                           L"title",
                                                                                                                             //!/< Window Title
327
                                                           dwStyle |
                                                                                                                      //!/< Defined Window Style
                                                                                                                      //!/< Required Window Style
328
                                                           WS_CLIPSIBLINGS |
329
                                                           WS_CLIPCHILDREN,
                                                                                                                       //!/< Required Window Style
                                                                                                                      //!/< Window Position
//!/< Calculate Window Width
330
                                                           WindowRect.right-WindowRect.left,
331
                                                                                                                      //!/< Calculate Window Height
332
                                                           WindowRect.bottom-WindowRect.top,
333
                                                           NULL,
                                                                                                                      //!/< No Parent Window
                                                                                                                       //!/< No Menu
334
                                                           NULL,
335
                                                           hInstance,
                                                                                                                     //!/< Instance
336
                                                           NULL)))
                                                                                                                      //!/< Dont Pass Anything To WM_CREATE
337
                                                                                             //!/< Reset The Display
338
                   CloseWindow();
339
                   MessageBox(NULL,L" indow Creation Error.",L"Error", MB_OK|MB_ICONEXCLAMATION);
340
                   return FALSE;
                                                                                            //!/< Return FALSE
341
342
343
             static PIXELFORMATDESCRIPTOR pfd=
                                                                                            //!/< pfd Tells Windows How We Want Things To Be
344
345
                   sizeof (PIXELFORMATDESCRIPTOR),
                                                                                            //!/< Size Of This Pixel Format Descriptor
                                                                                            //!/< Version Number
346
347
                   PFD_DRAW_TO_WINDOW |
                                                                                            //!/< Format Must Support Window
348
                   PFD_SUPPORT_OPENGL |
                                                                                            //!/< Format Must Support OpenGL
349
                   PFD DOUBLEBUFFER.
                                                                                            //!/< Must Support Double Buffering
                                                                                            //!/< Request An RGBA Format
350
                   PFD TYPE RGBA,
                                                                                            //!/< Select Our Color Depth
351
                   bits.
                                                                                            //!/< Color Bits Ignored
352
                    0, 0, 0, 0, 0, 0,
                                                                                            //!/< No Alpha Buffer
353
                    Ο,
354
                    0,
                                                                                            //!/< Shift Bit Ignored
355
                                                                                            //!/< No Accumulation Buffer
                    0,
                                                                                            //!/< Accumulation Bits Ignored
356
                    0, 0, 0, 0,
                                                                                            //!/< 16Bit Z-Buffer (Depth Buffer)
357
                    16.
                                                                                            //!/< No Stencil Buffer
358
                    0.
                                                                                            //!/< No Auxiliary Buffer
359
360
                    PFD_MAIN_PLANE,
                                                                                            //!/< Main Drawing Layer
361
                    Ο,
                                                                                            //!/< Reserved
362
                    0, 0, 0
                                                                                            //!/< Layer Masks Ignored
363
            };
364
365
             if (!(hDC=GetDC(hWnd)))
                                                                                            //!/< Did We Get A Device Context?
366
             {
                   367
368
                                                                                            //!/< Return FALSE
369
                   return FALSE;
370
371
372
             if (!(PixelFormat=ChoosePixelFormat(hDC,&pfd))) //!/< Did Windows Find A Matching Pixel Format?
373
                   CloseWindow();
//!/< Reset The Display
MessageBox(NULL, L" an't Find A Suitable PixelFormat.", L" RROR", MB_OK|MB_ICONEXCLAMATION);</pre>
374
375
376
                                                                                            //!/< Return FALSE
                    return FALSE;
377
             }
378
379
             if(!SetPixelFormat(hDC,PixelFormat,&pfd))
                                                                                             //!/< Are We Able To Set The Pixel Format?
380
                   381
382
383
                                                                                            //!/< Return FALSE
                   return FALSE;
384
385
386
             if (!(hRC=wglCreateContext(hDC)))
                                                                                        //!/< Are We Able To Get A Rendering Context?
387
                   388
389
390
                   return FALSE;
                                                                                            //!/< Return FALSE
391
392
393
             if(!wglMakeCurrent(hDC, hRC))
                                                                                        \begin{subarray}{ll} \end{subarray} // \end{subarray} // \end{subarray} \begin{subarray}{ll} \end{su
394
395
                   CloseWindow();
                                                                                             //!/< Reset The Display
```

```
MessageBox(NULL, L" an't Activate The GL Rendering Context.", L" RROR", MB_OK|MB_ICONEXCLAMATION);
397
                                                      //!/< Return FALSE
398
399
                                                     //!/< Show The Window
//!/< Slightly Higher Priority
//!/< Sets Keyboard Focus To The Window</pre>
       ShowWindow(hWnd,SW_SHOW);
400
       SetForegroundWindow(hWnd);
401
402
       SetFocus(hWnd);
403
       ReSizeGLScene(width, height);
                                                     //!/< Set Up Our Perspective GL Screen
404
       if (!InitGL())
                                                        //!/< Initialize Our Newly Created GL Window
405
406
           407
408
409
410
411
                                                      //!/< Success
412
       return TRUE;
413 }
```

Here is the call graph for this function:



Here is the caller graph for this function:

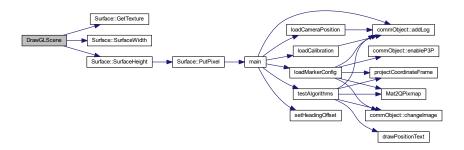


6.12.2.3 DrawGLScene()

Definition at line 76 of file supportcode.cpp.

```
78
        if(surf==NULL)
79
            return true;
80
81
        static bool frameThrottler = true;
        frameThrottler=!frameThrottler;
82
83
        if(frameThrottler) //!/<== Only display every other frame in case VSYNC is enabled. Otherwise
    return true; //!/<== application would get behind</pre>
84
85
86
        int pixelWidth = width;
87
        int pixelHeight= height;
88
89
90
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT); //!/< Clear The Screen And The Depth Buffer
91
        glLoadIdentity();
                                                                    //!/< Reset The View
92
        static GLuint ff = 0:
93
94
95
        int tex = surf->GetTexture();
        if(tex==0)
96
97
            tex=ff;
98
        else
            ff=tex;
99
100
         if(tex==0)
101
             return true;
102
103
         glBindTexture(GL_TEXTURE_2D, tex);
104
105
         glBegin(GL_QUADS);
             glTexCoord2f((0.0f, 0.0f); glVertex3f( 0, 0, 0); glTexCoord2f((GLfloat)pixelWidth/(GLfloat)surf->SurfaceWidth(), 0.0f); glVertex3f( (
106
107
      GLfloat) gWindowWidth,
                                0, 0);
108
             glTexCoord2f((GLfloat)pixelWidth/(GLfloat)surf->SurfaceWidth(), (GLfloat)pixelHeight/(
       GLfloat)surf->SurfaceHeight()); glVertex3f( (GLfloat)gWindowWidth, (GLfloat)
       gWindowHeight, 0);
      glTexCoord2f(0.0f, (GLfloat)pixelHeight/(GLfloat)surf->SurfaceHeight()); glVertex3f( 0, (GLfloat) gWindowHeight, 0);
109
110
        glEnd();
111
112
         SwapBuffers(hDC);
                                                   //!/< Swap Buffers (Double Buffering)
113
                                                                     //!/< Keep Going
114
         return true;
115 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



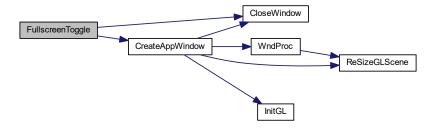
6.12.2.4 FullscreenToggle()

```
bool FullscreenToggle ( )
```

Definition at line 536 of file supportcode.cpp.

```
537 {
         keys[VK_F1]=FALSE;
                                                   //!/< If So Make Key FALSE
//!/< Kill Our Current Window
//!/< Toggle Fullscreen / Windowed Mode
538
         CloseWindow();
539
540
         gFullscreen=!gFullscreen;
541
542
          if (!CreateAppWindow(windowName,windowWidth,
       windowHeight,32,gFullscreen))
543
         {
544
              MessageBox(0, L"Unable to toggle to full screen", L"Error", MB_OK);
545
546
547
548
         return true;
549 }
```

Here is the call graph for this function:



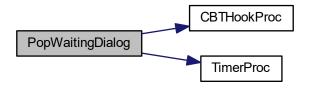
6.12.2.5 PopWaitingDialog()

```
bool PopWaitingDialog ( )
```

Definition at line 615 of file supportcode.cpp.

```
616 {
617
         //!/<== hook in so we can create a message box that has only a 'Cancel' button ==--
618
619
        hHook = SetWindowsHookEx(WH_CBT, reinterpret_cast<HOOKPROC>(&
      CBTHookProc), NULL, GetCurrentThreadId());
620
        UINT_PTR nTimer = SetTimer(0, 100, 3000,(TIMERPROC) TimerProc);
int iResult = MessageBox( 0, L"waiting for connected cameras...", L"Camera Initialization", MB_OK );
621
622
623
         if( iResult == IDOK )
624
625
              //!/<== user has clicked the cancel button ==--
626
             UnhookWindowsHookEx(hHook);
627
628
             return false;
629
630
        KillTimer(0, nTimer);
631
632
633
         return true;
634 }
```

Here is the call graph for this function:



6.12.2.6 PumpMessages()

```
bool PumpMessages ( )
```

Definition at line 551 of file supportcode.cpp.

```
552 {
553
         MSG msg;
554
555
          \  \  \, \text{if (PeekMessage(\&msg, NULL, 0, 0, PM\_REMOVE))} \qquad //!/< \, \, \text{Is There A Message Waiting?} 
556
             if (msg.message==WM_QUIT)
                                                          //!/< Have We Received A Quit Message?
558
                  return false;
                                                           //!/< If Not, Deal With Window Messages
559
            TranslateMessage(&msg);
560
                                                          //!/< Translate The Message
//!/< Dispatch The Message
561
562
                 DispatchMessage(&msg);
563
564
565
566
567 }
         return true;
```

6.12.2.7 WndProc()

```
LRESULT CALLBACK WndProc (

HWND ,

UINT ,

WPARAM ,

LPARAM )
```

/< Additional Message Information

Definition at line 118 of file supportcode.cpp.

```
122 {
123
        switch (uMsg)
                                                          //!/< Check For Windows Messages
124
            case WM_ACTIVATE:
                                                          //!/< Watch For Window Activate Message
125
126
                if (!HIWORD(wParam))
                                                          //!/< Check Minimization State
127
128
                {
129
                    gActive=TRUE;
                                                           //!/< Program Is Active
130
131
                else
132
                {
                    gActive=FALSE;
133
                                                          //!/< Program Is No Longer Active
134
                }
135
136
                return 0;
                                                          //!/< Return To The Message Loop
137
            }
138
139
            case WM POWERBROADCAST:
140
                if(wParam == PBT_APMSUSPEND)
141
142
                    CameraLibrary::CameraManager::X().PrepareForSuspend();
143
                if(wParam == PBT APMRESUMEAUTOMATIC)
144
145
                {
146
                    CameraLibrary::CameraManager::X().ResumeFromSuspend();
147
148
                break;
149
150
            case WM_SYSCOMMAND:
                                                          //!/< Intercept System Commands
151
152
                                                          //!/< Check System Calls
                switch (wParam)
153
154
                    case SC_SCREENSAVE:
                                                          //!/< Screensaver Trying To Start?
155
                    case SC_MONITORPOWER:
                                                          //!/< Monitor Trying To Enter Powersave?
156
                    return 0;
                                                          //!/< Prevent From Happening
157
                                                          //!/< Exit
                break:
158
159
            }
160
161
            case WM_CLOSE:
                                                          //!/< Did We Receive A Close Message?
162
                                                          //!/< Send A Quit Message
163
                PostQuitMessage(0);
164
                                                          //!/< Jump Back
                return 0;
165
            }
166
167
            case WM_KEYDOWN:
                                                          //!/< Is A Key Being Held Down?
168
                keys[wParam] = TRUE;
                                                          //\,!\,/\!< If So, Mark It As TRUE
169
170
                                                          //!/< Jump Back
                return 0;
171
172
173
            case WM_KEYUP:
                                                          //!/< Has A Key Been Released?
174
175
176
                keys[wParam] = FALSE;
                                                          //!/< If So, Mark It As FALSE
                                                          //!/< Jump Back
                return 0;
177
178
            case WM_MOVE:
179
                                                          //!/< Jump Back
180
            case WM_PAINT:
181
            case WM_SIZE:
                                                          //!/< Resize The OpenGL Window
182
183
            {
184
                ReSizeGLScene (LOWORD (1Param), HIWORD (1Param)); //!/< LoWord=Width, HiWord=Height
185
                                                         //!/< Jump Back
                return 0;
186
            }
187
        }
188
        //!/< Pass All Unhandled Messages To DefWindowProc
189
190
        return DefWindowProc(hWnd,uMsg,wParam,lParam);
191 }
```

Here is the call graph for this function:



Here is the caller graph for this function:



6.12.3 Variable Documentation

6.12.3.1 gActive

bool gActive

6.12.3.2 gFullscreen

bool gFullscreen

6.12.3.3 hDC

HDC hDC

Definition at line 24 of file supportcode.cpp.

6.12.3.4 keys

bool keys[256]

Index

BACKBUFFER_BITSPERPIXEL	main.h, 60
main.cpp, 40	l A l
1 B 10 B W	eulerAngles
calcBoardCornerPositions	main.cpp, 42
main.cpp, 15	eulerRef
calibrateCamera	main.cpp, 42
main.cpp, 16	exitRequested
main.h, 54	main.cpp, 43
calibrateGround	fromoTimo
main.cpp, 17	frameTime
main.h, 55	main.cpp, 43
camera_started	GET OPTIMIZED
main.cpp, 40	precomp.hpp, 77
cameraMatrix	getEulerAngles
main.cpp, 41	•
closeUDP	main.cpp, 23
main.cpp, 19	gotOrder
main.h, 57	main.cpp, 43
commObj	headingOffset
main.cpp, 41	_
main.h, 74	main.cpp, 43
coordinateFrame	IDI ICON1
main.cpp, 41	resource.h, 78
coordinateFrameProjected	IPAdressObject
main.cpp, 41	main.cpp, 44
currentMinIndex	main.h, 75
main.cpp, 41	IPAdressSafety
currentPointDistance	main.cpp, 44
main.cpp, 41	main.h, 75
тат.орр, т	
data	IPAdressSafety2
main.cpp, 42	main.cpp, 44
datagram	main.h, 75
main.cpp, 42	intExposure
Debug/moc_RigidTrack.cpp	main.cpp, 43
QT_MOC_LITERAL, 7	intFrameRate
Debug/moc_communication.cpp	main.cpp, 43
QT MOC LITERAL, 5	intIntensity
	main.cpp, 44
determineExposure	intThreshold
main.cpp, 20	main.cpp, 44
main.h, 57	invertZ
determineOrder	main.cpp, 44
main.cpp, 22	main.h, 75
main.h, 59	
distCoeffs	list_points2d
main.cpp, 42	main.cpp, 45
distModel	list_points2dDifference
main.cpp, 42	main.cpp, 45
drawPositionText	list_points2dOld
main.cpp, 23	main.cpp, 45

172 INDEX

list_points2dProjected	IPAdressSafety2, 44
main.cpp, 45	intExposure, 43
list_points2dUnsorted	intFrameRate, 43
main.cpp, 45	intIntensity, 44
list_points3d	intThreshold, 44
main.cpp, 45	invertZ, 44
IoadCalibration	list_points2d, 45
main.cpp, 24	list_points2dDifference, 45
main.h, 61	list_points2dOld, 45
IoadCameraPosition	list_points2dProjected, 45
main.cpp, 25	list_points2dUnsorted, 45
main.h, 61	list_points3d, 45
loadMarkerConfig	loadCalibration, 24
main.cpp, 25	loadCameraPosition, 25
main.h, 62	loadMarkerConfig, 25
logDate	logDate, 46
main.cpp, 46	logFileName, 46
logFileName	logName, 46
main.cpp, 46	logfile, 46
logName	M_CN, 46
main.cpp, 46	M_HeadingOffset, 46
logfile	main, 27
main.cpp, 46	Mat2QPixmap, 28
M CN	methodPNP, 47
M_CN	minPointDistance, 47
main.cpp, 46 M HeadingOffset	numberMarkers, 47
	pointOrderIndices, 47
main.cpp, 46 main	pointOrderIndicesNew, 47
	portObject, 47
main.cpp, 27	portSafety, 48
main.cpp BACKBUFFER BITSPERPIXEL, 40	portSafety2, 48
calcBoardCornerPositions, 15	posRef, 48
calibrateCamera, 16	position, 48
calibrateGamera, 10	positionOld, 48
camera_started, 40	projectCoordinateFrame, 29
cameraMatrix, 41	Rmat, 48
closeUDP, 19	RmatRef, 49
commObj, 41	Rvec, 49
coordinateFrame, 41	RvecOriginal, 49
coordinateFrameProjected, 41	safety2Enable, 49
currentMinIndex, 41	safetyAngle, 49
currentPointDistance, 41	safetyBoxLength, 49
data, 42	safetyEnable, 50
datagram, 42	sendDataUDP, 29
determineExposure, 20	setHeadingOffset, 30
determineOrder, 22	setReference, 30
distCoeffs, 42	setUpUDP, 32
distModel, 42	ss, 50
drawPositionText, 23	startStopCamera, 33
eulerAngles, 42	startTracking, 34
eulerRef, 42	strBuf, 50
exitRequested, 43	testAlgorithms, 39
frameTime, 43	timeFirstFrame, 50
getEulerAngles, 23	timeOld, 50
gotOrder, 43	Tvec, 50
headingOffset, 43	TvecOriginal, 51
IPAdressObject, 44	udpSocketObject, 51
•	
IPAdressSafety, 44	udpSocketSafety, 51

INDEX 173

udpSocketSafety2, 51	main.cpp, 48
useGuess, 51	position
velocity, 51	main.cpp, 48
main.h	positionOld
calibrateCamera, 54	main.cpp, 48
calibrateGround, 55	precomp.hpp
closeUDP, 57	GET_OPTIMIZED, 77
commObj, 74	projectCoordinateFrame
determineExposure, 57	main.cpp, 29
determineOrder, 59	main.h, 63
drawPositionText, 60	
IPAdressObject, 75	qCleanupResources_RigidTrack
IPAdressSafety, 75	qrc_RigidTrack.cpp, 9
IPAdressSafety2, 75	qInitResources_RigidTrack
invertZ, 75	qrc_RigidTrack.cpp, 9
loadCalibration, 61	QT_MOC_LITERAL
loadCameraPosition, 61	Debug/moc_RigidTrack.cpp, 7
loadMarkerConfig, 62	Debug/moc_communication.cpp, 5
methodPNP, 75	Release/moc_RigidTrack.cpp, 8
portObject, 75	Release/moc_communication.cpp, 6
portSafety, 76	QT_RCC_MANGLE_NAMESPACE
portSafety2, 76	qrc_RigidTrack.cpp, 8
projectCoordinateFrame, 63	QT_RCC_PREPEND_NAMESPACE
safety2Enable, 76	qrc_RigidTrack.cpp, 8
safetyAngle, 76	qrc_RigidTrack.cpp
safetyBoxLength, 76	qCleanupResources_RigidTrack, 9
safetyEnable, 76	qInitResources_RigidTrack, 9
sendDataUDP, 64	QT_RCC_MANGLE_NAMESPACE, 8
setHeadingOffset, 64	QT_RCC_PREPEND_NAMESPACE, 8
setReference, 65	Release/moc_RigidTrack.cpp
setUpUDP, 67	QT_MOC_LITERAL, 8
startStopCamera, 67	Release/moc_communication.cpp
startTracking, 68	QT_MOC_LITERAL, 6
testAlgorithms, 74	resource.h
Mat2QPixmap	IDI ICON1, 78
main.cpp, 28	RigidTrack/ modelest.h, 3
methodPNP	RigidTrack/GeneratedFiles/Debug/moc_RigidTrack.cpp,
main.cpp, 47	7
main.h, 75 minPointDistance	RigidTrack/GeneratedFiles/Debug/moc_communication.
	cpp, 5
main.cpp, 47	RigidTrack/GeneratedFiles/Release/moc RigidTrack.←
numberMarkers	cpp, 7
main.cpp, 47	RigidTrack/GeneratedFiles/Release/moc_communication.
	cpp, 6
pointOrderIndices	RigidTrack/GeneratedFiles/qrc_RigidTrack.cpp, 8
main.cpp, 47	RigidTrack/GeneratedFiles/ui_RigidTrack.h, 10
pointOrderIndicesNew	RigidTrack/RigidTrack.cpp, 78
main.cpp, 47	RigidTrack/RigidTrack.h, 79
portObject	RigidTrack/communication.cpp, 3
main.cpp, 47	RigidTrack/communication.h, 4
main.h, 75	RigidTrack/main.cpp, 11
portSafety	RigidTrack/main.h, 52
main.cpp, 48	RigidTrack/precomp.hpp, 77
main.h, 76	RigidTrack/resource.h, 78
portSafety2	Rmat
main.cpp, 48	main.cpp, 48
main.h, 76	RmatRef
posRef	main.cpp, 49

174 INDEX

Rvec main.cpp, 51 main.cpp, 49 velocity RvecOriginal main.cpp, 51 main.cpp, 49 safety2Enable main.cpp, 49 main.h, 76 safetyAngle main.cpp, 49 main.h, 76 safetyBoxLength main.cpp, 49 main.h, 76 safetyEnable main.cpp, 50 main.h, 76 sendDataUDP main.cpp, 29 main.h, 64 setHeadingOffset main.cpp, 30 main.h, 64 setReference main.cpp, 30 main.h, 65 setUpUDP main.cpp, 32 main.h, 67 main.cpp, 50 startStopCamera main.cpp, 33 main.h, 67 startTracking main.cpp, 34 main.h, 68 strBuf main.cpp, 50 testAlgorithms main.cpp, 39 main.h, 74 timeFirstFrame main.cpp, 50 timeOld main.cpp, 50 Tvec main.cpp, 50 TvecOriginal main.cpp, 51 udpSocketObject main.cpp, 51 udpSocketSafety main.cpp, 51 udpSocketSafety2 main.cpp, 51

useGuess