C++ Leistungskurs - Gruppe 7

1.0A

Generated by Doxygen 1.8.1.2

Tue Jan 27 2015 09:11:04

Contents

1 Main Page 1

2 Todo List 3

3 Namespace Index 5

3.1 Namespace List . . . 5

4 Class Index 7

4.1 Class List . . . 7

5 File Index 9

5.1 File List . . . 9

6 Namespace Documentation 11

6.1 config Namespace Reference . . . 11

6.1.1 Detailed Description . . . 14

6.1.2 Function Documentation . . . 15

6.1.2.1 gameAnnoyPositionL . . . 15

6.1.2.2 gameAnnoyPositionR . . . 15

6.1.2.3 gameGoalSlotL . . . 15

6.1.2.4 gameGoalSlotM . . . 15

6.1.2.5 gameGoalSlotOutsideLeft . . . 15

6.1.2.6 gameGoalSlotOutsideRight . . . 15

6.1.2.7 gameGoalSlotR . . . 15

6.1.2.8 guiBrushGoalBlue . . . 15

6.1.2.9 guiBrushGoalUndef . . . 15

6.1.2.10 guiBrushGoalYellow . . . 15

6.1.2.11 guiBrushMe . . . 15

6.1.2.12 guiBrushPole . . . 15

6.1.2.13 guiBrushPuckBlocked . . . 15

6.1.2.14 guiBrushPuckMoving . . . 15

6.1.2.15 guiPenDummy . . . 15

6.1.2.16 guiPenFieldPrimary . . . 16

6.1.2.17 guiPenFieldSecondary . . . 16
6.1.2.18 guiPenMe . . . 16
6.1.2.19 guiPenOpponent . . . 16
6.1.2.20 guiPenOrient . . . 16
6.1.2.21 guiPenPole . . . 16
6.1.2.22 guiPenPuckMe . . . 16
6.1.2.23 guiPenPuckMeOuter . . . 16
6.1.2.24 guiPenPuckOpponent . . . 16
6.1.2.25 guiPenPuckOpponentOuter . . . 16
6.1.2.26 guiPenPuckUndef . . . 16
6.1.2.27 guiPenPuckUndefOuter . . . 16
6.1.2.28 guiPenTarget . . . 16
6.1.3 Variable Documentation . . . 16
6.1.3.1 actorDistanceOfTargetOnSpline . . . 16
6.1.3.2 actorGatherPuckDistance . . . 16
6.1.3.3 actorLowPass . . . 16
6.1.3.4 actorMaxAngleLimiter . . . 17
6.1.3.5 actorMaxI . . . 17
6.1.3.6 actorMinAngleLimiter . . . 17
6.1.3.7 actorPeriodMotionControl . . . 17
6.1.3.8 actorPushAndReleaseAdditionalReverseDist . . . 17
6.1.3.9 actorPushPuckDistance . . . 17
6.1.3.10 actorReleasePuckDistance . . . 17
6.1.3.11 actorWaypointMaxAngleDeviation . . . 17
6.1.3.12 actorWaypointReachedDiffChange . . . 17
6.1.3.13 actorWaypointReachedDistance . . . 17
6.1.3.14 actorWPLowPassAlpha . . . 18
6.1.3.15 BORDERSIDE_CM . . . 18
6.1.3.16 BORDERTOP_CM . . . 18
6.1.3.17 CAM_PERCENTAGE_NON_COLOR_DETECTION . . . 18
6.1.3.18 CAM_ROI_HEIGHT_RELATIVE . . . 18
6.1.3.19 CAM_ROI_OFFSET_HORIZONTAL_RELATIVE . . . 18
6.1.3.20 CAM_ROI_OFFSET_VERTICAL_RELATIVE . . . 18
6.1.3.21 CAM_ROI_WIDTH_RELATIVE . . . 18
6.1.3.22 DIST_TO_PUCK_BEFORE_GATHERING_IT . . . 18
6.1.3.23 DISTANCE_TO_WAITING_LINE . . . 19
6.1.3.24 DUMP_SLOT_2_3 . . . 19
6.1.3.25 DUMP_SLOT_3_4 . . . 19
6.1.3.26 DUMP_SLOT_4_5 . . . 19
6.1.3.27 DUMP_SLOT_5_6 . . . 19

6.1.3.28 enableDebugActorHighLevel . . . 19
6.1.3.29 enableDebugActorLowLevel . . . 19
6.1.3.30 enableDebugCam . . . 19
6.1.3.31 enableDebugGame . . . 19
6.1.3.32 enableDebugMainwindow . . . 20
6.1.3.33 enableDebugMapData . . . 20
6.1.3.34 enableDebugOrientation . . . 20
6.1.3.35 enableDebugPathPlanning . . . 20
6.1.3.36 enableDebugSensorHighLevel . . . 20
6.1.3.37 enableDebugSensorLowLevel . . . 20
6.1.3.38 FIELDHEIGHT_CM . . . 20
6.1.3.39 FIELDWIDTH_CM . . . 20
6.1.3.40 gameBisZuWelchemAbstandWirdZielwackelnGefiltert . . . 20
6.1.3.41 gameMinimumDistanceEnemyToDumpSlot . . . 21
6.1.3.42 gameMinimumDistanceToEnemyRobot . . . 21
6.1.3.43 gameWievielBesserMussEinPuckSeinUmDasZielZuWechseln . . . 21
6.1.3.44 gameWieWeitMussDerGegnerVomZielEntferntSeinImAnnoyModus . . . 21
6.1.3.45 gameZielwackelfilterTiefpassKoeffizient . . . 21
6.1.3.46 geoFieldHeight . . . 21
6.1.3.47 geoFieldWidth . . . 21
6.1.3.48 geoGoalHeight . . . 21
6.1.3.49 geoGoalMarginBottom . . . 21
6.1.3.50 geoGoalMarginSide . . . 21
6.1.3.51 geoGoalWidth . . . 22
6.1.3.52 geoPol_1_14 . . . 22
6.1.3.53 geoPol_1_2 . . . 22
6.1.3.54 geoPol_2_3 . . . 22
6.1.3.55 geoPol_3_4 . . . 22
6.1.3.56 geoPoleRadiusReal . . . 22
6.1.3.57 geoPoleRadiusSim . . . 22
6.1.3.58 geoPuckRadiusBottom . . . 22
6.1.3.59 geoPuckRadiusTopReal . . . 22
6.1.3.60 geoPuckRadiusTopSim . . . 23
6.1.3.61 geoRobotForkCenterDist . . . 23
6.1.3.62 GUIopponent . . . 23
6.1.3.63 guiPuckKlickTolerance . . . 23
6.1.3.64 GUIpuckOpponent . . . 23
6.1.3.65 GUIpuckSelf . . . 23
6.1.3.66 GUIpuckUndef . . . 23
6.1.3.67 GUIREMOTETURNRATE . . . 23

6.1.3.68 GUIREMOTEVELOCITY . . . 23
6.1.3.69 GUIself . . . 24
6.1.3.70 GUItarget . . . 24
6.1.3.71 mapAbstandRoboterZentrumZuGabel . . . 24
6.1.3.72 mapIgnorePuckInsideEnemyDistance . . . 24
6.1.3.73 mapPolePuckFusionDistance . . . 24
6.1.3.74 mapPuckPuckFusionDistance . . . 24
6.1.3.75 mapToleranzBisWohinEinPuckInDerGabelIstMAX . . . 24
6.1.3.76 mapToleranzBisWohinEinPuckInDerGabelIstMIN . . . 24
6.1.3.77 obstacleCoordinateTolerance . . . 24
6.1.3.78 obstacleNumberOfPoles . . . 25
6.1.3.79 obstacleNumberOfPucks . . . 25
6.1.3.80 ORIENTATION_APPROXIMATION_VALUE . . . 25
6.1.3.81 ORIENTATION_SENSOR_ODOMETRIE_DELTA . . . 25
6.1.3.82 PATH_SHOWRES . . . 25
6.1.3.83 pathAdjacencyMultiplier . . . 25
6.1.3.84 pathArenaFieldAvoidMaxY . . . 25
6.1.3.85 pathArenaMaxX . . . 25
6.1.3.86 pathArenaMaxY . . . 25
6.1.3.87 pathArenaMinX . . . 26
6.1.3.88 pathArenaMinY . . . 26
6.1.3.89 pathEnemyCloseCost . . . 26
6.1.3.90 pathEnemyCloseDist . . . 26
6.1.3.91 pathEnemyFarDist . . . 26
6.1.3.92 pathEnemyStartCost . . . 26
6.1.3.93 pathGridSpacingBase . . . 26
6.1.3.94 pathMaxWPIterations . . . 26
6.1.3.95 pathPlanningEnabledUpwardsOfThisDistance . . . 26
6.1.3.96 pathPoleCloseCost . . . 27
6.1.3.97 pathPoleCloseDist . . . 27
6.1.3.98 pathPoleFarDist . . . 27
6.1.3.99 pathPoleStartCost . . . 27
6.1.3.100 pathPuckCloseCost . . . 27
6.1.3.101 pathPuckCloseDist . . . 27
6.1.3.102 pathPuckFarDist . . . 27
6.1.3.103 pathPuckStartCost . . . 27
6.1.3.104 pathRobotRadius . . . 27
6.1.3.105 pathTargetApproachAngleFullCost . . . 27
6.1.3.106 pathTargetApproachAngleInfluenceDistance . . . 28
6.1.3.107 pathTargetApproachAngleMaxDeviationWithoutFullCost . . . 28

6.1.3.108 periodAlive . . . 28
6.1.3.109 periodEgoPos . . . 28
6.1.3.110 periodTillAnnoy . . . 28
6.1.3.111 POLESIZE_CM . . . 28
6.1.3.112 puckIsCloseToPoleDistance . . . 28
6.1.3.113 PUCKSIZE_INNER_CM . . . 28
6.1.3.114 PUCKSIZE_OUTER_CM . . . 28
6.1.3.115 refIP . . . 29
6.1.3.116 refPort . . . 29
6.1.3.117 refVerbose . . . 29
6.1.3.118 ROBOSIZE_CM . . . 29
6.1.3.119 SENSOR_COLLISION_AT . . . 29
6.1.3.120 SENSOR_DELTA_ANGLE . . . 29
6.1.3.121 SENSOR_MAX_DISTANCE_OF_OBJ . . . 29
6.1.3.122 SENSOR_MAX_RANGE_ORIENTATION . . . 29
6.1.3.123 SENSOR_MAX_RANGE_RECOGNITION . . . 29
6.1.3.124 SENSOR_MEASUREMENT_DEVIATION . . . 30
6.1.3.125 SENSOR_OBJECTWIDTH_ROBO . . . 30
6.1.3.126 SENSOR_OUT_OF_FIELD_TOLERANCE . . . 30
6.1.3.127 SENSOR_RADIUS_ROBOT . . . 30
6.1.3.128 SENSOR_WAIT_COUNTER . . . 30
6.1.3.129 TARGET_POLE_VARIANCE . . . 30
6.1.3.130 TARGETSIZE_CM . . . 30
6.1.3.131 teamID . . . 30
6.2 cv Namespace Reference . . . 30
6.3 Filter Namespace Reference . . . 31
6.3.1 Typedef Documentation . . . 31
6.3.1.1 element . . . 31
6.3.2 Function Documentation . . . 31
6.3.2.1 _medianfilter . . . 31
6.3.2.2 _medianfilter . . . 31
6.3.2.3 medianfilter . . . 31
6.3.2.4 medianfilter . . . 31
6.4 PlayerCc Namespace Reference . . . 31
6.5 tkqt Namespace Reference . . . 31
6.5.1 Detailed Description . . . 32
6.6 trilateration Namespace Reference . . . 32
6.6.1 Detailed Description . . . 32
6.6.2 Function Documentation . . . 32
6.6.2.1 circle_circle_intersection . . . 32

6.7 Ui Namespace Reference . . . 33

7 Class Documentation 35

7.1 ActorHighLevel Class Reference . . . 35

7.1.1 Detailed Description . . . 37

7.1.2 Constructor & Destructor Documentation . . . 37

7.1.2.1 ActorHighLevel . . . 37

7.1.2.2 ~ActorHighLevel . . . 37

7.1.3 Member Function Documentation . . . 38

7.1.3.1 constrainAngle . . . 38

7.1.3.2 getState . . . 38

7.1.3.3 ignoreSignals . . . 38

7.1.3.4 lowPass . . . 38

7.1.3.5 resetPIDtempVars . . . 38

7.1.3.6 setState . . . 38

7.1.3.7 signalPIDPlot . . . 39

7.1.3.8 signalPuckDone . . . 39

7.1.3.9 signalSendRobotControlParams . . . 39

7.1.3.10 signalSplinePlot . . . 39

7.1.3.11 slotChangePIDParams . . . 39

7.1.3.12 slotGatherPuck . . . 39

7.1.3.13 slotPushAndReleasePuck . . . 40

7.1.3.14 slotReleasePuck . . . 40

7.1.3.15 slotTimerSendPIDPlot . . . 40

7.1.3.16 slotUpdateWaypoints . . . 40

7.1.3.17 startPIDController . . . 40

7.1.3.18 takeDimension . . . 40

7.1.4 Member Data Documentation . . . 41

7.1.4.1 additionalReleasePuckDistance . . . 41

7.1.4.2 elapsedTime . . . 41

7.1.4.3 enabled . . . 41

7.1.4.4 iDeltaA . . . 41

7.1.4.5 iDeltaL . . . 41

7.1.4.6 internalWP . . . 41

7.1.4.7 lastDeltaA . . . 41

7.1.4.8 lastDeltaL . . . 41

7.1.4.9 mutexPidHist . . . 42

7.1.4.10 mutexState . . . 42

7.1.4.11 numWP . . . 42

7.1.4.12 PID_A_D . . . 42

7.1.4.13 PID_A_I . . . 42
7.1.4.14 PID_A_P . . . 42
7.1.4.15 PID_V_D . . . 42
7.1.4.16 PID_V_I . . . 42
7.1.4.17 PID_V_P . . . 42
7.1.4.18 pidHistDistIst . . . 43
7.1.4.19 pidHistDistSoll . . . 43
7.1.4.20 pidHistTime . . . 43
7.1.4.21 pidHistWinkelIst . . . 43
7.1.4.22 pidHistWinkelSoll . . . 43
7.1.4.23 positionWasReached . . . 43
7.1.4.24 quitting . . . 43
7.1.4.25 releasePuckOrigin . . . 43
7.1.4.26 robotPosition . . . 43
7.1.4.27 splineX . . . 43
7.1.4.28 splineY . . . 44
7.1.4.29 state . . . 44
7.1.4.30 streamPIDEnabled . . . 44
7.1.4.31 targetDistDiffLast . . . 44
7.1.4.32 targetDistLast . . . 44
7.1.4.33 timeOfStart . . . 44
7.1.4.34 timerPIDPlot . . . 44
7.2 ActorLowLevel Class Reference . . . 44
7.2.1 Detailed Description . . . 45
7.2.2 Constructor & Destructor Documentation . . . 45
7.2.2.1 ActorLowLevel . . . 45
7.2.2.2 ∼ActorLowLevel . . . 45
7.2.3 Member Function Documentation . . . 45
7.2.3.1 moveRobot . . . 45
7.2.3.2 setOdometry . . . 46
7.2.3.3 setRobotControllParams . . . 46
7.2.3.4 setRobotRemoteControllParams . . . 46
7.2.3.5 slotEmergencyStopEnabled . . . 46
7.2.4 Member Data Documentation . . . 46
7.2.4.1 isRefereeEmergency . . . 47
7.2.4.2 moveForward . . . 47
7.2.4.3 positionProxy . . . 47
7.2.4.4 previousTurnRate . . . 47
7.2.4.5 previousVelocity . . . 47
7.2.4.6 tempTurnRate . . . 47

7.2.4.7 tempVelocity . . . 47
7.3 tkqt::band_matrix Class Reference . . . 47
7.3.1 Detailed Description . . . 48
7.3.2 Constructor & Destructor Documentation . . . 48
7.3.2.1 band_matrix . . . 48
7.3.2.2 band_matrix . . . 49
7.3.2.3 ~band_matrix . . . 49
7.3.3 Member Function Documentation . . . 49
7.3.3.1 dim . . . 49
7.3.3.2 l_solve . . . 49
7.3.3.3 lu_decompose . . . 49
7.3.3.4 lu_solve . . . 49
7.3.3.5 num_lower . . . 50
7.3.3.6 num_upper . . . 50
7.3.3.7 operator() . . . 50
7.3.3.8 operator() . . . 50
7.3.3.9 r_solve . . . 50
7.3.3.10 resize . . . 51
7.3.3.11 saved_diag . . . 51
7.3.3.12 saved_diag . . . 51
7.3.4 Member Data Documentation . . . 51
7.3.4.1 m_lower . . . 51
7.3.4.2 m_upper . . . 51
7.4 Cam Class Reference . . . 51
7.4.1 Detailed Description . . . 52
7.4.2 Constructor & Destructor Documentation . . . 52
7.4.2.1 Cam . . . 52
7.4.2.2 ~Cam . . . 53
7.4.3 Member Function Documentation . . . 53
7.4.3.1 getLastColor . . . 53
7.4.3.2 getPixelColor . . . 53
7.4.3.3 grabFrameAndColor . . . 53
7.4.3.4 signalColorDetected . . . 53
7.4.3.5 signalDisplayFrame . . . 53
7.4.3.6 slotSetCameraParams . . . 53
7.4.3.7 slotStartColorDetection . . . 54
7.4.3.8 timerSendFrame . . . 54
7.4.4 Member Data Documentation . . . 54
7.4.4.1 color . . . 54
7.4.4.2 cp . . . 54

7.4.4.3 enabled . . . 54
7.4.4.4 mutexVideoCapture . . . 54
7.4.4.5 streamCamEnabled . . . 54
7.4.4.6 timer . . . 54
7.4.4.7 videoCapture . . . 55
7.5 CameraParams Struct Reference . . . 55
7.5.1 Detailed Description . . . 55
7.5.2 Member Data Documentation . . . 55
7.5.2.1 colorBlueMax . . . 55
7.5.2.2 colorBlueMin . . . 55
7.5.2.3 colorGreenMax . . . 55
7.5.2.4 colorGreenMin . . . 56
7.5.2.5 colorYellowMax . . . 56
7.5.2.6 colorYellowMin . . . 56
7.5.2.7 height . . . 56
7.5.2.8 source . . . 56
7.5.2.9 updatePeriod . . . 56
7.5.2.10 width . . . 56
7.6 ConstrainedLaserData Class Reference . . . 56
7.6.1 Detailed Description . . . 57
7.6.2 Constructor & Destructor Documentation . . . 57
7.6.2.1 ConstrainedLaserData . . . 57
7.6.2.2 ~ConstrainedLaserData . . . 57
7.6.3 Member Function Documentation . . . 57
7.6.3.1 addAngle . . . 57
7.6.3.2 addFilteredDepth . . . 58
7.6.3.3 addRawDepth . . . 58
7.6.3.4 angles . . . 58
7.6.3.5 clearData . . . 58
7.6.3.6 filteredDepths . . . 58
7.6.3.7 rawDepths . . . 58
7.6.4 Member Data Documentation . . . 59
7.6.4.1 m_angles . . . 59
7.6.4.2 m_filteredDepth . . . 59
7.6.4.3 m_rawDepths . . . 59
7.7 CVImageWidget Class Reference . . . 59
7.7.1 Detailed Description . . . 60
7.7.2 Constructor & Destructor Documentation . . . 60
7.7.2.1 CVImageWidget . . . 60
7.7.2.2 CVImageWidget . . . 60

7.7.3 Member Function Documentation . . . 60
7.7.3.1 minimumSizeHint . . . 60
7.7.3.2 minimumSizeHint . . . 60
7.7.3.3 paintEvent . . . 60
7.7.3.4 paintEvent . . . 60
7.7.3.5 showImage . . . 60
7.7.3.6 showImage . . . 61
7.7.3.7 sizeHint . . . 61
7.7.3.8 sizeHint . . . 61
7.7.4 Member Data Documentation . . . 61
7.7.4.1 _qimage . . . 61
7.7.4.2 _tmp . . . 61
7.8 FilterParams Struct Reference . . . 61
7.8.1 Detailed Description . . . 61
7.8.2 Constructor & Destructor Documentation . . . 62
7.8.2.1 FilterParams . . . 62
7.8.2.2 FilterParams . . . 62
7.8.3 Member Data Documentation . . . 62
7.8.3.1 kernel . . . 62
7.8.3.2 ObsFilterAnzahl . . . 62
7.8.3.3 ObsFilterSchnitt . . . 62
7.8.3.4 PosFilterAnzahl . . . 62
7.8.3.5 PosFilterSchnitt . . . 62
7.9 Game Class Reference . . . 62
7.9.1 Detailed Description . . . 64
7.9.2 Member Enumeration Documentation . . . 64
7.9.2.1 GameState . . . 64
7.9.2.2 HandleEnemyPuckState . . . 65
7.9.2.3 SubGameStateAnnoy . . . 65
7.9.3 Constructor & Destructor Documentation . . . 65
7.9.3.1 Game . . . 65
7.9.3.2 ~Game . . . 65
7.9.4 Member Function Documentation . . . 65
7.9.4.1 annoyFoeState . . . 66
7.9.4.2 checkTargetPuckAvailable . . . 66
7.9.4.3 createTargetInfrontPuck . . . 66
7.9.4.4 driveToGoal . . . 66
7.9.4.5 findAndSelectBestPuck . . . 66
7.9.4.6 findBestGoalSlot . . . 66
7.9.4.7 findBestPuck . . . 66

7.9.4.8 findDumpSlot . . . 67
7.9.4.9 getParkPosition . . . 67
7.9.4.10 getPositionToTurnRoboToMiddlePoint . . . 67
7.9.4.11 getTargetForPuck . . . 67
7.9.4.12 getTargetForPuckBeforePole . . . 67
7.9.4.13 handleEnemyPuck . . . 67
7.9.4.14 isPuckAwayFromEnemy . . . 67
7.9.4.15 isPuckAwayFromPole . . . 68
7.9.4.16 quit . . . 68
7.9.4.17 run . . . 68
7.9.4.18 signalGatherPuck . . . 68
7.9.4.19 signalPuckRelease . . . 68
7.9.4.20 signalPushAndRelease . . . 68
7.9.4.21 signalReportGoal . . . 68
7.9.4.22 signalStartColorDetectAI . . . 68
7.9.4.23 slotActorHighLevelIsDoneWithPuck . . . 68
7.9.4.24 slotAnnoyFoe . . . 68
7.9.4.25 slotColorDetect . . . 69
7.9.4.26 slotStartGame . . . 69
7.9.5 Member Data Documentation . . . 69
7.9.5.1 colorfail . . . 69
7.9.5.2 currentPuck . . . 69
7.9.5.3 currentPuckPrio . . . 69
7.9.5.4 gameEngine . . . 69
7.9.5.5 goalSlotCounterList . . . 69
7.9.5.6 handleEnemyPuckState . . . 69
7.9.5.7 isGameStarted . . . 69
7.9.5.8 lastTargetedPuck . . . 70
7.9.5.9 quitting . . . 70
7.9.5.10 state . . . 70
7.9.5.11 subAnnoyState . . . 70
7.9.5.12 timerLostPuck . . . 70
7.9.5.13 timerUpdateGoalSlot . . . 70
7.10 GameEngine Class Reference . . . 70
7.10.1 Detailed Description . . . 72
7.10.2 Member Enumeration Documentation . . . 72
7.10.2.1 StateNameEnum . . . 72
7.10.3 Constructor & Destructor Documentation . . . 72
7.10.3.1 GameEngine . . . 72
7.10.3.2 ~GameEngine . . . 72

7.10.4 Member Function Documentation . . . 72
7.10.4.1 getState . . . 72
7.10.4.2 signalAnnoyFoe . . . 72
7.10.4.3 signalEmergencyStopEnabled . . . 73
7.10.4.4 signalStartDetection . . . 73
7.10.4.5 signalStartGame . . . 73
7.10.4.6 slotDetectionFinished . . . 73
7.10.4.7 slotRefConnected . . . 73
7.10.4.8 slotRefConnectFailed . . . 73
7.10.4.9 slotRefDetectionStart . . . 73
7.10.4.10 slotRefDisconnected . . . 73
7.10.4.11 slotRefGameOver . . . 74
7.10.4.12 slotRefGameStart . . . 74
7.10.4.13 slotRefStopMovement . . . 74
7.10.4.14 slotRefTrueColorOfTeam . . . 74
7.10.4.15 slotReportGoal . . . 74
7.10.4.16 slotTimerAlive . . . 74
7.10.4.17 slotTimerAnnoy . . . 74
7.10.4.18 slotTimerEgoPos . . . 74
7.10.4.19 startGameEngine . . . 74
7.10.5 Member Data Documentation . . . 75
7.10.5.1 referee . . . 75
7.10.5.2 state . . . 75
7.10.5.3 timerAlive . . . 75
7.10.5.4 timerEgoPos . . . 75
7.10.5.5 timerTillAnnoyEnemy . . . 75
7.11 PathPlanning::GridPoint Class Reference . . . 75
7.11.1 Detailed Description . . . 76
7.11.2 Member Enumeration Documentation . . . 76
7.11.2.1 PointType . . . 76
7.11.3 Constructor & Destructor Documentation . . . 76
7.11.3.1 GridPoint . . . 76
7.11.3.2 GridPoint . . . 77
7.11.4 Member Function Documentation . . . 77
7.11.4.1 calculateIntrinsicCost . . . 77
7.11.4.2 constrainAngle . . . 77
7.11.4.3 getDistance . . . 77
7.11.4.4 getGradientPoint . . . 77
7.11.4.5 getNeighbors . . . 78
7.11.4.6 init . . . 78

7.11.5 Member Data Documentation . . . 78
7.11.5.1 active . . . 78
7.11.5.2 gridA . . . 78
7.11.5.3 gridB . . . 78
7.11.5.4 intrinsicCost . . . 78
7.11.5.5 isOutsideArena . . . 78
7.11.5.6 pathPlanning . . . 79
7.11.5.7 positionX . . . 79
7.11.5.8 positionY . . . 79
7.11.5.9 type . . . 79
7.11.5.10 value . . . 79
7.12 Hermes Class Reference . . . 79
7.12.1 Detailed Description . . . 79
7.12.2 Constructor & Destructor Documentation . . . 80
7.12.2.1 Hermes . . . 80
7.12.2.2 ∼Hermes . . . 80
7.12.3 Member Data Documentation . . . 80
7.12.3.1 messageSize . . . 80
7.12.3.2 myTeamID . . . 80
7.12.3.3 referee . . . 80
7.13 LaserPlotData Struct Reference . . . 80
7.13.1 Detailed Description . . . 81
7.13.2 Member Enumeration Documentation . . . 81
7.13.2.1 DataTypeEnum . . . 81
7.13.3 Constructor & Destructor Documentation . . . 81
7.13.3.1 LaserPlotData . . . 81
7.13.3.2 LaserPlotData . . . 81
7.13.3.3 LaserPlotData . . . 81
7.13.4 Member Data Documentation . . . 82
7.13.4.1 angles . . . 82
7.13.4.2 data . . . 82
7.13.4.3 dataType . . . 82
7.13.4.4 sizes . . . 82
7.14 Log Class Reference . . . 82
7.14.1 Detailed Description . . . 83
7.14.2 Constructor & Destructor Documentation . . . 83
7.14.2.1 Log . . . 83
7.14.3 Member Function Documentation . . . 83
7.14.3.1 customLogger . . . 83
7.14.3.2 setMainWindowReference . . . 83

7.14.4 Member Data Documentation . . . 83
7.14.4.1 logParams . . . 83
7.14.4.2 mainWindow . . . 83
7.14.4.3 streamLogEnabled . . . 84
7.15 LogParams Struct Reference . . . 84
7.15.1 Detailed Description . . . 84
7.15.2 Member Enumeration Documentation . . . 84
7.15.2.1 logLevelEnum . . . 84
7.15.2.2 logLevelEnum . . . 85
7.15.3 Member Data Documentation . . . 85
7.15.3.1 logActor . . . 85
7.15.3.2 logAI . . . 85
7.15.3.3 logData . . . 85
7.15.3.4 logLevel . . . 85
7.15.3.5 logMain . . . 85
7.15.3.6 logOthers . . . 85
7.15.3.7 logPlots . . . 86
7.15.3.8 logSensor . . . 86
7.16 MainWindow Class Reference . . . 86
7.16.1 Detailed Description . . . 91
7.16.2 Constructor & Destructor Documentation . . . 92
7.16.2.1 MainWindow . . . 92
7.16.2.2 ~MainWindow . . . 92
7.16.2.3 MainWindow . . . 92
7.16.2.4 ~MainWindow . . . 92
7.16.3 Member Function Documentation . . . 92
7.16.3.1 back . . . 92
7.16.3.2 back . . . 92
7.16.3.3 changeCamParams . . . 92
7.16.3.4 changeCamParams . . . 92
7.16.3.5 changeFilterParams . . . 92
7.16.3.6 changeFilterParams . . . 92
7.16.3.7 changeLogParams . . . 92
7.16.3.8 changeLogParams . . . 93
7.16.3.9 changePIDParams . . . 93
7.16.3.10 changePIDParams . . . 93
7.16.3.11 clearTargets . . . 93
7.16.3.12 clearTargets . . . 93
7.16.3.13 convertX . . . 93
7.16.3.14 convertX . . . 93

7.16.3.15 convertY . . . 93
7.16.3.16 convertY . . . 94
7.16.3.17 drawMap . . . 94
7.16.3.18 drawMap . . . 94
7.16.3.19 forward . . . 94
7.16.3.20 forward . . . 94
7.16.3.21 left . . . 94
7.16.3.22 left . . . 94
7.16.3.23 mousePressEvent . . . 95
7.16.3.24 mousePressEvent . . . 95
7.16.3.25 on_btn_ReleasePuck_clicked . . . 95
7.16.3.26 on_btn_StartGame_clicked . . . 95
7.16.3.27 on_btnDetectColor_clicked . . . 95
7.16.3.28 on_camSourceSpin_valueChanged . . . 95
7.16.3.29 on_camSourceSpin_valueChanged . . . 95
7.16.3.30 on_cbStream_stateChanged . . . 96
7.16.3.31 on_cbStream_stateChanged . . . 96
7.16.3.32 on_logCBActor_stateChanged . . . 96
7.16.3.33 on_logCBActor_stateChanged . . . 96
7.16.3.34 on_logCBAI_stateChanged . . . 96
7.16.3.35 on_logCBAI_stateChanged . . . 96
7.16.3.36 on_logCBData_stateChanged . . . 97
7.16.3.37 on_logCBData_stateChanged . . . 97
7.16.3.38 on_logCBOther_stateChanged . . . 97
7.16.3.39 on_logCBOther_stateChanged . . . 97
7.16.3.40 on_logCBPlot_stateChanged . . . 97
7.16.3.41 on_logCBPlot_stateChanged . . . 97
7.16.3.42 on_logCBSensor_stateChanged . . . 98
7.16.3.43 on_logCBSensor_stateChanged . . . 98
7.16.3.44 on_logLevel_currentIndexChanged . . . 98
7.16.3.45 on_logLevel_currentIndexChanged . . . 98
7.16.3.46 on_pushButton_StartOrientation_clicked . . . 98
7.16.3.47 on_refreshTime_valueChanged . . . 98
7.16.3.48 on_sizeX_textChanged . . . 99
7.16.3.49 on_sizeX_textChanged . . . 99
7.16.3.50 on_sizeY_textChanged . . . 99
7.16.3.51 on_sizeY_textChanged . . . 99
7.16.3.52 on_spinBox_kernel_valueChanged . . . 99
7.16.3.53 on_spinPIDAD_valueChanged . . . 99
7.16.3.54 on_spinPIDAD_valueChanged . . . 100

7.16.3.55 on_spinPIDAI_valueChanged . . . 100
7.16.3.56 on_spinPIDAI_valueChanged . . . 100
7.16.3.57 on_spinPIDAP_valueChanged . . . 100
7.16.3.58 on_spinPIDAP_valueChanged . . . 100
7.16.3.59 on_spinPIDVD_valueChanged . . . 100
7.16.3.60 on_spinPIDVD_valueChanged . . . 101
7.16.3.61 on_spinPIDVI_valueChanged . . . 101
7.16.3.62 on_spinPIDVI_valueChanged . . . 101
7.16.3.63 on_spinPIDVP_valueChanged . . . 101
7.16.3.64 on_spinPIDVP_valueChanged . . . 101
7.16.3.65 on_streamLog_stateChanged . . . 101
7.16.3.66 on_streamLog_stateChanged . . . 101
7.16.3.67 on_streamPath_stateChanged . . . 102
7.16.3.68 on_streamPath_stateChanged . . . 102
7.16.3.69 on_streamPID_stateChanged . . . 102
7.16.3.70 on_streamPID_stateChanged . . . 102
7.16.3.71 on_streamSensor_stateChanged . . . 102
7.16.3.72 on_streamSensor_stateChanged . . . 102
7.16.3.73 on_updateSpinner_valueChanged . . . 102
7.16.3.74 on_updateSpinner_valueChanged . . . 102
7.16.3.75 orientationSetup . . . 103
7.16.3.76 refresh . . . 103
7.16.3.77 refresh . . . 103
7.16.3.78 right . . . 103
7.16.3.79 right . . . 103
7.16.3.80 robotRemoteControllUpdate . . . 103
7.16.3.81 robotRemoteControllUpdate . . . 103
7.16.3.82 setrefreshrate . . . 103
7.16.3.83 setup . . . 104
7.16.3.84 setup . . . 104
7.16.3.85 signalChangeCamParams . . . 104
7.16.3.86 signalChangeCamParams . . . 104
7.16.3.87 signalChangeFilterParams . . . 104
7.16.3.88 signalChangeFilterParams . . . 104
7.16.3.89 signalChangePIDParams . . . 104
7.16.3.90 signalChangePIDParams . . . 105
7.16.3.91 signalStartOrientation . . . 105
7.16.3.92 signalStartOrientation . . . 105
7.16.3.93 signalTestColorDetect . . . 105
7.16.3.94 signalTestPuckRelease . . . 105

7.16.3.95 signalTestStartGame . . . 105
7.16.3.96 slotDisplayFrame . . . 105
7.16.3.97 slotDisplayFrame . . . 105
7.16.3.98 slotLaserDisplay . . . 105
7.16.3.99 slotLaserDisplay . . . 106
7.16.3.100 slotLog . . . 106
7.16.3.101 slotLog . . . 106
7.16.3.102 slotPIDPlot . . . 106
7.16.3.103 slotPIDPlot . . . 106
7.16.3.104 slotRestartTimerDisplay . . . 106
7.16.3.105 slotSimulationDetect . . . 107
7.16.3.106 slotUpdateColorLabel . . . 107
7.16.3.107 stop . . . 107
7.16.3.108 stop . . . 107
7.16.3.109 strongleft . . . 107
7.16.3.110 strongleft . . . 107
7.16.3.111 strongright . . . 107
7.16.3.112 strongright . . . 107
7.16.3.113 updatePathDisplay . . . 107
7.16.3.114 updatePathDisplay . . . 108
7.16.3.115 updateRemoteOdometry . . . 108
7.16.3.116 updateRemoteOdometry . . . 108
7.16.4 Member Data Documentation . . . 108
7.16.4.1 colorMap . . . 108
7.16.4.2 colors . . . 108
7.16.4.3 graphicsScene_1 . . . 108
7.16.4.4 graphicsScene_2 . . . 108
7.16.4.5 graphicsScene_3 . . . 108
7.16.4.6 graphicsScene_4 . . . 108
7.16.4.7 graphicsScene_5 . . . 109
7.16.4.8 graphicsScene_6 . . . 109
7.16.4.9 graphLaserMedian . . . 109
7.16.4.10 graphLaserObjects . . . 109
7.16.4.11 graphLaserRaw . . . 109
7.16.4.12 graphLaserReduced . . . 109
7.16.4.13 graphPIDAIst . . . 109
7.16.4.14 graphPIDASoll . . . 109
7.16.4.15 graphPIDDIst . . . 109
7.16.4.16 graphPIDDSoll . . . 109
7.16.4.17 graphRobotScatter . . . 110

7.16.4.18 graphSplineCurve . . . 110
7.16.4.19 graphWPScatter . . . 110
7.16.4.20 m_changeOrientation . . . 110
7.16.4.21 map . . . 110
7.16.4.22 refreshtimer . . . 110
7.16.4.23 scatterLaserObjects . . . 110
7.16.4.24 scatterRobotStyle . . . 110
7.16.4.25 scatterWPStyle . . . 110
7.16.4.26 timer . . . 111
7.16.4.27 TrackingMe . . . 111
7.16.4.28 TrackingYou . . . 111
7.16.4.29 ui . . . 111
7.17 MapData Class Reference . . . 111
7.17.1 Detailed Description . . . 113
7.17.2 Constructor & Destructor Documentation . . . 114
7.17.2.1 MapData . . . 114
7.17.2.2 ~MapData . . . 114
7.17.2.3 MapData . . . 114
7.17.3 Member Function Documentation . . . 114
7.17.3.1 checkForEnemyNearTraget . . . 114
7.17.3.2 cleanup . . . 114
7.17.3.3 clearTargets . . . 114
7.17.3.4 compareObstacleTimestamps . . . 114
7.17.3.5 deleteFirstTarget . . . 115
7.17.3.6 deleteObstacle . . . 115
7.17.3.7 getDisableEmergency . . . 115
7.17.3.8 getFirstTarget . . . 115
7.17.3.9 getListByType . . . 115
7.17.3.10 getObstacle . . . 116
7.17.3.11 getObstacle . . . 116
7.17.3.12 getObstacle . . . 116
7.17.3.13 getObstacle . . . 117
7.17.3.14 getPointerToPathPlanner . . . 117
7.17.3.15 getRobotPosition . . . 117
7.17.3.16 getSimulationDetected . . . 117
7.17.3.17 getTargetNearEnemy . . . 117
7.17.3.18 getTeamColor . . . 118
7.17.3.19 isPuckInFork . . . 118
7.17.3.20 operator= . . . 118
7.17.3.21 organizeObstacles . . . 118

7.17.3.22 setActualColor . . . 119
7.17.3.23 setDisableEmergency . . . 119
7.17.3.24 setObstacle . . . 119
7.17.3.25 setObstacle . . . 119
7.17.3.26 setObstacle . . . 120
7.17.3.27 setObstacle . . . 120
7.17.3.28 setPointerToPathPlanner . . . 120
7.17.3.29 setProbableColor . . . 120
7.17.3.30 setPuckInFork . . . 120
7.17.3.31 setSimulationDetected . . . 121
7.17.3.32 setTargetNearEnemy . . . 121
7.17.4 Member Data Documentation . . . 121
7.17.4.1 disableEmergency . . . 121
7.17.4.2 mutexDisableEmergency . . . 121
7.17.4.3 mutexPointerToPathPlanner . . . 121
7.17.4.4 mutexPoles . . . 121
7.17.4.5 mutexPuckInFork . . . 121
7.17.4.6 mutexPucks . . . 122
7.17.4.7 mutexRobotDummy . . . 122
7.17.4.8 mutexRobotME . . . 122
7.17.4.9 mutexRobotOpponent . . . 122
7.17.4.10 mutexSimulationDetected . . . 122
7.17.4.11 mutexTargetNearEnemy . . . 122
7.17.4.12 mutexTargets . . . 122
7.17.4.13 mutexTeamColor . . . 122
7.17.4.14 mutexUnidentified . . . 122
7.17.4.15 obstacleDummy . . . 123
7.17.4.16 obstacleMe . . . 123
7.17.4.17 obstacleOpponent . . . 123
7.17.4.18 obstaclesPoles . . . 123
7.17.4.19 obstaclesPucks . . . 123
7.17.4.20 obstaclesTargets . . . 123
7.17.4.21 obstaclesUnidentified . . . 123
7.17.4.22 pointerToPathPlanner . . . 123
7.17.4.23 puckInFork . . . 123
7.17.4.24 simulationDetected . . . 124
7.17.4.25 targetNearEnemy . . . 124
7.17.4.26 teamColor . . . 124
7.18 MedianFilter Class Reference . . . 124
7.18.1 Detailed Description . . . 124

7.18.2 Member Function Documentation . . . 124
7.18.2.1 filter . . . 124
7.19 Obstacle Class Reference . . . 125
7.19.1 Detailed Description . . . 127
7.19.2 Constructor & Destructor Documentation . . . 127
7.19.2.1 Obstacle . . . 127
7.19.2.2 Obstacle . . . 127
7.19.2.3 Obstacle . . . 127
7.19.2.4 Obstacle . . . 127
7.19.2.5 Obstacle . . . 128
7.19.2.6 Obstacle . . . 128
7.19.2.7 Obstacle . . . 128
7.19.2.8 Obstacle . . . 128
7.19.2.9 Obstacle . . . 128
7.19.2.10 Obstacle . . . 129
7.19.2.11 Obstacle . . . 129
7.19.2.12 Obstacle . . . 129
7.19.2.13 ∼Obstacle . . . 129
7.19.3 Member Function Documentation . . . 129
7.19.3.1 getColor . . . 129
7.19.3.2 getCoords . . . 130
7.19.3.3 getDistanceTo . . . 130
7.19.3.4 getInitialized . . . 130
7.19.3.5 getLastUpdate . . . 130
7.19.3.6 getOrientation . . . 130
7.19.3.7 getPosition . . . 131
7.19.3.8 getStatus . . . 131
7.19.3.9 getType . . . 131
7.19.3.10 isInSpecifiedArea . . . 131
7.19.3.11 mergeWith . . . 131
7.19.3.12 operator< . . . 132
7.19.3.13 operator== . . . 132
7.19.3.14 setColor . . . 132
7.19.3.15 setCoords . . . 132
7.19.3.16 setInitialized . . . 133
7.19.3.17 setLastUpdate . . . 133
7.19.3.18 setOrientation . . . 133
7.19.3.19 setPosition . . . 133
7.19.3.20 setStatus . . . 133
7.19.3.21 setType . . . 134

7.19.4 Member Data Documentation . . . 134
7.19.4.1 bInitialized . . . 134
7.19.4.2 cLastUpdate . . . 134
7.19.4.3 cPosition . . . 134
7.19.4.4 enumColor . . . 134
7.19.4.5 enumStatus . . . 134
7.19.4.6 enumType . . . 134
7.20 Orientation Class Reference . . . 135
7.20.1 Detailed Description . . . 135
7.20.2 Constructor & Destructor Documentation . . . 136
7.20.2.1 Orientation . . . 136
7.20.2.2 ∼Orientation . . . 136
7.20.2.3 Orientation . . . 136
7.20.3 Member Function Documentation . . . 136
7.20.3.1 angleBetweenAB . . . 136
7.20.3.2 beginOrientation . . . 136
7.20.3.3 checkObjectsOnLine . . . 136
7.20.3.4 distancePolar . . . 137
7.20.3.5 distancePolar . . . 137
7.20.3.6 getGlobalPolarPosition . . . 137
7.20.3.7 getGlobalPosition . . . 138
7.20.3.8 operator= . . . 138
7.21 PathPlanning Class Reference . . . 138
7.21.1 Detailed Description . . . 140
7.21.2 Constructor & Destructor Documentation . . . 141
7.21.2.1 PathPlanning . . . 141
7.21.2.2 ∼PathPlanning . . . 141
7.21.3 Member Function Documentation . . . 141
7.21.3.1 calculatePathCosts . . . 141
7.21.3.2 calculateWaypoints . . . 141
7.21.3.3 generateGrid . . . 141
7.21.3.4 getAvoidRestOfField . . . 141
7.21.3.5 getEnabled . . . 141
7.21.3.6 getGridRotation . . . 142
7.21.3.7 getIgnorePucks . . . 142
7.21.3.8 grid2AB . . . 142
7.21.3.9 grid2AB . . . 142
7.21.3.10 grid2XY . . . 143
7.21.3.11 grid2XY . . . 143
7.21.3.12 gridIndex . . . 143

7.21.3.13 initGridPoint . . . 143
7.21.3.14 pathDisplay . . . 144
7.21.3.15 planPath . . . 144
7.21.3.16 sendUpdatedWaypoints . . . 144
7.21.3.17 setAvoidRestOfField . . . 144
7.21.3.18 setEnabled . . . 144
7.21.3.19 setIgnorePucks . . . 144
7.21.4 Member Data Documentation . . . 144
7.21.4.1 avoidRestOfField . . . 144
7.21.4.2 enabled . . . 145
7.21.4.3 grid . . . 145
7.21.4.4 gridRotation . . . 145
7.21.4.5 gridSizeA . . . 145
7.21.4.6 gridSizeB . . . 145
7.21.4.7 gridSpacing . . . 145
7.21.4.8 ignorePucks . . . 145
7.21.4.9 maxA . . . 145
7.21.4.10 maxB . . . 145
7.21.4.11 minA . . . 145
7.21.4.12 minB . . . 146
7.21.4.13 obstaclesEnemy . . . 146
7.21.4.14 obstaclesPole . . . 146
7.21.4.15 obstaclesPuck . . . 146
7.21.4.16 robot . . . 146
7.21.4.17 robotA . . . 146
7.21.4.18 robotB . . . 146
7.21.4.19 robotRot . . . 146
7.21.4.20 robotX . . . 146
7.21.4.21 robotY . . . 147
7.21.4.22 streamPathEnabled . . . 147
7.21.4.23 targetA . . . 147
7.21.4.24 targetB . . . 147
7.21.4.25 targetRot . . . 147
7.21.4.26 targetX . . . 147
7.21.4.27 targetY . . . 147
7.21.4.28 timer . . . 147
7.22 PathPlotData Struct Reference . . . 148
7.22.1 Detailed Description . . . 148
7.22.2 Member Enumeration Documentation . . . 148
7.22.2.1 DataTypeEnum . . . 148

7.22.2.2 DataTypeEnum . . . 149
7.22.3 Constructor & Destructor Documentation . . . 149
7.22.3.1 PathPlotData . . . 149
7.22.3.2 PathPlotData . . . 149
7.22.4 Member Data Documentation . . . 149
7.22.4.1 data . . . 149
7.22.4.2 dataSizeX . . . 149
7.22.4.3 dataSizeY . . . 149
7.22.4.4 dataType . . . 150
7.22.4.5 robot . . . 150
7.22.4.6 splineLength . . . 150
7.22.4.7 splineX . . . 150
7.22.4.8 splineY . . . 150
7.22.4.9 target . . . 150
7.22.4.10 waypoints . . . 150
7.23 PathRealizer Class Reference . . . 150
7.23.1 Detailed Description . . . 152
7.23.2 Member Enumeration Documentation . . . 153
7.23.2.1 StatePathProcessing . . . 153
7.23.3 Constructor & Destructor Documentation . . . 153
7.23.3.1 PathRealizer . . . 153
7.23.3.2 ~PathRealizer . . . 153
7.23.4 Member Function Documentation . . . 153
7.23.4.1 constrainAngle . . . 153
7.23.4.2 getDistance . . . 153
7.23.4.3 getVelocityProfile . . . 154
7.23.4.4 lowPass . . . 154
7.23.4.5 signalPIDPlot . . . 154
7.23.4.6 signalSendRobotControlParams . . . 154
7.23.4.7 signalSplinePlot . . . 154
7.23.4.8 slotChangePIDParams . . . 154
7.23.4.9 slotMotionControl . . . 155
7.23.4.10 slotTimerSendPIDPlot . . . 155
7.23.4.11 slotUpdateWaypoints . . . 155
7.23.4.12 splineToQVector . . . 155
7.23.4.13 takeDimension . . . 155
7.23.5 Member Data Documentation . . . 155
7.23.5.1 elapsedTime . . . 156
7.23.5.2 iDeltaA . . . 156
7.23.5.3 iDeltaL . . . 156

7.23.5.4 integrationTime . . . 156
7.23.5.5 internalWP . . . 156
7.23.5.6 lastDeltaA . . . 156
7.23.5.7 lastDeltaL . . . 156
7.23.5.8 maxWaviness . . . 156
7.23.5.9 numWP . . . 156
7.23.5.10 periodMotionControl . . . 157
7.23.5.11 PID_A_D . . . 157
7.23.5.12 PID_A_I . . . 157
7.23.5.13 PID_A_P . . . 157
7.23.5.14 PID_V_D . . . 157
7.23.5.15 PID_V_I . . . 157
7.23.5.16 PID_V_P . . . 157
7.23.5.17 pidHistDistIst . . . 157
7.23.5.18 pidHistDistSoll . . . 157
7.23.5.19 pidHistMutex . . . 158
7.23.5.20 pidHistTime . . . 158
7.23.5.21 pidHistWinkelIst . . . 158
7.23.5.22 pidHistWinkelSoll . . . 158
7.23.5.23 robotObstacle . . . 158
7.23.5.24 splineProgress . . . 158
7.23.5.25 splineX . . . 158
7.23.5.26 splineY . . . 158
7.23.5.27 state . . . 158
7.23.5.28 streamPIDEnabled . . . 159
7.23.5.29 timeOfStart . . . 159
7.23.5.30 timerMotionControl . . . 159
7.23.5.31 timerPIDPlot . . . 159
7.23.5.32 velProfile . . . 159
7.24 PIDParams Struct Reference . . . 159
7.24.1 Detailed Description . . . 159
7.24.2 Member Data Documentation . . . 160
7.24.2.1 PID_A_D . . . 160
7.24.2.2 PID_A_I . . . 160
7.24.2.3 PID_A_P . . . 160
7.24.2.4 PID_V_D . . . 160
7.24.2.5 PID_V_I . . . 160
7.24.2.6 PID_V_P . . . 160
7.25 PIDPlotData Struct Reference . . . 160
7.25.1 Detailed Description . . . 161

7.25.2 Member Data Documentation . . . 161
7.25.2.1 distanzIst . . . 161
7.25.2.2 distanzSoll . . . 161
7.25.2.3 time . . . 161
7.25.2.4 winkelIst . . . 161
7.25.2.5 winkelSoll . . . 161
7.26 PlayerX Class Reference . . . 161
7.26.1 Detailed Description . . . 162
7.26.2 Member Function Documentation . . . 162
7.26.2.1 getInstance . . . 162
7.26.2.2 getSelfpath . . . 162
7.26.2.3 startPlayer . . . 162
7.26.2.4 stopPlayerIfStarted . . . 162
7.26.3 Member Data Documentation . . . 163
7.26.3.1 didWeStartPlayerOurselves . . . 163
7.27 PathPlotData::Point Struct Reference . . . 163
7.27.1 Detailed Description . . . 163
7.27.2 Member Data Documentation . . . 163
7.27.2.1 value . . . 163
7.27.2.2 x . . . 163
7.27.2.3 y . . . 163
7.28 Position Class Reference . . . 164
7.28.1 Detailed Description . . . 165
7.28.2 Constructor & Destructor Documentation . . . 165
7.28.2.1 Position . . . 165
7.28.2.2 Position . . . 165
7.28.2.3 Position . . . 165
7.28.2.4 Position . . . 165
7.28.2.5 Position . . . 166
7.28.2.6 Position . . . 166
7.28.3 Member Function Documentation . . . 166
7.28.3.1 certainty . . . 166
7.28.3.2 getDistanceTo . . . 166
7.28.3.3 isConsimilarTo . . . 166
7.28.3.4 isConsimilarTo . . . 167
7.28.3.5 isPositionInStartField . . . 167
7.28.3.6 operator== . . . 167
7.28.3.7 rot . . . 167
7.28.3.8 rot . . . 168
7.28.3.9 setCertainty . . . 168

7.28.3.10 sizeType . . . 168
7.28.3.11 sizeType . . . 168
7.28.3.12 x . . . 168
7.28.3.13 x . . . 169
7.28.3.14 y . . . 169
7.28.3.15 y . . . 169
7.28.4 Member Data Documentation . . . 169
7.28.4.1 m_certainty . . . 169
7.28.4.2 m_rot . . . 169
7.28.4.3 m_size . . . 169
7.28.4.4 m_x . . . 169
7.28.4.5 m_y . . . 170
7.29 Referee Class Reference . . . 170
7.29.1 Detailed Description . . . 171
7.29.2 Constructor & Destructor Documentation . . . 171
7.29.2.1 Referee . . . 171
7.29.2.2 ~Referee . . . 172
7.29.3 Member Function Documentation . . . 172
7.29.3.1 abValues . . . 172
7.29.3.2 connected . . . 172
7.29.3.3 connectFailed . . . 172
7.29.3.4 connectToServer . . . 172
7.29.3.5 detectionStart . . . 172
7.29.3.6 disconnected . . . 172
7.29.3.7 gameOver . . . 172
7.29.3.8 gameStart . . . 172
7.29.3.9 isConnected . . . 172
7.29.3.10 isVerbose . . . 173
7.29.3.11 reportDone . . . 173
7.29.3.12 reportGoal . . . 173
7.29.3.13 reportReady . . . 173
7.29.3.14 sendAlive . . . 173
7.29.3.15 setVerbose . . . 173
7.29.3.16 slotConnected . . . 173
7.29.3.17 slotDisconnected . . . 173
7.29.3.18 slotRead . . . 173
7.29.3.19 stopMovement . . . 174
7.29.3.20 tellAbRatio . . . 174
7.29.3.21 tellEgoPos . . . 174
7.29.3.22 tellTeamColor . . . 174

7.29.3.23 trueColorOfTeam . . . 174
7.29.4 Member Data Documentation . . . 174
7.29.4.1 connection . . . 174
7.29.4.2 messageSize . . . 174
7.29.4.3 messengerOfTheGods . . . 174
7.29.4.4 myTeamID . . . 175
7.29.4.5 ready . . . 175
7.29.4.6 testMode . . . 175
7.29.4.7 verbose . . . 175
7.29.4.8 wLimit . . . 175
7.30 RobotThread Class Reference . . . 175
7.30.1 Detailed Description . . . 176
7.30.2 Constructor & Destructor Documentation . . . 176
7.30.2.1 RobotThread . . . 176
7.30.2.2 ∼RobotThread . . . 176
7.30.2.3 RobotThread . . . 176
7.30.2.4 ∼RobotThread . . . 177
7.30.3 Member Function Documentation . . . 177
7.30.3.1 getPathPlanning . . . 177
7.30.3.2 getPathPlanning . . . 177
7.30.4 Member Data Documentation . . . 177
7.30.4.1 actorHighLevel . . . 177
7.30.4.2 actorLowLevel . . . 177
7.30.4.3 cam . . . 177
7.30.4.4 game . . . 177
7.30.4.5 gameEngine . . . 177
7.30.4.6 mainWindow . . . 178
7.30.4.7 pathPlanner . . . 178
7.30.4.8 pathRealizer . . . 178
7.30.4.9 sensorHighLevel . . . 178
7.30.4.10 sensorLowLevel . . . 178
7.30.4.11 threadActorHighLevel . . . 178
7.30.4.12 threadCam . . . 178
7.30.4.13 threadGame . . . 178
7.30.4.14 threadGameEngine . . . 178
7.30.4.15 threadPathPlanner . . . 179
7.30.4.16 threadPathRealizer . . . 179
7.30.4.17 threadRobotLowLevel . . . 179
7.30.4.18 threadSensorHighLevel . . . 179
7.31 SensorHighLevel Class Reference . . . 179

7.31.1 Detailed Description . . . 182
7.31.2 Constructor & Destructor Documentation . . . 182
7.31.2.1 SensorHighLevel . . . 182
7.31.2.2 ~SensorHighLevel . . . 182
7.31.2.3 SensorHighLevel . . . 182
7.31.2.4 ~SensorHighLevel . . . 182
7.31.3 Member Function Documentation . . . 182
7.31.3.1 avoideCollision . . . 182
7.31.3.2 avoideCollision . . . 182
7.31.3.3 calculateObjCenter . . . 182
7.31.3.4 calculateObjCenter . . . 182
7.31.3.5 constrainData . . . 183
7.31.3.6 constrainData . . . 183
7.31.3.7 convertPolToGlobalCoordinates . . . 183
7.31.3.8 distanceKartesisch . . . 183
7.31.3.9 distancePolar . . . 183
7.31.3.10 driveToPreposition . . . 183
7.31.3.11 extractObjects . . . 183
7.31.3.12 extractObjects . . . 183
7.31.3.13 getLaserData . . . 184
7.31.3.14 getSonarData . . . 184
7.31.3.15 getState . . . 184
7.31.3.16 getState . . . 184
7.31.3.17 getTeamColor . . . 184
7.31.3.18 getTeamColor . . . 184
7.31.3.19 puckGrabbed . . . 184
7.31.3.20 recognition . . . 185
7.31.3.21 recognition . . . 185
7.31.3.22 sendOdometryData . . . 185
7.31.3.23 setState . . . 185
7.31.3.24 setState . . . 185
7.31.3.25 setTeamColor . . . 185
7.31.3.26 setTeamColor . . . 185
7.31.3.27 signalEmergencyStopEnabled . . . 185
7.31.3.28 signalEmergencyStopEnabled . . . 185
7.31.3.29 signalPlanNewPath . . . 185
7.31.3.30 signalSendLaserData . . . 186
7.31.3.31 signalSendLaserData . . . 186
7.31.3.32 signalSendOdometryData . . . 186
7.31.3.33 signalSendRobotControlParams . . . 186

7.31.3.34 signalSendRobotControlParams . . . 186
7.31.3.35 signalSendTeamColor . . . 186
7.31.3.36 signalSendTeamColor . . . 186
7.31.3.37 signalStartColorDetection . . . 186
7.31.3.38 signalStartColorDetection . . . 186
7.31.3.39 slotColorDetected . . . 186
7.31.3.40 slotColorDetected . . . 186
7.31.3.41 slotGetLaserData . . . 186
7.31.3.42 slotSetFilterParams . . . 186
7.31.3.43 slotSetFilterParams . . . 186
7.31.3.44 slotStartDetection . . . 186
7.31.3.45 slotStartDetection . . . 186
7.31.4 Member Data Documentation . . . 186
7.31.4.1 collectorObj . . . 186
7.31.4.2 constrainedData . . . 187
7.31.4.3 counter . . . 187
7.31.4.4 cPolePositions . . . 187
7.31.4.5 currentObjects . . . 187
7.31.4.6 currentPosition . . . 187
7.31.4.7 currentState . . . 187
7.31.4.8 dummyAngleOffset . . . 187
7.31.4.9 dummyPosition . . . 187
7.31.4.10 filterParameter . . . 187
7.31.4.11 hadEmergency . . . 188
7.31.4.12 isInSlowTurn . . . 188
7.31.4.13 maxAngle . . . 188
7.31.4.14 minAngle . . . 188
7.31.4.15 mutexFilterParameter . . . 188
7.31.4.16 mutexFilterParameter . . . 188
7.31.4.17 mutexState . . . 188
7.31.4.18 mutexState . . . 188
7.31.4.19 mutexTeamColor . . . 188
7.31.4.20 mutexTeamColor . . . 188
7.31.4.21 prepositionInitialized . . . 189
7.31.4.22 previousObjects . . . 189
7.31.4.23 previousPosition . . . 189
7.31.4.24 streamSensorEnabled . . . 189
7.31.4.25 targetsSet . . . 189
7.31.4.26 teamColor . . . 189
7.31.4.27 timerToAbondonColorDetection . . . 189

7.31.4.28 timerWaitForValidColorFrame . . . 189
7.31.4.29 timeSinceStart . . . 189
7.31.4.30 transmissionPosition . . . 189
7.32 SensorLowLevel Class Reference . . . 190
7.32.1 Detailed Description . . . 191
7.32.2 Member Enumeration Documentation . . . 191
7.32.2.1 SensorState . . . 191
7.32.3 Constructor & Destructor Documentation . . . 191
7.32.3.1 SensorLowLevel . . . 191
7.32.3.2 ~SensorLowLevel . . . 191
7.32.4 Member Function Documentation . . . 191
7.32.4.1 angleWeightedAverage . . . 191
7.32.4.2 quit . . . 192
7.32.4.3 readSensorData . . . 192
7.32.4.4 run . . . 192
7.32.4.5 signalLaserDataReady . . . 192
7.32.4.6 signalLaserPlotRaw . . . 192
7.32.4.7 signalSimulationDetect . . . 192
7.32.5 Member Data Documentation . . . 192
7.32.5.1 currentOdometryPosition . . . 193
7.32.5.2 currentOdometryTime . . . 193
7.32.5.3 elapsedTimer . . . 193
7.32.5.4 laserArrayLength . . . 193
7.32.5.5 laserData . . . 193
7.32.5.6 laserProxy . . . 193
7.32.5.7 laserTime . . . 193
7.32.5.8 positionProxy . . . 193
7.32.5.9 previousOdometryPosition . . . 193
7.32.5.10 previousOdometryTime . . . 194
7.32.5.11 quitting . . . 194
7.32.5.12 robot . . . 194
7.32.5.13 state . . . 194
7.33 tkqt::spline Class Reference . . . 194
7.33.1 Detailed Description . . . 195
7.33.2 Member Function Documentation . . . 195
7.33.2.1 operator() . . . 195
7.33.2.2 set_points . . . 195
7.33.3 Member Data Documentation . . . 195
7.33.3.1 m_a . . . 195
7.33.3.2 m_b . . . 195

7.33.3.3 m_c . . . 196
7.33.3.4 m_d . . . 196
7.33.3.5 m_x . . . 196
7.33.3.6 m_y . . . 196

8 File Documentation **197**

8.1 actorhighlevel.cpp File Reference . . . 197
8.2 actorhighlevel.h File Reference . . . 197
8.2.1 Enumeration Type Documentation . . . 198
8.2.1.1 StatePathProcessing . . . 198
8.3 actorLowLevel.cpp File Reference . . . 198
8.4 actorLowLevel.h File Reference . . . 198
8.5 cam.cpp File Reference . . . 199
8.6 cam.h File Reference . . . 199
8.6.1 Enumeration Type Documentation . . . 199
8.6.1.1 CamColor . . . 199
8.7 cameraparams.h File Reference . . . 200
8.8 cameraparams.h File Reference . . . 200
8.9 constrainedlaserdata.cpp File Reference . . . 200
8.10 constrainedlaserdata.h File Reference . . . 200
8.11 cvimagewidget.h File Reference . . . 200
8.12 cvimagewidget.h File Reference . . . 201
8.13 define.h File Reference . . . 201
8.13.1 Macro Definition Documentation . . . 205
8.13.1.1 _USE_MATH_DEFINES . . . 205
8.14 dynsections.js File Reference . . . 205
8.14.1 Function Documentation . . . 205
8.14.1.1 toggleFolder . . . 205
8.14.1.2 toggleInherit . . . 205
8.14.1.3 toggleLevel . . . 205
8.14.1.4 toggleVisibility . . . 205
8.14.1.5 updateStripes . . . 206
8.15 filterparams.h File Reference . . . 206
8.16 game.cpp File Reference . . . 206
8.17 game.h File Reference . . . 206
8.18 gameengine.cpp File Reference . . . 207
8.19 gameengine.h File Reference . . . 207
8.20 hermes.cpp File Reference . . . 207
8.21 hermes.h File Reference . . . 207
8.22 hermescodes.h File Reference . . . 207

8.22.1 Macro Definition Documentation . . . 208
8.22.1.1 HERMES_A_B . . . 208
8.22.1.2 HERMES_ANGELINAFOUND . . . 208
8.22.1.3 HERMES_CONNECT . . . 208
8.22.1.4 HERMES_DATA_T1 . . . 208
8.22.1.5 HERMES_DATA_T2 . . . 208
8.22.1.6 HERMES_DATA_T3 . . . 208
8.22.1.7 HERMES_DATA_T4 . . . 208
8.22.1.8 HERMES_DETECTION_START . . . 209
8.22.1.9 HERMES_DONE . . . 209
8.22.1.10 HERMES_ERROR . . . 209
8.22.1.11 HERMES_GAME_OVER . . . 209
8.22.1.12 HERMES_GAME_START . . . 209
8.22.1.13 HERMES_KEEP_ALIVE . . . 209
8.22.1.14 HERMES_LEFT_PLAYGROUND . . . 209
8.22.1.15 HERMES_LOOKINGFOR . . . 209
8.22.1.16 HERMES_READY . . . 209
8.22.1.17 HERMES_SCORE . . . 209
8.22.1.18 HERMES_STATUS . . . 209
8.22.1.19 HERMES_STOP_MOVEMENT . . . 209
8.22.1.20 HERMES_TEAMCOLOR . . . 210
8.23 laserplotdata.h File Reference . . . 210
8.24 log.cpp File Reference . . . 210
8.25 log.h File Reference . . . 210
8.26 logparams.h File Reference . . . 210
8.27 LogParams.h File Reference . . . 211
8.28 main.cpp File Reference . . . 211
8.28.1 Function Documentation . . . 211
8.28.1.1 main . . . 211
8.29 mainwindow.cpp File Reference . . . 212
8.29.1 Variable Documentation . . . 212
8.29.1.1 timer . . . 212
8.30 mainwindow.cpp File Reference . . . 212
8.31 mainwindow.h File Reference . . . 213
8.32 mainwindow.h File Reference . . . 213
8.32.1 Variable Documentation . . . 214
8.32.1.1 logParams . . . 214
8.33 mapdata.cpp File Reference . . . 214
8.34 mapdata.h File Reference . . . 214
8.35 medianfilter.cpp File Reference . . . 214

8.36 medianfilter.h File Reference . . . 214
8.37 medianfilter_new.cpp File Reference . . . 215
8.38 medianfilter_new.h File Reference . . . 215
8.39 obstacle.cpp File Reference . . . 215
8.40 obstacle.h File Reference . . . 215
8.40.1 Enumeration Type Documentation . . . 216
8.40.1.1 FieldArea . . . 216
8.40.1.2 ObstacleColor . . . 217
8.40.1.3 ObstacleStatus . . . 217
8.40.1.4 ObstacleType . . . 217
8.41 orientierung.cpp File Reference . . . 218
8.42 orientierung.h File Reference . . . 218
8.43 pathplanning.cpp File Reference . . . 218
8.43.1 Function Documentation . . . 218
8.43.1.1 round . . . 218
8.44 pathplanning.h File Reference . . . 219
8.45 pathplotdata.h File Reference . . . 219
8.46 pathplotdata.h File Reference . . . 219
8.47 pathRealizer.h File Reference . . . 220
8.48 pidparams.h File Reference . . . 220
8.49 pidparams.h File Reference . . . 220
8.50 pidplotdata.h File Reference . . . 220
8.51 pidplotdata.h File Reference . . . 221
8.52 player.cpp File Reference . . . 221
8.53 player.h File Reference . . . 221
8.54 position.cpp File Reference . . . 221
8.55 position.h File Reference . . . 222
8.55.1 Enumeration Type Documentation . . . 222
8.55.1.1 SizeType . . . 222
8.56 qcustomplot.cpp File Reference . . . 222
8.56.1 Detailed Description . . . 222
8.57 qcustomplot.h File Reference . . . 223
8.57.1 Detailed Description . . . 223
8.57.2 Function Documentation . . . 223
8.57.2.1 operator* . . . 223
8.57.2.2 operator* . . . 224
8.57.2.3 operator+ . . . 224
8.57.2.4 operator+ . . . 224
8.57.2.5 operator- . . . 224
8.57.2.6 operator/ . . . 224

8.57.2.7 Q_DECLARE_TYPEINFO . . . 224
8.57.2.8 Q_DECLARE_TYPEINFO . . . 224
8.57.2.9 Q_DECLARE_TYPEINFO . . . 224
8.57.2.10 Q_DECLARE_TYPEINFO . . . 224
8.57.2.11 Q_DECLARE_TYPEINFO . . . 224
8.57.2.12 Q_DECLARE_TYPEINFO . . . 224
8.57.2.13 Q_DECLARE_TYPEINFO . . . 224
8.58 referee.cpp File Reference . . . 224
8.59 referee.h File Reference . . . 224
8.59.1 Enumeration Type Documentation . . . 225
8.59.1.1 TeamColor . . . 225
8.60 robotThread.cpp File Reference . . . 225
8.61 robotThread.cpp File Reference . . . 226
8.61.1 Variable Documentation . . . 226
8.61.1.1 mainWindow . . . 226
8.62 robotThread.h File Reference . . . 226
8.63 robotThread.h File Reference . . . 226
8.64 sensor.h File Reference . . . 227
8.64.1 Enumeration Type Documentation . . . 227
8.64.1.1 SensorStates . . . 227
8.65 sensorhighlevel.cpp File Reference . . . 228
8.66 sensorhighlevel.h File Reference . . . 228
8.66.1 Enumeration Type Documentation . . . 228
8.66.1.1 SensorStates . . . 228
8.67 sensorLowLevel.cpp File Reference . . . 229
8.68 sensorLowLevel.h File Reference . . . 229
8.69 spline.cpp File Reference . . . 230
8.70 spline.h File Reference . . . 230
8.71 trilateration.cpp File Reference . . . 230
8.72 trilateration.h File Reference . . . 230

Chapter 1

Main Page

Das ist die Schnittstelle zu Angelina, dem Server. Sie kümmert sich um das Netzwerk und ermöglicht es euch, Angelina eure Ergebnisse mitzuteilen.

Einbinden der Referee-Schnittstelle: -CMake Script für libreferee.a und referee.h (./cmake/FindReferee.cmake)

Der Referee kann unabhängig vom Server benutzt werden. Im Verbose-Modus weden die Aktionen via qDebug() auf der Konsole ausgegeben. Alle Längenangaben sind in Metern! Das Programm testgui stellt eine beispielhafte Verwendung der Schnittstelle dar. Bugs in Angelina oder dem Referee bitte an die Mailingliste (cpp-tutor@ldv.ei.tum.de). Wenn möglich mit einer Beschreibung wie man sie reproduziert.

Author

Andreas Rittinger (a.rittinger@web.de)
Tim Habigt (tim@tum.de)
Julian Habigt (jh@tum.de)

Chapter 2

Todo List

Member ActorHighLevel::slotReleasePuck ()

piepsen?

Member Game::findAndSelectBestPuck ()

eigentlich würde ich hier lieber etwas besser differenzieren

Member GameEngine::slotDetectionFinished (CamColor color)

: Was tun wenn wir keine Frabe erkennen?

Member MainWindow::~MainWindow ()

segmentation fault when trying to call delete on the pointer

Member MapData::getListByType (const ObstacleType &type)

warning scheimssen

Member MapData::organizeObstacles (const Obstacle &constObstacle, const Position &currentRoboPos)

: klären wie verschiewdene obstacles gemerged werden

: warning für default state

Member Obstacle::mergeWith (const Obstacle &newObst)

merge conditions are a topic for discussion

average of coordinates

Member Obstacle::operator< (const Obstacle &b) const

check condition

Member PathPlanning::generateGrid ()

: Maybe remove a puck from the list if it is the target?

Member Position::operator== (const Position &b) const

we may change this condition here

Member RobotThread::RobotThread ()

: Das ist erstmal nur zum Debuggen drinnen.

Member SensorHighLevel::calculateObjCenter (const ConstrainedLaserData &constrainedData, int objectBeginn, int objectEnd, SensorStates &tempState)

may be cheating here with the distAB

Member SensorHighLevel::extractObjects (const ConstrainedLaserData &constrainedData)

also ich mache es hier mal von der simulation abhängig, ob 2 oder 3 werte für ein object genügen!

Member SensorHighLevel::slotGetLaserData (QVector< double > sensorData, Position positionSignal)

magic number for certainty here

consider removing the counter and replace through = new QElapsedTimer;

angle range?

consider removing the counter and replace through = new QElapsedTimer;

Member tkqt::spline::set_points (const QVector< double > &x, const QVector< double > &y, bool cubic_spline=true)

: sort x and y, rather than returning an error

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

config 11
cv 30
Filter 31
PlayerCc 31
tkqt 31
trilateration 32
Ui 33

Chapter 4

Class Index

4.1 Class List

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

ActorHighLevel
: The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER) . . . 35

ActorLowLevel
: The ActorLowLevel class Class for accessing the PlayerCc::Position2dProxy, as single point of access . . . 44

tkqt::band_matrix
: The band_matrix class, which is the basis for cubic hermite spline creation . . . 47

Cam
: Will stream cam data to GUI-panal for recognition of poles color . . . 51

CameraParams
: The CameraParams struct represents the cam params for calibration . . . 55

ConstrainedLaserData
: Datapacket containing the processed data from the LowLevelSensor and is used for sharing references of the data through the SensorHighLevel . . . 56

CVImageWidget
: The CVImageWidget class will draw the video stream directy instead of bytewise . . . 59

FilterParams . . . 61

Game . . . 62

GameEngine . . . 70

PathPlanning::GridPoint . . . 75

Hermes . . . 79

LaserPlotData
: The LaserPlotData struct is the Datapacket for plotting the laser data . . . 80

Log
: The Log class . . . 82

LogParams
: The LogParams struct describes the current logging level . . . 84

MainWindow
: Creates the GUI and connect user actions with programm functionalities for displaying and recording gathered data . . . 86

MapData
: Static class for inter-thread communication and saving information for other parts of the programm . . . 111

MedianFilter
: Will filter data with an median filter which will return the centered value of a given set . . . 124

Obstacle
Describes all objects on field as obstacles, which can be distinguish by type 125
Orientation
Try to compute the position and the orientation of the robot due to distance values and angles from sensor data 135
PathPlanning 138
PathPlotData
The PathPlotData struct is the data holder for plotting the path 148
PathRealizer
The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER) 150
PIDParams
The PIDParams struct, values for the PID controler for angular PID and velocity PID 159
PIDPlotData
The PIDPlotData struct represents the data of the PID controler of the last n-time steps 160
PlayerX
The Player class this class contains the instance of the player client to access in 'global' scope 161
PathPlotData::Point
The Point struct represent Waypoints data coming from PathPlanning 163
Position
The Position struct will represent the current pose of the robot 164
Referee
Die Schiedsrichterklasse 170
RobotThread
Responsible for the communication of all classes and is the software representation of the robot, all threads are forked there and will joinen in the end. The class will move diverent tasks to different threads 175
SensorHighLevel
Responsible for the processing of the raw laser data and adds all objects to the MapData . . . 179
SensorLowLevel
Collecting odometry and laser data from the player client 190
tkqt::spline
=> This class will create an cubic hermite spline from two given vectors 194

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

actorhighlevel.cpp . . . 197
actorhighlevel.h . . . 197
actorLowLevel.cpp . . . 198
actorLowLevel.h . . . 198
cam.cpp . . . 199
cam.h . . . 199
Sensor/cameraparams.h . . . 200
Structs/cameraparams.h . . . 200
constrainedlaserdata.cpp . . . 200
constrainedlaserdata.h . . . 200
GUI/cvimagewidget.h . . . 200
Plots/cvimagewidget.h . . . 201
define.h . . . 201
dynsections.js . . . 205
filterparams.h . . . 206
game.cpp . . . 206
game.h . . . 206
gameengine.cpp . . . 207
gameengine.h . . . 207
hermes.cpp . . . 207
hermes.h . . . 207
hermescodes.h . . . 207
laserplotdata.h . . . 210
log.cpp . . . 210
log.h . . . 210
logparams.h . . . 210
LogParams.h . . . 211
main.cpp . . . 211
GUI/mainwindow.cpp . . . 212
mainwindow.cpp . . . 212
GUI/mainwindow.h . . . 213
mainwindow.h . . . 213
mapdata.cpp . . . 214
mapdata.h . . . 214
medianfilter.cpp . . . 214
medianfilter.h . . . 214
medianfilter_new.cpp . . . 215
medianfilter_new.h . . . 215

obstacle.cpp . . . 215
obstacle.h . . . 215
orientierung.cpp . . . 218
orientierung.h . . . 218
pathplanning.cpp . . . 218
pathplanning.h . . . 219
Plots/pathplotdata.h . . . 219
Structs/pathplotdata.h . . . 219
pathRealizer.h . . . 220
Actor/pidparams.h . . . 220
Structs/pidparams.h . . . 220
Plots/pidplotdata.h . . . 220
Structs/pidplotdata.h . . . 221
player.cpp . . . 221
player.h . . . 221
position.cpp . . . 221
position.h . . . 222
qcustomplot.cpp . . . 222
qcustomplot.h . . . 223
referee.cpp . . . 224
referee.h . . . 224
Main/robotThread.cpp . . . 225
robotThread.cpp . . . 226
Main/robotThread.h . . . 226
robotThread.h . . . 226
sensor.h . . . 227
sensorhighlevel.cpp . . . 228
sensorhighlevel.h . . . 228
sensorLowLevel.cpp . . . 229
sensorLowLevel.h . . . 229
spline.cpp . . . 230
spline.h . . . 230
trilateration.cpp . . . 230
trilateration.h . . . 230

Chapter 6

Namespace Documentation

6.1 config Namespace Reference

Functions

- static const QPen guiPenOpponent (QBrush(QColor(201, 40, 27)), 4.0, Qt::SolidLine)
- static const QPen guiPenMe (QBrush(QColor(109, 191, 55)), 4.0, Qt::SolidLine)
- static const QBrush guiBrushMe (QBrush(QColor(109, 191, 55)))
- static const QPen guiPenOrient (QBrush(QColor(133, 100, 84)), 1.0, Qt::SolidLine)
- static const QPen guiPenDummy (QBrush(QColor(Qt::cyan)), 1.0, Qt::SolidLine)
- static const QPen guiPenTarget (QBrush(QColor(Qt::gray)), 1.0, Qt::SolidLine)
- static const QPen guiPenPole (QBrush(QColor(109, 191, 55)), 1.0, Qt::SolidLine)
- static const QBrush guiBrushPole (QBrush(QColor(109, 191, 55)))
- static const QPen guiPenPuckMe (QBrush(QColor(109, 191, 55)), 2.0, Qt::SolidLine)
- static const QPen guiPenPuckOpponent (QBrush(QColor(201, 40, 27)), 2.0, Qt::SolidLine)
- static const QPen guiPenPuckUndef (QBrush(QColor(127, 127, 127)), 2.0, Qt::SolidLine)
- static const QPen guiPenPuckMeOuter (QBrush(QColor(109, 191, 55, 127)), 2.0, Qt::SolidLine)
- static const QPen guiPenPuckOpponentOuter (QBrush(QColor(201, 40, 27, 127)), 2.0, Qt::SolidLine)
- static const QPen guiPenPuckUndefOuter (QBrush(QColor(127, 127, 127, 127)), 2.0, Qt::SolidLine)
- static const QPen guiPenFieldPrimary (QBrush(QColor(Qt::white)), 1.0, Qt::SolidLine)
- static const QPen guiPenFieldSecondary (QBrush(QColor(Qt::lightGray)), 1.0, Qt::SolidLine)
- static const QBrush guiBrushPuckMoving (QColor(QColor(215, 69, 232)))
- static const QBrush guiBrushPuckBlocked (QColor(QColor(Qt::black)))
- static const QBrush guiBrushGoalBlue (QColor(QColor(38, 66, 115)))
- static const QBrush guiBrushGoalYellow (QColor(QColor(255, 211, 36)))
- static const QBrush guiBrushGoalUndef (QColor(Qt::lightGray))
- static const Position gameGoalSlotL (geoGoalMarginSide-geoRobotForkCenterDist, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2.0, 0)
- static const Position gameGoalSlotM (geoGoalMarginSide+0.5 ∗geoGoalWidth, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight-geoRobotForkCenterDist, M_PI_2)
- static const Position gameGoalSlotR (geoGoalMarginSide+1.0 ∗geoGoalWidth+geoRobotForkCenterDist, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2.0, M_PI)
- static const Position gameGoalSlotOutsideLeft (0.50 ∗geoGoalMarginSide, geoFieldHeight-geoGoalMarginBottom-2 ∗geoGoalHeight, 3.0/4.0 ∗M_PI)
- static const Position gameGoalSlotOutsideRight (geoFieldWidth-0.50 ∗geoGoalMarginSide, geoFieldHeight-geoGoalMarginBottom-2 ∗geoGoalHeight, 1.0/4.0 ∗M_PI)
- static const Position gameAnnoyPositionL (geoGoalMarginSide, geoGoalMarginBottom+geoGoalHeight/2.0, 0)
- static const Position gameAnnoyPositionR (geoGoalMarginSide+geoGoalWidth, geoGoalMarginBottom+geoGoalHeight/2.0, M_PI)

Variables

- static const double geoFieldWidth = 3.0
- static const double geoFieldHeight = 5.0
- static const double geoGoalWidth = 1.0
- static const double geoGoalHeight = (5.0/3.0)/4.0
- static const double geoGoalMarginBottom = (5.0/3.0)/4.0
- static const double geoGoalMarginSide = 1.0
- static const double geoRobotForkCenterDist = 0.20
- static const double geoPol_1_2 = (5.0/3.0)/4.0
- static const double geoPol_2_3 = 0.75∗(5.0/3.0)
- static const double geoPol_3_4 = (5.0/3.0)/2.0
- static const double geoPol_1_14 = geoFieldWidth
- static const double geoPoleRadiusReal = 0.03
- static const double geoPoleRadiusSim = 0.06
- static const double geoPuckRadiusTopReal = 0.02
- static const double geoPuckRadiusTopSim = 0.03525
- static const double geoPuckRadiusBottom = 0.13 / 2.0
- static const int teamID = 7
- static const double periodAlive = 1000
- static const double periodEgoPos = 250
- static const QString refIP = "localhost"
- static const int refPort = 10000
- static const bool refVerbose = false
- static const int periodTillAnnoy = 250000
- static const int GUIopponent = 0
- static const int GUIself = 1
- static const int GUIpuckOpponent = 2
- static const int GUIpuckSelf = 3
- static const int GUIpuckUndef = 4
- static const int GUItarget = 5
- static const double GUIREMOTEVELOCITY = 0.1
- static const double GUIREMOTETURNRATE = 0.174
- static const double guiPuckKlickTolerance = 10/100.0

  wie nah muss auf das Zentrum eines Pucks in der Map geklickt werden, um ihn als geklickt wahrzunehmen. In Meter, bzw 100px
- static const int ROBOSIZE_CM = 40
- static const int PUCKSIZE_INNER_CM = 5
- static const int PUCKSIZE_OUTER_CM = 13
- static const int POLESIZE_CM = 6
- static const int BORDERTOP_CM = 75
- static const int BORDERSIDE_CM = 75
- static const int FIELDHEIGHT_CM = geoFieldHeight∗100
- static const int FIELDWIDTH_CM = geoFieldWidth∗100
- static const int TARGETSIZE_CM = 20
- static const int PATH_SHOWRES = 160
- static const double SENSOR_OUT_OF_FIELD_TOLERANCE = 0.15
- static const double SENSOR_OBJECTWIDTH_ROBO = 0.25
- static const double SENSOR_MAX_DISTANCE_OF_OBJ = 0.08
- static const double SENSOR_MAX_RANGE_ORIENTATION = 3.40
- static const double SENSOR_MAX_RANGE_RECOGNITION
- static const double SENSOR_COLLISION_AT = 0.5
- static const double SENSOR_DELTA_ANGLE = M_PI/180∗10
- static const double SENSOR_RADIUS_ROBOT = 0.23
- static const int SENSOR_WAIT_COUNTER = 10

- static const double SENSOR_MEASUREMENT_DEVIATION = 0.15
- static const double CAM_ROI_WIDTH_RELATIVE = 0.2
- static const double CAM_ROI_HEIGHT_RELATIVE = 0.3
- static const double CAM_ROI_OFFSET_HORIZONTAL_RELATIVE = 0.5 - CAM_ROI_WIDTH_RELATIVE / 2
- static const double CAM_ROI_OFFSET_VERTICAL_RELATIVE = 1 - 0.05 - CAM_ROI_HEIGHT_RELATIVE
- static const double CAM_PERCENTAGE_NON_COLOR_DETECTION = 0.5
- static const double ORIENTATION_APPROXIMATION_VALUE = 0.05
- static const double ORIENTATION_SENSOR_ODOMETRIE_DELTA = 0.03
- static const double DIST_TO_PUCK_BEFORE_GATHERING_IT = 0.40
- static const double gameWieWeitMussDerGegnerVomZielEntferntSeinImAnnoyModus = 1.0
- static const double DUMP_SLOT_2_3 = 1.042
- static const double DUMP_SLOT_3_4 = 2.083
- static const double DUMP_SLOT_4_5 = 2.917
- static const double DUMP_SLOT_5_6 = 3.958
- static const double TARGET_POLE_VARIANCE = 0.13
- static const double DISTANCE_TO_WAITING_LINE = DIST_TO_PUCK_BEFORE_GATHERING_IT
- static const double gameBisZuWelchemAbstandWirdZielwackelnGefiltert = 0.10
- static const double gameZielwackelfilterTiefpassKoeffizient = 1e-1
- static const double gameWievielBesserMussEinPuckSeinUmDasZielZuWechseln = 0.1

  in Prozent
- static const double gameMinimumDistanceEnemyToDumpSlot = 1.0

  Wie weit muss der Gegner davon entfernt sein, damit ein Dump Slot ausgewählt werden kann.
- static const double gameMinimumDistanceToEnemyRobot = 1.5 ∗ SENSOR_RADIUS_ROBOT
- static const double puckIsCloseToPoleDistance = 0.50
- static const int pathMaxWPIterations = 10000
- static const double pathGridSpacingBase = 0.05
- static const double pathPlanningEnabledUpwardsOfThisDistance = 0.10
- static const double pathArenaMinX = 0
- static const double pathArenaMaxX = geoFieldWidth
- static const double pathArenaMinY = 0
- static const double pathArenaMaxY = geoFieldHeight
- static const double pathArenaFieldAvoidMaxY = geoPol_1_2 + geoPol_2_3
- static const double pathRobotRadius = 0.26
- static const double pathPoleCloseDist = pathRobotRadius + 0.10 + geoPoleRadiusReal
- static const double pathPoleCloseCost = 100.0
- static const double pathPoleStartCost = 5.0
- static const double pathPoleFarDist = pathPoleCloseDist + 0.2
- static const double pathEnemyCloseDist = pathRobotRadius + 0.10 + pathRobotRadius
- static const double pathEnemyCloseCost = 10.0
- static const double pathEnemyStartCost = 1.0
- static const double pathEnemyFarDist = pathEnemyCloseDist + 0.30
- static const double pathPuckCloseDist = pathRobotRadius + 0.01 + geoPuckRadiusBottom
- static const double pathPuckCloseCost = 0.1
- static const double pathPuckStartCost = 0.05
- static const double pathPuckFarDist = pathPuckCloseDist + 0.10
- static const double pathTargetApproachAngleInfluenceDistance = 0.0
- static const double pathTargetApproachAngleMaxDeviationWithoutFullCost = 15.0 ∗M_PI/180.0
- static const double pathTargetApproachAngleFullCost = 100.0
- static const double pathAdjacencyMultiplier = 50.0
- static const double actorWaypointReachedDistance = 0.08

  Theshold in m to destination is reached.
- static const double actorWaypointReachedDiffChange = 0
- static const double actorWaypointMaxAngleDeviation = 2.5 /180.0∗M_PI

Wie weit darf der Roboterwinkel vom Zielwinkel abweichen um noch als erreicht zu gelten.

- static const double actorDistanceOfTargetOnSpline = 0.2/ pathGridSpacingBase

  Wie weit soll der PID-Sollpunkt dem der Roboter hinterherfährt auf dem Spline maximal entfernt sein (in #-Wegpunkten, muss keine ganze Zahl sein)
- static const double actorGatherPuckDistance

  Wie weit wird vorwärts gefahren um einen Puck aufzunehmen.
- static const double actorReleasePuckDistance = 0.25

  Wie weit wird beim Puck loslassen zurückgefahren.
- static const double actorPushPuckDistance = 0.20

  Wie weit wird der Puck aus der Arena gefahren.
- static const double actorPushAndReleaseAdditionalReverseDist = 0.025

  Wie weit wird mehr zurück gefahren als vor bei push and release.
- static const double actorPeriodMotionControl = 1000.0 / 200.0

  Wie schnell wird der PID Regler ausgeführt (1000.0 / x Hz)
- static const double actorWPLowPassAlpha = 1e-20

  Koeffizient bei Wegpunkt-Tiefpass. Sollte nahe, aber nicht 0 sein. Guter Wert: (1.0 / config::actorPeriodMotionControl) / ((1.0 / config::actorPeriodMotionControl) + (config::actorPeriodMotionControl+1))
- static const double actorMinAngleLimiter = 15.0 ∗M_PI/180.0
- static const double actorMaxAngleLimiter = 60.0 ∗M_PI/180.0
- static const double actorMaxI = 10

  Maximaler PID-I-Anteil.
- static const double actorLowPass = 1e-0

  Tiefpassfilterkoeffizient für Winkel (1e-10 = stark, 1e-0 = aus)
- static const double obstacleCoordinateTolerance = 0.10
- static const int obstacleNumberOfPucks = 6
- static const int obstacleNumberOfPoles = 14
- static const double mapPolePuckFusionDistance = geoPoleRadiusReal + geoPuckRadiusBottom + 0.20
- static const double mapPuckPuckFusionDistance = 3 ∗ geoPuckRadiusBottom

  Abstand, bei dem zwei Pucks zu einem zusammengefasst werden (m). (Wenn der Abstand zwischen zwei Pucks exakt 2∗Puckradius ist, berühren sie sich bereits)
- static const double mapIgnorePuckInsideEnemyDistance = SENSOR_RADIUS_ROBOT + 0.5 ∗ geoPuckRadiusBottom

  Wenn ein Puck innerhalb diesen Abstands vom Gegner erkannt wird, ist es gar kein Puck.
- static const double mapAbstandRoboterZentrumZuGabel = 0.20
- static const double mapToleranzBisWohinEinPuckInDerGabelIstMIN = 0.17
- static const double mapToleranzBisWohinEinPuckInDerGabelIstMAX = 0.27
- static const bool enableDebugMapData = false
- static const bool enableDebugOrientation = false
- static const bool enableDebugActorLowLevel = false
- static const bool enableDebugActorHighLevel = false
- static const bool enableDebugSensorLowLevel = false
- static const bool enableDebugSensorHighLevel = false
- static const bool enableDebugPathPlanning = false
- static const bool enableDebugGame = false
- static const bool enableDebugMainwindow = false
- static const bool enableDebugCam = false

6.1.1 Detailed Description

This namespace will cover the static const declaration from global namespace

6.1.2 Function Documentation

6.1.2.1 static const **Position** config::gameAnnoyPositionL (geoGoalMarginSide , geoGoalMarginBottom+geoGoalHeight/2. *0,* 0) [static]

A position slot in order to annoy the enemy while standing in my goal area

6.1.2.2 static const **Position** config::gameAnnoyPositionR (geoGoalMarginSide+ *geoGoalWidth,* geoGoalMarginBottom+geoGoalHeight/2. *0,* M_PI) [static]

A position slot in order to annoy the enemy while standing in my goal area

6.1.2.3 static const **Position** config::gameGoalSlotL (geoGoalMarginSide- *geoRobotForkCenterDist,* geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2. *0,* 0) [static]

A position slot on the left side of the enemy goal area to place puck

6.1.2.4 static const **Position** config::gameGoalSlotM (geoGoalMarginSide+0.5 ∗ *geoGoalWidth,* geoFieldHeight-geoGoalMarginBottom-geoGoalHeight- *geoRobotForkCenterDist,* M_PI_2) [static]

A position slot in the middle of the enemy goal area to place puck

6.1.2.5 static const **Position** config::gameGoalSlotOutsideLeft (0.50 ∗ *geoGoalMarginSide,* geoFieldHeight-geoGoalMarginBottom-2 ∗ *geoGoalHeight,* 3.0/4.0 ∗ *M_PI*) [static]

A position slot outside of the enemy goal area to place puck

6.1.2.6 static const **Position** config::gameGoalSlotOutsideRight (geoFieldWidth-0.50 ∗ *geoGoalMarginSide,* geoFieldHeight-geoGoalMarginBottom-2 ∗ *geoGoalHeight,* 1.0/4.0 ∗ *M_PI*) [static]

A position slot outside of the enemy goal area to place puck

6.1.2.7 static const **Position** config::gameGoalSlotR (geoGoalMarginSide+1.0 ∗geoGoalWidth+ *geoRobotForkCenterDist,* geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2. *0,* M_PI) [static]

A position slot on the right side of the enemy goal area to place puck

6.1.2.8 static const QBrush config::guiBrushGoalBlue (QColor(QColor(38, 66, 115))) [static]

6.1.2.9 static const QBrush config::guiBrushGoalUndef (QColor(Qt::lightGray)) [static]

6.1.2.10 static const QBrush config::guiBrushGoalYellow (QColor(QColor(255, 211, 36))) [static]

6.1.2.11 static const QBrush config::guiBrushMe (QBrush(QColor(109, 191, 55))) [static]

6.1.2.12 static const QBrush config::guiBrushPole (QBrush(QColor(109, 191, 55))) [static]

6.1.2.13 static const QBrush config::guiBrushPuckBlocked (QColor(QColor(Qt::black))) [static]

6.1.2.14 static const QBrush config::guiBrushPuckMoving (QColor(QColor(215, 69, 232))) [static]

6.1.2.15 static const QPen config::guiPenDummy (QBrush(QColor(Qt::cyan)) , 1. *0,* Qt::SolidLine) [static]

6.1.2.16 static const QPen config::guiPenFieldPrimary (QBrush(QColor(Qt::white)) , 1. *0,* Qt::SolidLine) [static]

6.1.2.17 static const QPen config::guiPenFieldSecondary (QBrush(QColor(Qt::lightGray)) , 1. *0,* Qt::SolidLine) [static]

6.1.2.18 static const QPen config::guiPenMe (QBrush(QColor(109, 191, 55)) , 4. *0,* Qt::SolidLine) [static]

6.1.2.19 static const QPen config::guiPenOpponent (QBrush(QColor(201, 40, 27)) , 4. *0,* Qt::SolidLine) [static]

6.1.2.20 static const QPen config::guiPenOrient (QBrush(QColor(133, 100, 84)) , 1. *0,* Qt::SolidLine) [static]

6.1.2.21 static const QPen config::guiPenPole (QBrush(QColor(109, 191, 55)) , 1. *0,* Qt::SolidLine) [static]

6.1.2.22 static const QPen config::guiPenPuckMe (QBrush(QColor(109, 191, 55)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.23 static const QPen config::guiPenPuckMeOuter (QBrush(QColor(109, 191, 55, 127)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.24 static const QPen config::guiPenPuckOpponent (QBrush(QColor(201, 40, 27)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.25 static const QPen config::guiPenPuckOpponentOuter (QBrush(QColor(201, 40, 27, 127)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.26 static const QPen config::guiPenPuckUndef (QBrush(QColor(127, 127, 127)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.27 static const QPen config::guiPenPuckUndefOuter (QBrush(QColor(127, 127, 127, 127)) , 2. *0,* Qt::SolidLine) [static]

6.1.2.28 static const QPen config::guiPenTarget (QBrush(QColor(Qt::gray)) , 1. *0,* Qt::SolidLine) [static]

6.1.3 Variable Documentation

6.1.3.1 const double config::actorDistanceOfTargetOnSpline = 0.2/ pathGridSpacingBase [static]

Wie weit soll der PID-Sollpunkt dem der Roboter hinterherfährt auf dem Spline maximal entfernt sein (in #-Wegpunkten, muss keine ganze Zahl sein)

Definition at line 220 of file define.h.

6.1.3.2 const double config::actorGatherPuckDistance [static]

Initial value:

```
DIST_TO_PUCK_BEFORE_GATHERING_IT -
                                        geoRobotForkCenterDist
      + 0.05
```

Wie weit wird vorwärts gefahren um einen Puck aufzunehmen.

Definition at line 221 of file define.h.

6.1.3.3 const double config::actorLowPass = 1e-0 [static]

Tiefpassfilterkoeffizient für Winkel (1e-10 = stark, 1e-0 = aus)

Definition at line 231 of file define.h.

6.1.3.4 const double config::actorMaxAngleLimiter = 60.0 ∗M_PI/180.0 [static]

Definition at line 229 of file define.h.

6.1.3.5 const double config::actorMaxI = 10 [static]

Maximaler PID-I-Anteil.

Definition at line 230 of file define.h.

6.1.3.6 const double config::actorMinAngleLimiter = 15.0 ∗M_PI/180.0 [static]

Definition at line 228 of file define.h.

6.1.3.7 const double config::actorPeriodMotionControl = 1000.0 / 200.0 [static]

Wie schnell wird der PID Regler ausgeführt (1000.0 / x Hz)

Definition at line 226 of file define.h.

6.1.3.8 const double config::actorPushAndReleaseAdditionalReverseDist = 0.025 [static]

Wie weit wird mehr zurück gefahren als vor bei push and release.

Definition at line 225 of file define.h.

6.1.3.9 const double config::actorPushPuckDistance = 0.20 [static]

Wie weit wird der Puck aus der Arena gefahren.

Definition at line 224 of file define.h.

6.1.3.10 const double config::actorReleasePuckDistance = 0.25 [static]

Wie weit wird beim Puck loslassen zurückgefahren.

Definition at line 223 of file define.h.

6.1.3.11 const double config::actorWaypointMaxAngleDeviation = 2.5 /180.0∗M_PI [static]

Wie weit darf der Roboterwinkel vom Zielwinkel abweichen um noch als erreicht zu gelten.

Definition at line 219 of file define.h.

6.1.3.12 const double config::actorWaypointReachedDiffChange = 0 [static]

Definition at line 218 of file define.h.

6.1.3.13 const double config::actorWaypointReachedDistance = 0.08 [static]

Theshold in m to destination is reached.

Definition at line 217 of file define.h.

6.1.3.14 const double config::actorWPLowPassAlpha = 1e-20 [static]

Koeffizient bei Wegpunkt-Tiefpass. Sollte nahe, aber nicht 0 sein. Guter Wert: (1.0 / config::actorPeriodMotionControl) / ((1.0 / config::actorPeriodMotionControl) + (config::actorPeriodMotionControl+1))

Definition at line 227 of file define.h.

6.1.3.15 const int config::BORDERSIDE_CM = 75 [static]

Distance between (side) in cm

Definition at line 90 of file define.h.

6.1.3.16 const int config::BORDERTOP_CM = 75 [static]

Distance between start region (right and left) in cm

Definition at line 89 of file define.h.

6.1.3.17 const double config::CAM_PERCENTAGE_NON_COLOR_DETECTION = 0.5 [static]

With 50% non-color you cannot recognize any color at all

Definition at line 122 of file define.h.

6.1.3.18 const double config::CAM_ROI_HEIGHT_RELATIVE = 0.3 [static]

Height of ROI for color detection

Definition at line 119 of file define.h.

6.1.3.19 const double config::CAM_ROI_OFFSET_HORIZONTAL_RELATIVE = 0.5 - CAM_ROI_WIDTH_RELATIVE / 2 [static]

OFFSET for the height of ROI (relative to center of picture)

Definition at line 120 of file define.h.

6.1.3.20 const double config::CAM_ROI_OFFSET_VERTICAL_RELATIVE = 1 - 0.05 - CAM_ROI_HEIGHT_RELATIVE [static]

OFFSET for the height of ROI (relative to center of picture)

Definition at line 121 of file define.h.

6.1.3.21 const double config::CAM_ROI_WIDTH_RELATIVE = 0.2 [static]

Width of ROI for color detection

Definition at line 118 of file define.h.

6.1.3.22 const double config::DIST_TO_PUCK_BEFORE_GATHERING_IT = 0.40 [static]

distance from robot center to puck

Definition at line 134 of file define.h.

6.1.3.23 const double config::DISTANCE_TO_WAITING_LINE = DIST_TO_PUCK_BEFORE_GATHERING_IT [static]

m die vor der neutralen Zone eingehalten werden

Definition at line 169 of file define.h.

6.1.3.24 const double config::DUMP_SLOT_2_3 = 1.042 [static]

y-Value for the SLOT between the Pole 2&3

Definition at line 164 of file define.h.

6.1.3.25 const double config::DUMP_SLOT_3_4 = 2.083 [static]

y-Value for the SLOT between the Pole 3&4

Definition at line 165 of file define.h.

6.1.3.26 const double config::DUMP_SLOT_4_5 = 2.917 [static]

y-Value for the SLOT between the Pole 4&5

Definition at line 166 of file define.h.

6.1.3.27 const double config::DUMP_SLOT_5_6 = 3.958 [static]

y-Value for the SLOT between the Pole 5&6

Definition at line 167 of file define.h.

6.1.3.28 const bool config::enableDebugActorHighLevel = false [static]

Enable Debug information for actorHighLevel

Definition at line 257 of file define.h.

6.1.3.29 const bool config::enableDebugActorLowLevel = false [static]

Enable Debug information for actorLowLevel

Definition at line 256 of file define.h.

6.1.3.30 const bool config::enableDebugCam = false [static]

Enable Debug information for cam.cpp

Definition at line 263 of file define.h.

6.1.3.31 const bool config::enableDebugGame = false [static]

Enable Debug information for Game.cpp

Definition at line 261 of file define.h.

6.1.3.32 const bool config::enableDebugMainwindow = false [static]

Enable Debug information for mainwindow.cpp

Definition at line 262 of file define.h.

6.1.3.33 const bool config::enableDebugMapData = false [static]

Enable Debug information for MapData

Definition at line 254 of file define.h.

6.1.3.34 const bool config::enableDebugOrientation = false [static]

Enable Debug information for orientation

Definition at line 255 of file define.h.

6.1.3.35 const bool config::enableDebugPathPlanning = false [static]

Enable Debug information for PathPlanning

Definition at line 260 of file define.h.

6.1.3.36 const bool config::enableDebugSensorHighLevel = false [static]

Enable Debug information for sensorHighLevel

Definition at line 259 of file define.h.

6.1.3.37 const bool config::enableDebugSensorLowLevel = false [static]

Enable Debug information for sensorLowLevel

Definition at line 258 of file define.h.

6.1.3.38 const int config::FIELDHEIGHT_CM = geoFieldHeight∗100 [static]

Field height in cm

Definition at line 91 of file define.h.

6.1.3.39 const int config::FIELDWIDTH_CM = geoFieldWidth∗100 [static]

Field width in cm

Definition at line 92 of file define.h.

6.1.3.40 const double config::gameBisZuWelchemAbstandWirdZielwackelnGefiltert = 0.10 [static]

Definition at line 170 of file define.h.

6.1.3.41 const double config::gameMinimumDistanceEnemyToDumpSlot = 1.0 [static]

Wie weit muss der Gegner davon entfernt sein, damit ein Dump Slot ausgewählt werden kann.

Definition at line 173 of file define.h.

6.1.3.42 const double config::gameMinimumDistanceToEnemyRobot = 1.5 * SENSOR_RADIUS_ROBOT [static]

Definition at line 174 of file define.h.

6.1.3.43 const double config::gameWievielBesserMussEinPuckSeinUmDasZielZuWechseln = 0.1 [static]

in Prozent

Definition at line 172 of file define.h.

6.1.3.44 const double config::gameWieWeitMussDerGegnerVomZielEntferntSeinImAnnoyModus = 1.0 [static]

Definition at line 163 of file define.h.

6.1.3.45 const double config::gameZielwackelfilterTiefpassKoeffizient = 1e-1 [static]

Definition at line 171 of file define.h.

6.1.3.46 const double config::geoFieldHeight = 5.0 [static]

Field height in m

Definition at line 22 of file define.h.

6.1.3.47 const double config::geoFieldWidth = 3.0 [static]

Field width in m

Definition at line 21 of file define.h.

6.1.3.48 const double config::geoGoalHeight = (5.0/3.0)/4.0 [static]

Height of the colored goal area

Definition at line 24 of file define.h.

6.1.3.49 const double config::geoGoalMarginBottom = (5.0/3.0)/4.0 [static]

Definition at line 25 of file define.h.

6.1.3.50 const double config::geoGoalMarginSide = 1.0 [static]

Definition at line 26 of file define.h.

6.1.3.51 const double config::geoGoalWidth = 1.0 [static]

Width of the colored goal area

Definition at line 23 of file define.h.

6.1.3.52 const double config::geoPol_1_14 = geoFieldWidth [static]

b distance between 1&14 ; 2&13 ; 3&12 and 4&11

Definition at line 31 of file define.h.

6.1.3.53 const double config::geoPol_1_2 = (5.0/3.0)/4.0 [static]

a/4 distance between pole 1&2 and 13&14

Definition at line 28 of file define.h.

6.1.3.54 const double config::geoPol_2_3 = 0.75*(5.0/3.0) [static]

3a/4 distance between pole 2&3 and 12&13

Definition at line 29 of file define.h.

6.1.3.55 const double config::geoPol_3_4 = (5.0/3.0)/2.0 [static]

a/2 distance between pole 3&4 and 11&12

Definition at line 30 of file define.h.

6.1.3.56 const double config::geoPoleRadiusReal = 0.03 [static]

pole radius in m real value

Definition at line 32 of file define.h.

6.1.3.57 const double config::geoPoleRadiusSim = 0.06 [static]

pole radius in m simu value

Definition at line 33 of file define.h.

6.1.3.58 const double config::geoPuckRadiusBottom = 0.13 / 2.0 [static]

puck radius in m

Definition at line 36 of file define.h.

6.1.3.59 const double config::geoPuckRadiusTopReal = 0.02 [static]

puck radius in m real value

Definition at line 34 of file define.h.

6.1.3.60 const double config::geoPuckRadiusTopSim = 0.03525 [static]

puck radius in m simu value

Definition at line 35 of file define.h.

6.1.3.61 const double config::geoRobotForkCenterDist = 0.20 [static]

distance between laser and puckfork center from laser sensor

Definition at line 27 of file define.h.

6.1.3.62 const int config::GUIopponent = 0 [static]

Definition of the Foe

Definition at line 52 of file define.h.

6.1.3.63 const double config::guiPuckKlickTolerance = 10/100.0 [static]

wie nah muss auf das Zentrum eines Pucks in der Map geklickt werden, um ihn als geklickt wahrzunehmen. In Meter, bzw 100px

Definition at line 61 of file define.h.

6.1.3.64 const int config::GUIpuckOpponent = 2 [static]

Definition of the opponent puck

Definition at line 54 of file define.h.

6.1.3.65 const int config::GUIpuckSelf = 3 [static]

Definition of the own puck

Definition at line 55 of file define.h.

6.1.3.66 const int config::GUIpuckUndef = 4 [static]

Definition of the unknown puck

Definition at line 56 of file define.h.

6.1.3.67 const double config::GUIREMOTETURNRATE = 0.174 [static]

Remote control angular velocity

Definition at line 59 of file define.h.

6.1.3.68 const double config::GUIREMOTEVELOCITY = 0.1 [static]

Remote control tangential velocity

Definition at line 58 of file define.h.

6.1.3.69 const int config::GUIself = 1 [static]

Definition of self

Definition at line 53 of file define.h.

6.1.3.70 const int config::GUItarget = 5 [static]

Definition of the target (movement)

Definition at line 57 of file define.h.

6.1.3.71 const double config::mapAbstandRoboterZentrumZuGabel = 0.20 [static]

Definition at line 246 of file define.h.

6.1.3.72 const double config::mapIgnorePuckInsideEnemyDistance = SENSOR_RADIUS_ROBOT + 0.5 ∗ geoPuckRadiusBottom [static]

Wenn ein Puck innerhalb diesen Abstands vom Gegner erkannt wird, ist es gar kein Puck.

Definition at line 245 of file define.h.

6.1.3.73 const double config::mapPolePuckFusionDistance = geoPoleRadiusReal + geoPuckRadiusBottom + 0.20 [static]

Definition at line 243 of file define.h.

6.1.3.74 const double config::mapPuckPuckFusionDistance = 3 ∗ geoPuckRadiusBottom [static]

Abstand, bei dem zwei Pucks zu einem zusammengefasst werden (m). (Wenn der Abstand zwischen zwei Pucks exakt 2∗Puckradius ist, berühren sie sich bereits)

Definition at line 244 of file define.h.

6.1.3.75 const double config::mapToleranzBisWohinEinPuckInDerGabelIstMAX = 0.27 [static]

Maximum value for puck is in fork calculation

Definition at line 248 of file define.h.

6.1.3.76 const double config::mapToleranzBisWohinEinPuckInDerGabelIstMIN = 0.17 [static]

Minimum value for puck is in fork calculation

Definition at line 247 of file define.h.

6.1.3.77 const double config::obstacleCoordinateTolerance = 0.10 [static]

tolerance in m in which an two obstacles will recognised as one

Definition at line 236 of file define.h.

6.1.3.78 const int config::obstacleNumberOfPoles = 14 [static]

number of poles

Definition at line 238 of file define.h.

6.1.3.79 const int config::obstacleNumberOfPucks = 6 [static]

number of pucks

Definition at line 237 of file define.h.

6.1.3.80 const double config::ORIENTATION_APPROXIMATION_VALUE = 0.05 [static]

value of allowed distance variance of the poles

Definition at line 128 of file define.h.

6.1.3.81 const double config::ORIENTATION_SENSOR_ODOMETRIE_DELTA = 0.03 [static]

delta between the x,y odometry origin and the laser sensor origin

Definition at line 129 of file define.h.

6.1.3.82 const int config::PATH_SHOWRES = 160 [static]

resolution of pathplanning map

Definition at line 94 of file define.h.

6.1.3.83 const double config::pathAdjacencyMultiplier = 50.0 [static]

Definition at line 212 of file define.h.

6.1.3.84 const double config::pathArenaFieldAvoidMaxY = geoPol_1_2 + geoPol_2_3 [static]

max of y-axis in field avoidance mode

Definition at line 189 of file define.h.

6.1.3.85 const double config::pathArenaMaxX = geoFieldWidth [static]

max of x-axis

Definition at line 186 of file define.h.

6.1.3.86 const double config::pathArenaMaxY = geoFieldHeight [static]

max of y-axis

Definition at line 188 of file define.h.

6.1.3.87 const double config::pathArenaMinX = 0 [static]

min of x-axis

Definition at line 185 of file define.h.

6.1.3.88 const double config::pathArenaMinY = 0 [static]

min of y-axis

Definition at line 187 of file define.h.

6.1.3.89 const double config::pathEnemyCloseCost = 10.0 [static]

a high value that makes the robot not want to collide (close to the acceptable detour for not coming here)

Definition at line 199 of file define.h.

6.1.3.90 const double config::pathEnemyCloseDist = **pathRobotRadius** + 0.10 + **pathRobotRadius** [static]

the robot can't approach closer than this. in meters

Definition at line 198 of file define.h.

6.1.3.91 const double config::pathEnemyFarDist = **pathEnemyCloseDist** + 0.30 [static]

an obstacle does not cause costs if it is further away than this. in meters

Definition at line 201 of file define.h.

6.1.3.92 const double config::pathEnemyStartCost = 1.0 [static]

a medium value that makes the robot only go here if necessary (close to the acceptable detour for not coming here)

Definition at line 200 of file define.h.

6.1.3.93 const double config::pathGridSpacingBase = 0.05 [static]

value to use for path planning grid. smaller is more precise and computationally expensive. in meters

Definition at line 183 of file define.h.

6.1.3.94 const int config::pathMaxWPIterations = 10000 [static]

how many times to get the next gradient point to find the target, before giving up

Definition at line 182 of file define.h.

6.1.3.95 const double config::pathPlanningEnabledUpwardsOfThisDistance = 0.10 [static]

Wie Nah muss der Roboter dem Ziel sein, damit die Pfadplanung keine Punkte mehr erzeugt, sondern nur den Zielpunkt direkt ausgibt

Definition at line 184 of file define.h.

6.1.3.96 const double config::pathPoleCloseCost = 100.0 [static]

a high value that makes the robot not want to collide (close to the acceptable detour for not coming here)
Definition at line 194 of file define.h.

6.1.3.97 const double config::pathPoleCloseDist = pathRobotRadius + 0.10 + geoPoleRadiusReal [static]

the robot can't approach closer than this. in meters
Definition at line 193 of file define.h.

6.1.3.98 const double config::pathPoleFarDist = pathPoleCloseDist + 0.2 [static]

an obstacle does not cause costs if it is further away than this. in meters
Definition at line 196 of file define.h.

6.1.3.99 const double config::pathPoleStartCost = 5.0 [static]

a medium value that makes the robot only go here if necessary (close to the acceptable detour for not coming here)
Definition at line 195 of file define.h.

6.1.3.100 const double config::pathPuckCloseCost = 0.1 [static]

a high value that makes the robot not want to collide (close to the acceptable detour for not coming here)
Definition at line 204 of file define.h.

6.1.3.101 const double config::pathPuckCloseDist = pathRobotRadius + 0.01 + geoPuckRadiusBottom [static]

the robot can't approach closer than this. in meters
Definition at line 203 of file define.h.

6.1.3.102 const double config::pathPuckFarDist = pathPuckCloseDist + 0.10 [static]

an obstacle does not cause costs if it is further away than this. in meters
Definition at line 206 of file define.h.

6.1.3.103 const double config::pathPuckStartCost = 0.05 [static]

a medium value that makes the robot only go here if necessary (close to the acceptable detour for not coming here)
Definition at line 205 of file define.h.

6.1.3.104 const double config::pathRobotRadius = 0.26 [static]

Definition at line 191 of file define.h.

6.1.3.105 const double config::pathTargetApproachAngleFullCost = 100.0 [static]

Definition at line 210 of file define.h.

6.1.3.106 const double config::pathTargetApproachAngleInfluenceDistance = 0.0 [static]

Bis zu welcher Entfernung vom Ziel wird der Anfahrtswinkel die intrinsischen Kosten beeinflussen?

Definition at line 208 of file define.h.

6.1.3.107 const double config::pathTargetApproachAngleMaxDeviationWithoutFullCost = 15.0 ∗M_PI/180.0 [static]

Beeinflusst Öffnungswinkel des Anfahrtswinkel-Potentialfelds

Definition at line 209 of file define.h.

6.1.3.108 const double config::periodAlive = 1000 [static]

how often the keep alive signal should be send to server

Definition at line 42 of file define.h.

6.1.3.109 const double config::periodEgoPos = 250 [static]

how often our position will be transmitted

Definition at line 43 of file define.h.

6.1.3.110 const int config::periodTillAnnoy = 250000 [static]

how often will we update the internal time

Definition at line 47 of file define.h.

6.1.3.111 const int config::POLESIZE_CM = 6 [static]

Pole size in cm

Definition at line 88 of file define.h.

6.1.3.112 const double config::puckIsCloseToPoleDistance = 0.50 [static]

Definition at line 177 of file define.h.

6.1.3.113 const int config::PUCKSIZE_INNER_CM = 5 [static]

Puck size in cm

Definition at line 86 of file define.h.

6.1.3.114 const int config::PUCKSIZE_OUTER_CM = 13 [static]

Puck size in cm

Definition at line 87 of file define.h.

6.1.3.115 const QString config::refIP = "localhost" [static]

Angelina IP

Definition at line 44 of file define.h.

6.1.3.116 const int config::refPort = 10000 [static]

Angelina Port

Definition at line 45 of file define.h.

6.1.3.117 const bool config::refVerbose = false [static]

allow debugging-msgs of referee

Definition at line 46 of file define.h.

6.1.3.118 const int config::ROBOSIZE_CM = 40 [static]

Robots diameter in cm

Definition at line 85 of file define.h.

6.1.3.119 const double config::SENSOR_COLLISION_AT = 0.5 [static]

Recognition of collisions in m

Definition at line 108 of file define.h.

6.1.3.120 const double config::SENSOR_DELTA_ANGLE = M_PI/180*10 [static]

Definition at line 109 of file define.h.

6.1.3.121 const double config::SENSOR_MAX_DISTANCE_OF_OBJ = 0.08 [static]

maximal distance to an obj (area of interest)

Definition at line 102 of file define.h.

6.1.3.122 const double config::SENSOR_MAX_RANGE_ORIENTATION = 3.40 [static]

maximal value during the orientation stage

Definition at line 103 of file define.h.

6.1.3.123 const double config::SENSOR_MAX_RANGE_RECOGNITION [static]

Initial value:

```
sqrt(geoFieldWidth * geoFieldWidth +
                                                    geoFieldHeight
     * geoFieldHeight) +
     SENSOR_OUT_OF_FIELD_TOLERANCE
```

maximal value during the recognition stage

Definition at line 104 of file define.h.

6.1.3.124 const double config::SENSOR_MEASUREMENT_DEVIATION = 0.15 [static]

Abort if deviation is higher than Xcm

Definition at line 112 of file define.h.

6.1.3.125 const double config::SENSOR_OBJECTWIDTH_ROBO = 0.25 [static]

everything over this size seems to be the opponent

Definition at line 101 of file define.h.

6.1.3.126 const double config::SENSOR_OUT_OF_FIELD_TOLERANCE = 0.15 [static]

Region of influence from outer field

Definition at line 100 of file define.h.

6.1.3.127 const double config::SENSOR_RADIUS_ROBOT = 0.23 [static]

Angledelta for 180° rotation in sensor robot radius in m

Definition at line 110 of file define.h.

6.1.3.128 const int config::SENSOR_WAIT_COUNTER = 10 [static]

Iterations to wait

Definition at line 111 of file define.h.

6.1.3.129 const double config::TARGET_POLE_VARIANCE = 0.13 [static]

Wert um die der Target Pole verschoben sein darf

Definition at line 168 of file define.h.

6.1.3.130 const int config::TARGETSIZE_CM = 20 [static]

Target size in cm

Definition at line 93 of file define.h.

6.1.3.131 const int config::teamID = 7 [static]

groups team id

Definition at line 41 of file define.h.

6.2 cv Namespace Reference

6.3 Filter Namespace Reference

Typedefs

- typedef double element

Functions

- void _medianfilter (const element *signal, element *result, int N)
- void medianfilter (element *signal, element *result, int N)
- void _medianfilter (const element *image, element *result, int N, int M)
- void medianfilter (element *image, element *result, int N, int M)

6.3.1 Typedef Documentation

6.3.1.1 typedef double Filter::element

Definition at line 15 of file medianfilter_new.h.

6.3.2 Function Documentation

6.3.2.1 void Filter::_medianfilter (const element * *signal*, element * *result*, int *N*)

Definition at line 17 of file medianfilter_new.cpp.

6.3.2.2 void Filter::_medianfilter (const element * *image*, element * *result*, int *N*, int *M*)

Definition at line 83 of file medianfilter_new.cpp.

6.3.2.3 void Filter::medianfilter (element * *signal*, element * *result*, int *N*)

Definition at line 48 of file medianfilter_new.cpp.

6.3.2.4 void Filter::medianfilter (element * *image*, element * *result*, int *N*, int *M*)

Definition at line 118 of file medianfilter_new.cpp.

6.4 PlayerCc Namespace Reference

6.5 tkqt Namespace Reference

Classes

- class band_matrix

  The band_matrix class, which is the basis for cubic hermite spline creation.
- class spline

  The spline class => This class will create an cubic hermite spline from two given vectors.

6.5.1 Detailed Description

spline.h

simple cubic spline interpolation library without external dependencies

Copyright (C) 2011, 2014 Tino Kluge (ttk448 at gmail.com) %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
-> replace STD:: stuff with QT:: functions (Jan Ehrensperger 29.11.14) %%%%%%%%%%%%%%%%%%%%%%%%%
This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

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

You should have received a copy of the GNU General Public License

along with this program. If not, see `http://www.gnu.org/licenses/`.

6.6 trilateration Namespace Reference

Functions

- int circle_circle_intersection (double x0, double y0, double r0, double x1, double y1, double r1, double *xi, double *yi, double *xi_prime, double *yi_prime)

  circle_circle_intersection: Determine the points where 2 circles in a common plane intersect.

6.6.1 Detailed Description

See Also

source: `http://en.wikipedia.org/wiki/Talk:Trilateration#Example_C_program`

6.6.2 Function Documentation

6.6.2.1 int trilateration::circle_circle_intersection (double *x0*, double *y0*, double *r0*, double *x1*, double *y1*, double *r1*, double * *xi*, double * *yi*, double * *xi_prime*, double * *yi_prime*)

circle_circle_intersection: Determine the points where 2 circles in a common plane intersect.

See Also

paulbourke.net/geometry/circlesphere/tvoght.c

Parameters

in	*x0*	(double)
in	*y0*	(double)
in	*r0*	(double)
in	*x1*	(double)
in	*y1*	(double)
in	*r1*	(double)
in	*xi*	(double*)
in	*yi*	(double*)
in	*xi_prime*	(double*)
in	*yi_prime*	(double*)

Returns

of the error value

Definition at line 28 of file trilateration.cpp.

6.7 Ui Namespace Reference

Chapter 7

Class Documentation

7.1 ActorHighLevel Class Reference

The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

`#include <actorhighlevel.h>`

Collaboration diagram for ActorHighLevel:

Public Slots

- void slotUpdateWaypoints (QList< QPair< double, double > > waypoints)

  slotUpdateWaypoints receive computed waypoints vom AI/pathplanning. Will hard reset the waypoints if new waypoints are available.
- void slotReleasePuck ()

  slotReleasePuck, release the puck by moving backwards without pathplanning
- void slotPushAndReleasePuck (double m_releasePuckDistance)

  slotPushAndReleasePuck, move forwards to push a puck on field line, than move the same distance backwards.
- void slotGatherPuck ()

  slotGatherPuck, move forwards for gathering the puck
- void slotChangePIDParams (PIDParams p)

  slotChangePIDParams => This method will change the PID-values caused by changing the values in the GUI.

Signals

- void signalSendRobotControlParams (double velocity, double turnangle)

  signalSendRobotControlParams => This method will emit the new velocity and turnrate to the actorLowLevel class.
- void signalSplinePlot (PathPlotData pathPlotData)

  signalSplinePlot this method will emit a data-struct, to display the path in GUI
- void signalPIDPlot (PIDPlotData d)

  signalPIDPlot this method will emit a data-struct to display the current PID-values
- void signalPuckDone ()

  signalReleaseDone wird gesendet, wenn RELEASE_PUCK fertig ist (auch der Fall wenn PUSH_AND_RELEASE_PUCK eingestellt war) oder GATHER_PUCK fertig ist

Public Member Functions

- ActorHighLevel ()

  PathRealizer => default constructor intialising member variables with default values.
- ~ActorHighLevel ()

  ~PathRealizer => default destructor which will clear the heap and delete other objects
- void ignoreSignals ()

  ignoreSignals to ignore all incomming signals which would start the PIDController

Static Public Member Functions

- static StatePathProcessing getState ()

  getState
- static void setState (const StatePathProcessing &newState)

  setstate

Static Public Attributes

- static std::atomic_bool streamPIDEnabled

Private Slots

- void slotTimerSendPIDPlot ()

  slotTimerSendPIDPlot this method will update the data in GUI.

Private Member Functions

- void startPIDController ()

  startPIDController => This method represents the internal state-machine implementing a PID-controller for motion control.
- void resetPIDtempVars ()

  resetPIDtempVars
- QVector< double > lowPass (QVector< double > in, double alpha=0)

  lowPass => This method will lowpass filter the given data

Static Private Member Functions

- static const double constrainAngle (const double inRad)

  constrainAngle => This method is responsible to prohibit a phase-shift.
- static QVector< double > takeDimension (const QVector< QPair< double, double > > in, const int dimension)

  takeDimension => This method will reduce the dimensions of a given QVector of points.

Private Attributes

- std::atomic_bool quitting
- std::atomic_bool enabled
- Position robotPosition
- Position releasePuckOrigin
- QList< QPair< double, double > > internalWP

- int numWP
- tkqt::spline splineX
- tkqt::spline splineY
- bool positionWasReached
- QMutex * mutexPidHist
- QTimer * timerPIDPlot
- qint64 timeOfStart
- QList< double > pidHistTime
- QList< double > pidHistWinkelSoll
- QList< double > pidHistWinkelIst
- QList< double > pidHistDistIst
- QList< double > pidHistDistSoll
- QElapsedTimer * elapsedTime
- double PID_A_P
- double PID_A_I
- double PID_A_D
- double PID_V_P
- double PID_V_I
- double PID_V_D
- double lastDeltaA
- double iDeltaA
- double lastDeltaL
- double iDeltaL
- double targetDistLast
- double targetDistDiffLast
- double additionalReleasePuckDistance

Static Private Attributes

- static QMutex mutexState
- static StatePathProcessing state = StatePathProcessing::RUNNING

7.1.1 Detailed Description

The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

Definition at line 35 of file actorhighlevel.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ActorHighLevel::ActorHighLevel ( )

PathRealizer => default constructor intialising member variables with default values.

Definition at line 23 of file actorhighlevel.cpp.

7.1.2.2 ActorHighLevel::~ActorHighLevel ( )

~PathRealizer => default destructor which will clear the heap and delete other objects

Definition at line 57 of file actorhighlevel.cpp.

7.1.3 Member Function Documentation

7.1.3.1 const double ActorHighLevel::constrainAngle (const double *inRad*) [static],[private]

constrainAngle => This method is responsible to prohibit a phase-shift.

Parameters

in	*inRad*	(double) angle in rad which should be checked

Returns

the corrected angle in rad as double

Definition at line 107 of file actorhighlevel.cpp.

7.1.3.2 StatePathProcessing ActorHighLevel::getState ( ) [static]

getState

Returns

Definition at line 695 of file actorhighlevel.cpp.

7.1.3.3 void ActorHighLevel::ignoreSignals ( )

ignoreSignals to ignore all incomming signals which would start the PIDController

Definition at line 65 of file actorhighlevel.cpp.

7.1.3.4 QVector< double > ActorHighLevel::lowPass (QVector< double > *in,* double *alpha* = 0) [private]

lowPass => This method will lowpass filter the given data

Parameters

in	*in*	(QVector<double>) vector of values
in	*alpha*	(double) weighting of previous values during iteration

Returns

QVector<double> as filtered values

Definition at line 71 of file actorhighlevel.cpp.

7.1.3.5 void ActorHighLevel::resetPIDtempVars ( ) [private]

resetPIDtempVars

Definition at line 685 of file actorhighlevel.cpp.

7.1.3.6 void ActorHighLevel::setState (const StatePathProcessing & *newState*) [static]

setstate

Parameters

newState	

Definition at line 701 of file actorhighlevel.cpp.

7.1.3.7 void ActorHighLevel::signalPIDPlot (PIDPlotData *d*) [signal]

signalPIDPlot this method will emit a data-struct to display the current PID-values

Parameters

in	*d*	(struct of QList<double>) an struct with data for the GUI plot of PID-values.

7.1.3.8 void ActorHighLevel::signalPuckDone ( ) [signal]

signalReleaseDone wird gesendet, wenn RELEASE_PUCK fertig ist (auch der Fall wenn PUSH_AND_RELEASE-_PUCK eingestellt war) oder GATHER_PUCK fertig ist

7.1.3.9 void ActorHighLevel::signalSendRobotControlParams (double *velocity*, double *turnangle*) [signal]

signalSendRobotControlParams => This method will emit the new velocity and turnrate to the actorLowLevel class.

Parameters

in	*velocity*	(double) m/s
in	*turnangle*	(double) rad/s

7.1.3.10 void ActorHighLevel::signalSplinePlot (PathPlotData *pathPlotData*) [signal]

signalSplinePlot this method will emit a data-struct, to display the path in GUI

Parameters

in	*pathPlotData*	(PathPlotData) an struct which included the values of the GUI tab to display the path.

7.1.3.11 void ActorHighLevel::slotChangePIDParams (PIDParams *p*) [slot]

slotChangePIDParams => This method will change the PID-values caused by changing the values in the GUI.

Parameters

in	*p*	(struct of double values) for changing the PID-controler values.

Definition at line 650 of file actorhighlevel.cpp.

7.1.3.12 void ActorHighLevel::slotGatherPuck ( ) [slot]

slotGatherPuck, move forwards for gathering the puck

Definition at line 254 of file actorhighlevel.cpp.

7.1.3.13 void ActorHighLevel::slotPushAndReleasePuck (double *m_releasePuckDistance*) [slot]

slotPushAndReleasePuck, move forwards to push a puck on field line, than move the same distance backwards.

Parameters

in	*m_releasePuck-Distance*	(double)

Definition at line 233 of file actorhighlevel.cpp.

7.1.3.14 void ActorHighLevel::slotReleasePuck ( ) [slot]

slotReleasePuck, release the puck by moving backwards without pathplanning

Todo piepsen?

Definition at line 213 of file actorhighlevel.cpp.

7.1.3.15 void ActorHighLevel::slotTimerSendPIDPlot ( ) [private], [slot]

slotTimerSendPIDPlot this method will update the data in GUI.

Definition at line 660 of file actorhighlevel.cpp.

7.1.3.16 void ActorHighLevel::slotUpdateWaypoints (QList< QPair< double, double > > *waypoints*) [slot]

slotUpdateWaypoints receive computed waypoints vom AI/pathplanning. Will hard reset the waypoints if new waypoints are available.

Parameters

in	*waypoints*	(QList of QPair of <double,double>) representing the given waypoints in (x,y).

Definition at line 134 of file actorhighlevel.cpp.

7.1.3.17 void ActorHighLevel::startPIDController ( ) [private]

startPIDController => This method represents the internal state-machine implementing a PID-controller for motion control.

hier sollte doch 20cm unten und oben stehen?

Definition at line 275 of file actorhighlevel.cpp.

7.1.3.18 QVector< double > ActorHighLevel::takeDimension (const QVector< QPair< double, double > > *in*, const int *dimension*) [static], [private]

takeDimension => This method will reduce the dimensions of a given QVector of points.

Parameters

in	*in*	(QVector<QPair<double,double>> multi-dimensional array of points
in	*dimension*	(int) the desired dimension which should returned.

Returns

an empty QVector<double> if length of QVector is equal to zero, otherwise the flatten (one dimensional) QVector<double>

Definition at line 118 of file actorhighlevel.cpp.

7.1.4 Member Data Documentation

7.1.4.1 double ActorHighLevel::additionalReleasePuckDistance [private]

Definition at line 181 of file actorhighlevel.h.

7.1.4.2 QElapsedTimer* ActorHighLevel::elapsedTime [private]

member object for determination of new time delta

Definition at line 166 of file actorhighlevel.h.

7.1.4.3 std::atomic_bool ActorHighLevel::enabled [private]

Definition at line 140 of file actorhighlevel.h.

7.1.4.4 double ActorHighLevel::iDeltaA [private]

desired angle derivation

Definition at line 176 of file actorhighlevel.h.

7.1.4.5 double ActorHighLevel::iDeltaL [private]

desiered length derivation

Definition at line 178 of file actorhighlevel.h.

7.1.4.6 QList< QPair<double,double> > ActorHighLevel::internalWP [private]

internal member-variable to save the received waypoints

Definition at line 147 of file actorhighlevel.h.

7.1.4.7 double ActorHighLevel::lastDeltaA [private]

previous desired angle derivation

Definition at line 175 of file actorhighlevel.h.

7.1.4.8 double ActorHighLevel::lastDeltaL [private]

pervious desired length derivation

Definition at line 177 of file actorhighlevel.h.

7.1.4.9 QMutex* ActorHighLevel::mutexPidHist [private]

Mutex to prohibit racing condition

Definition at line 156 of file actorhighlevel.h.

7.1.4.10 QMutex ActorHighLevel::mutexState [static], [private]

Mutex to prohibit racing condition

Definition at line 137 of file actorhighlevel.h.

7.1.4.11 int ActorHighLevel::numWP [private]

length of the internal waypoints

Definition at line 148 of file actorhighlevel.h.

7.1.4.12 double ActorHighLevel::PID_A_D [private]

D-part of the angle-PID

Definition at line 170 of file actorhighlevel.h.

7.1.4.13 double ActorHighLevel::PID_A_I [private]

I-part of the angle-PID

Definition at line 169 of file actorhighlevel.h.

7.1.4.14 double ActorHighLevel::PID_A_P [private]

P-part of the angle-PID

Definition at line 168 of file actorhighlevel.h.

7.1.4.15 double ActorHighLevel::PID_V_D [private]

D-part of the velocity-PID

Definition at line 173 of file actorhighlevel.h.

7.1.4.16 double ActorHighLevel::PID_V_I [private]

I-part of the velocity-PID

Definition at line 172 of file actorhighlevel.h.

7.1.4.17 double ActorHighLevel::PID_V_P [private]

P-part of the velocity-PID

Definition at line 171 of file actorhighlevel.h.

7.1.4.18 QList<double> ActorHighLevel::pidHistDistIst [private]

Display the past n-secounds in GUI plot

Definition at line 163 of file actorhighlevel.h.

7.1.4.19 QList<double> ActorHighLevel::pidHistDistSoll [private]

Display the past n-secounds in GUI plot

Definition at line 164 of file actorhighlevel.h.

7.1.4.20 QList<double> ActorHighLevel::pidHistTime [private]

Display the past n-secounds in GUI plot

Definition at line 160 of file actorhighlevel.h.

7.1.4.21 QList<double> ActorHighLevel::pidHistWinkelIst [private]

Display the past n-secounds in GUI plot

Definition at line 162 of file actorhighlevel.h.

7.1.4.22 QList<double> ActorHighLevel::pidHistWinkelSoll [private]

Display the past n-secounds in GUI plot

Definition at line 161 of file actorhighlevel.h.

7.1.4.23 bool ActorHighLevel::positionWasReached [private]

Definition at line 153 of file actorhighlevel.h.

7.1.4.24 std::atomic_bool ActorHighLevel::quitting [private]

Definition at line 139 of file actorhighlevel.h.

7.1.4.25 Position ActorHighLevel::releasePuckOrigin [private]

current position of the robot at the time of the beginning of release puck including: x, y and orientation

Definition at line 145 of file actorhighlevel.h.

7.1.4.26 Position ActorHighLevel::robotPosition [private]

current position of the robot including: x, y and orientation

Definition at line 144 of file actorhighlevel.h.

7.1.4.27 tkqt::spline ActorHighLevel::splineX [private]

cubic hermite spline created of received waypoints in x direction

Definition at line 150 of file actorhighlevel.h.

7.1.4.28 tkqt::spline ActorHighLevel::splineY [private]

cubic hermite spline created of received waypoints in y direction

Definition at line 151 of file actorhighlevel.h.

7.1.4.29 StatePathProcessing ActorHighLevel::state = StatePathProcessing::RUNNING [static], [private]

internal state-machine

Definition at line 142 of file actorhighlevel.h.

7.1.4.30 std::atomic_bool ActorHighLevel::streamPIDEnabled [static]

Definition at line 49 of file actorhighlevel.h.

7.1.4.31 double ActorHighLevel::targetDistDiffLast [private]

Definition at line 180 of file actorhighlevel.h.

7.1.4.32 double ActorHighLevel::targetDistLast [private]

Definition at line 179 of file actorhighlevel.h.

7.1.4.33 qint64 ActorHighLevel::timeOfStart [private]

ms since epoch, used to display time since program start in pid plot

Definition at line 159 of file actorhighlevel.h.

7.1.4.34 QTimer* ActorHighLevel::timerPIDPlot [private]

update timer for the PID-plot

Definition at line 158 of file actorhighlevel.h.

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

- actorhighlevel.h
- actorhighlevel.cpp

7.2 ActorLowLevel Class Reference

The ActorLowLevel class Class for accessing the PlayerCc::Position2dProxy, as single point of access.

`#include <actorLowLevel.h>`

Collaboration diagram for ActorLowLevel:

Public Slots

- void setRobotRemoteControllParams (double velocity, double turnangle)

  setRobotRemoteControllParams will set the class internal private attributes for velocity and turnrate and is used by the remote controll of the GUI

- void setRobotControllParams (double velocity, double turnangle)

  setRobotControllParams will set the new velocity commands from controller.
- void slotEmergencyStopEnabled (bool isEmergency)

  slotEmergencyStopEnabled if an emergency occures, the velocity is set to zero.
- void setOdometry (Position currentPosition)

  setOdometry will set the odometry via the player proxy

Public Member Functions

- ActorLowLevel ()

  Default-Constructor.
- ∼ActorLowLevel ()

  The Desturctor will clean the heap objects -> e.g: PositionProxy2D.
- void moveRobot (double speed, double turn)

  oveRobot is the most important function for robots actors

Private Attributes

- PlayerCc::Position2dProxy ∗ positionProxy
- double previousTurnRate
- double previousVelocity
- double tempVelocity
- double tempTurnRate
- bool moveForward
- bool isRefereeEmergency

7.2.1 Detailed Description

The ActorLowLevel class Class for accessing the PlayerCc::Position2dProxy, as single point of access.

Definition at line 18 of file actorLowLevel.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ActorLowLevel::ActorLowLevel ( )

Default-Constructor.

Definition at line 15 of file actorLowLevel.cpp.

7.2.2.2 ActorLowLevel::∼ActorLowLevel ( )

The Desturctor will clean the heap objects -> e.g: PositionProxy2D.

Definition at line 30 of file actorLowLevel.cpp.

7.2.3 Member Function Documentation

7.2.3.1 void ActorLowLevel::moveRobot (double *speed,* double *turn*)

oveRobot is the most important function for robots actors

Parameters

in	*speed*	(double) in m/s
in	*turn*	(double) in rad/s

Definition at line 39 of file actorLowLevel.cpp.

7.2.3.2 void ActorLowLevel::setOdometry (Position *currentPosition*) [slot]

setOdometry will set the odometry via the player proxy

Parameters

in	*Position*	struct with x(double) position in x direction in m, y (double) position in y direction in m, orientation (double) in radian

Definition at line 141 of file actorLowLevel.cpp.

7.2.3.3 void ActorLowLevel::setRobotControllParams (double *velocity,* double *turnangle*) [slot]

setRobotControllParams will set the new velocity commands from controller.

Parameters

in	*velocity*	(double) in m/s
in	*turnangle*	(double) in rad/s

Definition at line 94 of file actorLowLevel.cpp.

7.2.3.4 void ActorLowLevel::setRobotRemoteControllParams (double *velocity,* double *turnangle*) [slot]

setRobotRemoteControllParams will set the class internal private attributes for velocity and turnrate and is used by the remote controll of the GUI

Parameters

in	*vel*	(double): Velocity of robot in m/s
in	*deg*	(double): Angle in degree to turn the robot

Definition at line 51 of file actorLowLevel.cpp.

7.2.3.5 void ActorLowLevel::slotEmergencyStopEnabled (bool *isEmergency*) [slot]

slotEmergencyStopEnabled if an emergency occures, the velocity is set to zero.

Parameters

in	*isEmergency*	(bool) send by sensors.

Definition at line 104 of file actorLowLevel.cpp.

7.2.4 Member Data Documentation

7.2.4.1 bool ActorLowLevel::isRefereeEmergency [private]

true if the game engine killed the path planning

Definition at line 69 of file actorLowLevel.h.

7.2.4.2 bool ActorLowLevel::moveForward [private]

Check if driving forward is allowed

Definition at line 68 of file actorLowLevel.h.

7.2.4.3 PlayerCc::Position2dProxy* ActorLowLevel::positionProxy [private]

positionProxy of the player stage => initialised on heap

Definition at line 62 of file actorLowLevel.h.

7.2.4.4 double ActorLowLevel::previousTurnRate [private]

internal param to save the previous turn rate of remote control with collision avoidance

Definition at line 63 of file actorLowLevel.h.

7.2.4.5 double ActorLowLevel::previousVelocity [private]

internal param to sage the previous velocity of remote control with collision avoidance

Definition at line 64 of file actorLowLevel.h.

7.2.4.6 double ActorLowLevel::tempTurnRate [private]

used for internal triangle swap

Definition at line 66 of file actorLowLevel.h.

7.2.4.7 double ActorLowLevel::tempVelocity [private]

used for internal triangle swap

Definition at line 65 of file actorLowLevel.h.

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

- actorLowLevel.h
- actorLowLevel.cpp

7.3 tkqt::band_matrix Class Reference

The band_matrix class, which is the basis for cubic hermite spline creation.

`#include <spline.h>`

Collaboration diagram for tkqt::band_matrix:

Public Member Functions

- band_matrix ()

  band_matrix => standard constructor
- band_matrix (int dim, int n_u, int n_l)

  band_matrix => overwritten constructor with dimension
- ~band_matrix ()

  Destructor.
- void resize (int dim, int n_u, int n_l)

  resize This method will intialise the band matrix
- int dim () const

  dim This method will return the dimension of the matrix
- int num_upper () const

  num_upper This method will return the size of the internal matrix m_upper
- int num_lower () const

  num_lower This method will return the size of the internal matrix m_lower
- double & operator() (int i, int j)
- double operator() (int i, int j) const
- double & saved_diag (int i)
- double saved_diag (int i) const
- void lu_decompose ()

  lu_decompose : This method will compute a LU-Decomposition
- QVector< double > r_solve (const QVector< double > &b) const

  r_solve : This method solves Rx=y
- QVector< double > l_solve (const QVector< double > &b) const

  l_solve : This method solves Ly=b
- QVector< double > lu_solve (const QVector< double > &b, bool is_lu_decomposed=false)

  lu_solve: This method will solve A = LU with LU-decomposite

Private Attributes

- QVector< QVector< double > > m_upper
- QVector< QVector< double > > m_lower

7.3.1 Detailed Description

The band_matrix class, which is the basis for cubic hermite spline creation.

Definition at line 37 of file spline.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 tkqt::band_matrix::band_matrix ( ) [inline]

band_matrix => standard constructor

Definition at line 47 of file spline.h.

7.3.2.2 tkqt::band_matrix::band_matrix (int *dim*, int *n_u*, int *n_l*)

band_matrix => overwritten constructor with dimension

Parameters

in	*dim*	(int) dimension
in	*n_u*	(int)
in	*n_l*	(int)

Definition at line 14 of file spline.cpp.

7.3.2.3 tkqt::band_matrix::~band_matrix ( ) [inline]

Destructor.

Definition at line 58 of file spline.h.

7.3.3 Member Function Documentation

7.3.3.1 int tkqt::band_matrix::dim ( ) const

dim This method will return the dimension of the matrix

Returns

The dimension of the matrix (int)

Definition at line 35 of file spline.cpp.

7.3.3.2 QVector< double > tkqt::band_matrix::l_solve (const QVector< double > & *b*) const

l_solve : This method solves Ly=b

Parameters

b	:is the result vector (QVector<double>) Reference

Returns

return QVector<double> y

Definition at line 122 of file spline.cpp.

7.3.3.3 void tkqt::band_matrix::lu_decompose ( )

lu_decompose : This method will compute a LU-Decomposition

Definition at line 84 of file spline.cpp.

7.3.3.4 QVector< double > tkqt::band_matrix::lu_solve (const QVector< double > & *b*, bool *is_lu_decomposed* = false)

lu_solve: This method will solve A = LU with LU-decomposite

Parameters

b	
is_lu_decomposed	

Returns

an QVector<double>

Definition at line 154 of file spline.cpp.

7.3.3.5 int tkqt::band_matrix::num_lower ( ) const [inline]

num_lower This method will return the size of the internal matrix m_lower

Returns

the size of the m_lower matrix

Definition at line 84 of file spline.h.

7.3.3.6 int tkqt::band_matrix::num_upper ( ) const [inline]

num_upper This method will return the size of the internal matrix m_upper

Returns

the size of the m_upper matrix

Definition at line 77 of file spline.h.

7.3.3.7 double & tkqt::band_matrix::operator() (int *i*, int *j*)

Definition at line 48 of file spline.cpp.

7.3.3.8 double tkqt::band_matrix::operator() (int *i*, int *j*) const

Definition at line 58 of file spline.cpp.

7.3.3.9 QVector< double > tkqt::band_matrix::r_solve (const QVector< double > & *b*) const

r_solve : This method solves Rx=y

Parameters

b	Reference of QVector<double>

Returns

the QVector<double> x

Definition at line 139 of file spline.cpp.

7.3.3.10 void tkqt::band_matrix::resize (int *dim*, int *n_u*, int *n_l*)

resize This method will intialise the band matrix

Parameters

in	*dim*	(int)
in	*n_u*	(int)
in	*n_l*	(int)

Definition at line 19 of file spline.cpp.

7.3.3.11 double & tkqt::band_matrix::saved_diag (int *i*)

Definition at line 76 of file spline.cpp.

7.3.3.12 double tkqt::band_matrix::saved_diag (int *i*) const

Definition at line 70 of file spline.cpp.

7.3.4 Member Data Documentation

7.3.4.1 QVector< QVector<double> > tkqt::band_matrix::m_lower [private]

lower band QVector of QVector

Definition at line 41 of file spline.h.

7.3.4.2 QVector< QVector<double> > tkqt::band_matrix::m_upper [private]

upper band QVector of QVector

Definition at line 40 of file spline.h.

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

- spline.h
- spline.cpp

7.4 Cam Class Reference

The Cam class will stream cam data to GUI-panal for recognition of poles color.

`#include <cam.h>`

Collaboration diagram for Cam:

Public Slots

- void slotSetCameraParams (CameraParams cameraParams)

  slotSetCameraParams
- void timerSendFrame ()

  timerGrabFrame will determine which color is in front of the cam (called with an timer heartbeat)
- void slotStartColorDetection ()

  slotGetColor

Signals

- void signalDisplayFrame (cv::Mat)

  signalDisplayFrame
- void signalColorDetected (CamColor)

  signalColorDetected

Public Member Functions

- Cam ()

  Cam default constructor for setting up the video stream.
- ∼Cam ()

  ∼Cam default destructor
- CamColor getLastColor ()

  getLastColor

Static Public Attributes

- static std::atomic_bool streamCamEnabled

Private Member Functions

- cv::Mat grabFrameAndColor ()

  grabFrameAndColor
- CamColor getPixelColor (const QColor &pixel) const

  getPixelColor

Private Attributes

- CameraParams cp
- bool enabled
- cv::VideoCapture videoCapture
- QTimer ∗ timer
- CamColor color
- QMutex ∗ mutexVideoCapture

7.4.1 Detailed Description

The Cam class will stream cam data to GUI-panal for recognition of poles color.

Definition at line 33 of file cam.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 Cam::Cam ( )

Cam default constructor for setting up the video stream.

Definition at line 15 of file cam.cpp.

7.4.2.2 Cam::∼Cam ( )

∼Cam default destructor

Definition at line 42 of file cam.cpp.

7.4.3 Member Function Documentation

7.4.3.1 CamColor Cam::getLastColor ( )

getLastColor

Returns

the last recognised color

Definition at line 49 of file cam.cpp.

7.4.3.2 CamColor Cam::getPixelColor (const QColor & *pixel*) const [private]

getPixelColor

Parameters

in	*pixel*	(QColor)

Returns

Definition at line 267 of file cam.cpp.

7.4.3.3 cv::Mat Cam::grabFrameAndColor ( ) [private]

grabFrameAndColor

Definition at line 137 of file cam.cpp.

7.4.3.4 void Cam::signalColorDetected (CamColor) [signal]

signalColorDetected

7.4.3.5 void Cam::signalDisplayFrame (cv::Mat) [signal]

signalDisplayFrame

7.4.3.6 void Cam::slotSetCameraParams (CameraParams *cameraParams*) [slot]

slotSetCameraParams

Parameters

in	*cameraParams*	which are the new cam parameter

Definition at line 54 of file cam.cpp.

7.4.3.7 void Cam::slotStartColorDetection ( ) [slot]

slotGetColor

Returns

the last gathered color.

Definition at line 95 of file cam.cpp.

7.4.3.8 void Cam::timerSendFrame ( ) [slot]

timerGrabFrame will determine which color is in front of the cam (called with an timer heartbeat)

Definition at line 89 of file cam.cpp.

7.4.4 Member Data Documentation

7.4.4.1 CamColor Cam::color [private]

internal color Enum

Definition at line 89 of file cam.h.

7.4.4.2 CameraParams Cam::cp [private]

parameters of the cam

Definition at line 85 of file cam.h.

7.4.4.3 bool Cam::enabled [private]

should the cam stream be enabled

Definition at line 86 of file cam.h.

7.4.4.4 QMutex* Cam::mutexVideoCapture [private]

mutex for the video capture

Definition at line 90 of file cam.h.

7.4.4.5 std::atomic_bool Cam::streamCamEnabled [static]

Definition at line 47 of file cam.h.

7.4.4.6 QTimer* Cam::timer [private]

captured video timer to gather a new frame

Definition at line 88 of file cam.h.

7.4.4.7 cv::VideoCapture Cam::videoCapture [private]

Definition at line 87 of file cam.h.

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

- cam.h
- cam.cpp

7.5 CameraParams Struct Reference

The CameraParams struct represents the cam params for calibration.

`#include <cameraparams.h>`

Collaboration diagram for CameraParams:

Public Attributes

- int source
- int updatePeriod
- int width
- int height
- QColor colorGreenMin
- QColor colorGreenMax
- QColor colorBlueMin
- QColor colorBlueMax
- QColor colorYellowMin
- QColor colorYellowMax

7.5.1 Detailed Description

The CameraParams struct represents the cam params for calibration.

Definition at line 9 of file Sensor/cameraparams.h.

7.5.2 Member Data Documentation

7.5.2.1 QColor CameraParams::colorBlueMax

maximal value for color blue

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.2 QColor CameraParams::colorBlueMin

minimal value for color blue

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.3 QColor CameraParams::colorGreenMax

maximal value for color green

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.4 QColor CameraParams::colorGreenMin

minimal value for color green

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.5 QColor CameraParams::colorYellowMax

maximal value for color yelow

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.6 QColor CameraParams::colorYellowMin

minimal value for color yellow

Definition at line 16 of file Sensor/cameraparams.h.

7.5.2.7 int CameraParams::height

stream height

Definition at line 11 of file Sensor/cameraparams.h.

7.5.2.8 int CameraParams::source

stream source

Definition at line 11 of file Sensor/cameraparams.h.

7.5.2.9 int CameraParams::updatePeriod

stream update rate

Definition at line 11 of file Sensor/cameraparams.h.

7.5.2.10 int CameraParams::width

stream width

Definition at line 11 of file Sensor/cameraparams.h.

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

- Sensor/cameraparams.h
- Structs/cameraparams.h

7.6 ConstrainedLaserData Class Reference

The ConstrainedLaserData class is a Datapacket containing the processed data from the LowLevelSensor and is used for sharing references of the data through the SensorHighLevel.

`#include <constrainedlaserdata.h>`

Collaboration diagram for ConstrainedLaserData:

Public Member Functions

- ConstrainedLaserData ()

  ConstrainedLaserData.
- ~ConstrainedLaserData ()
- void clearData ()

  clearData does delete all the items collected in the internal lists
- QList< double > filteredDepths () const

  filteredDepths
- void addFilteredDepth (const double &value)

  addFilteredDepth to the internal QList
- QList< double > rawDepths () const

  rawDepths
- void addRawDepth (const double &value)

  addRawDepth
- QList< double > angles () const

  angles
- void addAngle (const double &value)

  addAngle

Private Attributes

- QList< double > m_filteredDepth
- QList< double > m_rawDepths
- QList< double > m_angles

7.6.1 Detailed Description

The ConstrainedLaserData class is a Datapacket containing the processed data from the LowLevelSensor and is used for sharing references of the data through the SensorHighLevel.

Definition at line 10 of file constrainedlaserdata.h.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 ConstrainedLaserData::ConstrainedLaserData ( )

ConstrainedLaserData.

Definition at line 4 of file constrainedlaserdata.cpp.

7.6.2.2 ConstrainedLaserData::~ConstrainedLaserData ( )

Definition at line 11 of file constrainedlaserdata.cpp.

7.6.3 Member Function Documentation

7.6.3.1 void ConstrainedLaserData::addAngle (const double & *value*)

addAngle

Parameters

in	*value*	(double)

Definition at line 48 of file constrainedlaserdata.cpp.

7.6.3.2 void ConstrainedLaserData::addFilteredDepth (const double & *value*)

addFilteredDepth to the internal QList

Parameters

in	*value*	(double)

Definition at line 28 of file constrainedlaserdata.cpp.

7.6.3.3 void ConstrainedLaserData::addRawDepth (const double & *value*)

addRawDepth

Parameters

in	*value*	(double)

Definition at line 38 of file constrainedlaserdata.cpp.

7.6.3.4 QList< double > ConstrainedLaserData::angles ( ) const

angles

Returns

QList<double> of corresponding angles to the constrained laser data

Definition at line 43 of file constrainedlaserdata.cpp.

7.6.3.5 void ConstrainedLaserData::clearData ( )

clearData does delete all the items collected in the internal lists

Definition at line 16 of file constrainedlaserdata.cpp.

7.6.3.6 QList< double > ConstrainedLaserData::filteredDepths ( ) const

filteredDepths

Returns

QList<double> of filtered laser data by the median filter

Definition at line 23 of file constrainedlaserdata.cpp.

7.6.3.7 QList< double > ConstrainedLaserData::rawDepths ( ) const

rawDepths

Returns

QList<double> of raw laser data from the sensor

Definition at line 33 of file constrainedlaserdata.cpp.

7.6.4 Member Data Documentation

7.6.4.1 QList<double> ConstrainedLaserData::m_angles [private]

Containes the corresponding angles to both depth lists

Definition at line 64 of file constrainedlaserdata.h.

7.6.4.2 QList<double> ConstrainedLaserData::m_filteredDepth [private]

Containes the depths processed by the median filter

Definition at line 62 of file constrainedlaserdata.h.

7.6.4.3 QList<double> ConstrainedLaserData::m_rawDepths [private]

Containes the depths directly received by the LowLevelSensor

Definition at line 63 of file constrainedlaserdata.h.

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

- constrainedlaserdata.h
- constrainedlaserdata.cpp

7.7 CVImageWidget Class Reference

The CVImageWidget class will draw the video stream directy instead of bytewise.

`#include <cvimagewidget.h>`

Collaboration diagram for CVImageWidget:

Public Slots

- void showImage (const cv::Mat &image)
- void showImage (const cv::Mat &image)

Public Member Functions

- CVImageWidget (QWidget *parent=0)
- QSize sizeHint () const
- QSize minimumSizeHint () const
- CVImageWidget (QWidget *parent=0)
- QSize sizeHint () const
- QSize minimumSizeHint () const

Protected Member Functions

- void paintEvent (QPaintEvent *)
- void paintEvent (QPaintEvent *)

Protected Attributes

- QImage _qimage
- cv::Mat _tmp

7.7.1 Detailed Description

The CVImageWidget class will draw the video stream directy instead of bytewise.

See Also

source: http://develnoter.blogspot.de/2012/05/integrating-opencv-in-qt-gui.-html

Definition at line 15 of file GUI/cvimagewidget.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 CVImageWidget::CVImageWidget (QWidget * *parent* = 0) [inline], [explicit]

Definition at line 19 of file GUI/cvimagewidget.h.

7.7.2.2 CVImageWidget::CVImageWidget (QWidget * *parent* = 0) [inline], [explicit]

Definition at line 20 of file Plots/cvimagewidget.h.

7.7.3 Member Function Documentation

7.7.3.1 QSize CVImageWidget::minimumSizeHint ( ) const [inline]

Definition at line 22 of file GUI/cvimagewidget.h.

7.7.3.2 QSize CVImageWidget::minimumSizeHint ( ) const [inline]

Definition at line 23 of file Plots/cvimagewidget.h.

7.7.3.3 void CVImageWidget::paintEvent (QPaintEvent *) [inline], [protected]

Definition at line 50 of file GUI/cvimagewidget.h.

7.7.3.4 void CVImageWidget::paintEvent (QPaintEvent *) [inline], [protected]

Definition at line 51 of file Plots/cvimagewidget.h.

7.7.3.5 void CVImageWidget::showImage (const cv::Mat & *image*) [inline], [slot]

Definition at line 26 of file GUI/cvimagewidget.h.

7.7.3.6 void CVImageWidget::showImage (const cv::Mat & *image*) [inline], [slot]

Definition at line 27 of file Plots/cvimagewidget.h.

7.7.3.7 QSize CVImageWidget::sizeHint ( ) const [inline]

Definition at line 21 of file GUI/cvimagewidget.h.

7.7.3.8 QSize CVImageWidget::sizeHint ( ) const [inline]

Definition at line 22 of file Plots/cvimagewidget.h.

7.7.4 Member Data Documentation

7.7.4.1 QImage CVImageWidget::_qimage [protected]

Definition at line 57 of file GUI/cvimagewidget.h.

7.7.4.2 cv::Mat CVImageWidget::_tmp [protected]

Definition at line 58 of file GUI/cvimagewidget.h.

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

- GUI/cvimagewidget.h
- Plots/cvimagewidget.h

7.8 FilterParams Struct Reference

`#include <sensor.h>`

Collaboration diagram for FilterParams:

Public Member Functions

- FilterParams ()
- FilterParams (int kernel)

Public Attributes

- int ObsFilterAnzahl
- int ObsFilterSchnitt
- int PosFilterAnzahl
- int PosFilterSchnitt
- int kernel

7.8.1 Detailed Description

Definition at line 18 of file sensor.h.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 FilterParams::FilterParams ( ) [inline]

Definition at line 10 of file filterparams.h.

7.8.2.2 FilterParams::FilterParams (int *kernel*) [inline]

Definition at line 11 of file filterparams.h.

7.8.3 Member Data Documentation

7.8.3.1 int FilterParams::kernel

Definition at line 8 of file filterparams.h.

7.8.3.2 int FilterParams::ObsFilterAnzahl

Definition at line 20 of file sensor.h.

7.8.3.3 int FilterParams::ObsFilterSchnitt

Definition at line 20 of file sensor.h.

7.8.3.4 int FilterParams::PosFilterAnzahl

Definition at line 20 of file sensor.h.

7.8.3.5 int FilterParams::PosFilterSchnitt

Definition at line 20 of file sensor.h.

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

- sensor.h
- filterparams.h

7.9 Game Class Reference

`#include <game.h>`

Collaboration diagram for Game:

Public Types

- enum GameState {
  SELECT_PUCK, DRIVE_TO_PUCK, GATHERING_PUCK, SELECT_GOAL_SLOT,
  DRIVE_TO_GOAL, DRIVE_TO_DUMP, ANNOY_FOE, WAIT_FOR_RELEASE }

  The GameState enum, internal state machine which represents the game logic.
- enum SubGameStateAnnoy { DRIVE_RIGHT, DRIVE_LEFT }

The SubGameStateAnnoy enum : This enum represents sub-state machine in the annoy foe game state. For cyclic generation of motion targets in own goal area.

- enum HandleEnemyPuckState { FIND_SLOT, DRIVE_2_SLOT, RELEASE }

  The HandleEnemyPuckState enum, represents the sub-state machine for handling pucks with foes color.

Public Slots

- void slotStartGame ()

  slotStartGame, slot to listen to the refree signal, when the normal game starts.
- void slotColorDetect (CamColor color)

  slotColorDetect, slot and implementation of the gathering-puck state and is connected with the cam object for color recognition.
- void slotActorHighLevelIsDoneWithPuck ()

  slotActorHighLevelIsDoneWithPuck, connects to the actorHighLevel signal if the releasePuck, gatheringPuck or pushAndReleasePuck motion is done.
- void quit ()

  quit method to terminate the internal game-state loop
- void run ()

  run cyclic method which represents the game logic.
- void slotAnnoyFoe ()

  slotAnnoyFoe switch the game state to annoy foe

Signals

- void signalReportGoal ()

  signalReportGoal, signal for communication with refree object.
- void signalPuckRelease ()

  signalPuckRelease, signal to actorHighlevel to start the puck release, without pathplanning
- void signalStartColorDetectAI ()

  signalStartColorDetectAI, signal to cam-object to start the color-recognition for determination of the pucks color.
- void signalPushAndRelease (double releasePuckDistance)

  signalPushAndRelease, signal to actorHighlevel to move forwards and backwards a defined distance to push the pucks of foes on the boarder between some poles.
- void signalGatherPuck ()

  signalGatherPuck, signal to actorHighlevel to start the puck gathering => move straight forward without pathplanning.

Public Member Functions

- Game (GameEngine *gameEngine)

  Game constructor child object of gameEngine.
- ~Game ()

  Game destructor.

Private Member Functions

- bool findAndSelectBestPuck ()

  updateBestPuckPosition, will determine the puck-list in MapData to get the newest puck-target
- void findBestPuck (QList< Obstacle > &pucksList)

  findBestPuck, will determine the best puck to score (min(dist(puck<->bot))).
- void createTargetInfrontPuck ()

createTargetFromPuck, will create a motion target from given puck-position

- void driveToGoal ()

  makeAGoal method to drive to foes goal and score with gathered puck

- void annoyFoeState ()

  annoyFoeState, representation of the annoy-foe state, will drive to home base and clear our goal.

- Position getParkPosition (const Position &robPos)

  getParkPosition will determine the puck-position in foes goal

- void handleEnemyPuck ()

  handleEnemyPuck, implementation of DRIVE_TO_DUMP-state.

- Position findDumpSlot (Position ownPosition)

  findDumpSlot, will evaluate which slot is the best for dumping gathered foe's puck.

- bool checkTargetPuckAvailable ()

  checkTargetPuckAvailable, will check if the selected puck is still available.

- bool isPuckAwayFromPole (Obstacle puck)
- bool isPuckAwayFromEnemy (Obstacle puck)
- void getPositionToTurnRoboToMiddlePoint (Position &robot)
- Position getTargetForPuckBeforePole ()
- Position getTargetForPuck ()
- void findBestGoalSlot ()

Private Attributes

- std::atomic_bool quitting
- std::atomic_bool isGameStarted
- GameEngine * gameEngine
- GameState state
- SubGameStateAnnoy subAnnoyState
- HandleEnemyPuckState handleEnemyPuckState
- Obstacle currentPuck
- Obstacle lastTargetedPuck
- int currentPuckPrio
- int colorfail
- QElapsedTimer * timerLostPuck
- QElapsedTimer * timerUpdateGoalSlot
- QList< QPair< Position, double > > goalSlotCounterList

7.9.1 Detailed Description

Definition at line 19 of file game.h.

7.9.2 Member Enumeration Documentation

7.9.2.1 enum Game::GameState

The GameState enum, internal state machine which represents the game logic.

Enumerator:

SELECT_PUCK

DRIVE_TO_PUCK

GATHERING_PUCK

SELECT_GOAL_SLOT

DRIVE_TO_GOAL

DRIVE_TO_DUMP

ANNOY_FOE

WAIT_FOR_RELEASE

Definition at line 37 of file game.h.

7.9.2.2 enum Game::HandleEnemyPuckState

The HandleEnemyPuckState enum, represents the sub-state machine for handling pucks with foes color.

Enumerator:

FIND_SLOT

DRIVE_2_SLOT

RELEASE

Definition at line 60 of file game.h.

7.9.2.3 enum Game::SubGameStateAnnoy

The SubGameStateAnnoy enum : This enum represents sub-state machine in the annoy foe game state. For cyclic generation of motion targets in own goal area.

Enumerator:

DRIVE_RIGHT

DRIVE_LEFT

Definition at line 52 of file game.h.

7.9.3 Constructor & Destructor Documentation

7.9.3.1 Game::Game (GameEngine * *gameEngine*)

Game constructor child object of gameEngine.

Parameters

gameEngine	

Definition at line 15 of file game.cpp.

7.9.3.2 Game::~Game ( )

Game destructor.

Definition at line 35 of file game.cpp.

7.9.4 Member Function Documentation

7.9.4.1 void Game::annoyFoeState ( ) [private]

annoyFoeState, representation of the annoy-foe state, will drive to home base and clear our goal.

Definition at line 643 of file game.cpp.

7.9.4.2 bool Game::checkTargetPuckAvailable ( ) [private]

checkTargetPuckAvailable, will check if the selected puck is still available.

Returns

bool, if the selected puck is available

Definition at line 797 of file game.cpp.

7.9.4.3 void Game::createTargetInfrontPuck ( ) [private]

createTargetFromPuck, will create a motion target from given puck-position

Game::createTargetFromPuck: Create motion target for given puck. (Direction: goal of foe)

Parameters

in	*curPuck*	(Obstacle): reference of the best puck
	curPuck	

Definition at line 483 of file game.cpp.

7.9.4.4 void Game::driveToGoal ( ) [private]

makeAGoal method to drive to foes goal and score with gathered puck

Game::driveToGoal drive to foes goal and score.

Returns

True if a goal was scored

Definition at line 586 of file game.cpp.

7.9.4.5 bool Game::findAndSelectBestPuck ( ) [private]

updateBestPuckPosition, will determine the puck-list in MapData to get the newest puck-target

Todo eigentlich würde ich hier lieber etwas besser differenzieren

Definition at line 368 of file game.cpp.

7.9.4.6 void Game::findBestGoalSlot ( ) [private]

Definition at line 201 of file game.cpp.

7.9.4.7 void Game::findBestPuck (QList< Obstacle > & *pucksList*) [private]

findBestPuck, will determine the best puck to score (min(dist(puck<->bot))).

Parameters

in	*pucksList*	(List of obstacle):reference of a list of obstacle pucks to find the puck with shortest distance to the robot

7.9.4.8 Position Game::findDumpSlot (Position *ownPosition*) [private]

findDumpSlot, will evaluate which slot is the best for dumping gathered foe's puck.

Parameters

in	*ownPosition*	(Position) of our robot.

Returns

the postion for current dump.

Definition at line 751 of file game.cpp.

7.9.4.9 Position Game::getParkPosition (const Position & *robPos*) [private]

getParkPosition will determine the puck-position in foes goal

Parameters

in	*robPos*	(Position)-> current position of the rob

Returns

a position for current puck to create a target in foe's goal

Definition at line 789 of file game.cpp.

7.9.4.10 void Game::getPositionToTurnRoboToMiddlePoint (Position & *robot*) [private]

Definition at line 883 of file game.cpp.

7.9.4.11 Position Game::getTargetForPuck ( ) [private]

Definition at line 959 of file game.cpp.

7.9.4.12 Position Game::getTargetForPuckBeforePole ( ) [private]

Definition at line 904 of file game.cpp.

7.9.4.13 void Game::handleEnemyPuck ( ) [private]

handleEnemyPuck, implementation of DRIVE_TO_DUMP-state.

Definition at line 700 of file game.cpp.

7.9.4.14 bool Game::isPuckAwayFromEnemy (Obstacle *puck*) [private]

Definition at line 469 of file game.cpp.

7.9.4.15 bool Game::isPuckAwayFromPole (Obstacle *puck*) [private]

Definition at line 459 of file game.cpp.

7.9.4.16 void Game::quit ( ) [slot]

quit method to terminate the internal game-state loop

Definition at line 41 of file game.cpp.

7.9.4.17 void Game::run ( ) [slot]

run cyclic method which represents the game logic.

Definition at line 56 of file game.cpp.

7.9.4.18 void Game::signalGatherPuck ( ) [signal]

signalGatherPuck, signal to actorHighlevel to start the puck gathering => move straight forward without pathplanning.

7.9.4.19 void Game::signalPuckRelease ( ) [signal]

signalPuckRelease, signal to actorHighlevel to start the puck release, without pathplanning

7.9.4.20 void Game::signalPushAndRelease (double *releasePuckDistance*) [signal]

signalPushAndRelease, signal to actorHighlevel to move forwards and backwards a defined distance to push the pucks of foes on the boarder between some poles.

7.9.4.21 void Game::signalReportGoal ( ) [signal]

signalReportGoal, signal for communication with refree object.

7.9.4.22 void Game::signalStartColorDetectAI ( ) [signal]

signalStartColorDetectAI, signal to cam-object to start the color-recognition for determination of the pucks color.

7.9.4.23 void Game::slotActorHighLevelIsDoneWithPuck ( ) [slot]

slotActorHighLevelIsDoneWithPuck, connects to the actorHighLevel signal if the releasePuck, gatheringPuck or pushAndReleasePuck motion is done.

Definition at line 540 of file game.cpp.

7.9.4.24 void Game::slotAnnoyFoe ( ) [slot]

slotAnnoyFoe switch the game state to annoy foe

Definition at line 1003 of file game.cpp.

7.9.4.25 void Game::slotColorDetect (CamColor *color*) [slot]

slotColorDetect, slot and implementation of the gathering-puck state and is connected with the cam object for color recognition.

Parameters

in	*color*	

Definition at line 810 of file game.cpp.

7.9.4.26 void Game::slotStartGame ( ) [slot]

slotStartGame, slot to listen to the refree signal, when the normal game starts.

Definition at line 45 of file game.cpp.

7.9.5 Member Data Documentation

7.9.5.1 int Game::colorfail [private]

Definition at line 135 of file game.h.

7.9.5.2 Obstacle Game::currentPuck [private]

current target puck, which is determined as best option to score

Definition at line 132 of file game.h.

7.9.5.3 int Game::currentPuckPrio [private]

Definition at line 134 of file game.h.

7.9.5.4 GameEngine* Game::gameEngine [private]

pointer to the parent object

Definition at line 128 of file game.h.

7.9.5.5 QList<QPair<Position,double> > Game::goalSlotCounterList [private]

Definition at line 203 of file game.h.

7.9.5.6 HandleEnemyPuckState Game::handleEnemyPuckState [private]

current state of the internal sub-state-machine for handling foes pucks

Definition at line 131 of file game.h.

7.9.5.7 std::atomic_bool Game::isGameStarted [private]

bool if the game was started by refree signal

Definition at line 127 of file game.h.

7.9.5.8 Obstacle Game::lastTargetedPuck [private]

Definition at line 133 of file game.h.

7.9.5.9 std::atomic_bool Game::quitting [private]

bool if the state-machine should be terminated

Definition at line 126 of file game.h.

7.9.5.10 GameState Game::state [private]

current state of the internal state-machine

Definition at line 129 of file game.h.

7.9.5.11 SubGameStateAnnoy Game::subAnnoyState [private]

current state of the internal sub-state-machine for annoying foes

Definition at line 130 of file game.h.

7.9.5.12 QElapsedTimer* Game::timerLostPuck [private]

Definition at line 200 of file game.h.

7.9.5.13 QElapsedTimer* Game::timerUpdateGoalSlot [private]

Definition at line 201 of file game.h.

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

- game.h
- game.cpp

7.10 GameEngine Class Reference

`#include <gameengine.h>`

Collaboration diagram for GameEngine:

Public Types

- enum StateNameEnum {
  INIT, WAIT_FOR_DETECTION, DETECTION, WAIT_FOR_GAME,
  GAME, STOP }
  The StateNameEnum enum represents the current state, emited by refree signals.

Public Slots

- void slotDetectionFinished (CamColor color)
  slotDetectionFinished will get the right color from angelina after 1:30 mins. => save it to our static map obj.
- void slotReportGoal ()
  slotReportGoal, if we scored a goal-> tell it angelina.

Signals

- void signalEmergencyStopEnabled (bool enableEmergencyStop)

  signalEmergencyStopEnabled, if this signal is emited, the actor low level will be stopped immediately.
- void signalStartGame ()

  signalStartGame. This signal will be emited to init the state-machine of the game-obj.
- void signalStartDetection (bool startGameEngine)

  signalStartDetection will be emited to start the orientation state in sensor-highlevel.
- void signalAnnoyFoe ()

  signalAnnoyFoe will be emited if 4:15 mins are finished to switch to annoy enemy.

Public Member Functions

- GameEngine ()

  GameEngine, constructor of the game engine. This class will communicate with the refree obj.
- ~GameEngine ()
- StateNameEnum getState () const

  getState
- void startGameEngine ()

  startGameEngine. This method will be invoked by robot-thread for starting the game engine.

Private Slots

- void slotTimerAlive ()

  slotTimerAlive, which is needed to send in defined time slots a keep alive signal to angelina to avoid disqualification.
- void slotTimerEgoPos ()

  slotTimerEgoPos, will tell angelina our determined position.
- void slotRefConnected ()

  slotRefConnected to refree signal
- void slotRefConnectFailed ()

  slotRefConnectFailed to refree signal
- void slotRefDisconnected ()

  slotRefDisconnected to refree signal
- void slotRefDetectionStart ()

  slotRefDetectionStart to refree signal
- void slotRefTrueColorOfTeam (TeamColor color)

  slotRefTrueColorOfTeam to refree signal
- void slotRefGameStart ()

  slotRefGameStart to refree signal
- void slotRefStopMovement ()

  slotRefStopMovement to refree signal
- void slotRefGameOver ()

  slotRefGameOver to refree signal
- void slotTimerAnnoy ()

  slotTimerAnnoy until the timer will set the game state to annoy foe

Private Attributes

- Referee * referee
- QTimer * timerAlive
- QTimer * timerEgoPos
- QTimer * timerTillAnnoyEnemy
- StateNameEnum state

7.10.1 Detailed Description

Definition at line 12 of file gameengine.h.

7.10.2 Member Enumeration Documentation

7.10.2.1 enum GameEngine::StateNameEnum

The StateNameEnum enum represents the current state, emited by refree signals.

Enumerator:

INIT

WAIT_FOR_DETECTION

DETECTION

WAIT_FOR_GAME

GAME

STOP

Definition at line 26 of file gameengine.h.

7.10.3 Constructor & Destructor Documentation

7.10.3.1 GameEngine::GameEngine ( )

GameEngine, constructor of the game engine. This class will communicate with the refree obj.

Definition at line 16 of file gameengine.cpp.

7.10.3.2 GameEngine::~GameEngine ( )

Definition at line 68 of file gameengine.cpp.

7.10.4 Member Function Documentation

7.10.4.1 StateNameEnum GameEngine::getState ( ) const [inline]

getState

Returns

s the current internal state

Definition at line 32 of file gameengine.h.

7.10.4.2 void GameEngine::signalAnnoyFoe ( ) [signal]

signalAnnoyFoe will be emited if 4:15 mins are finished to switch to annoy enemy.

7.10.4.3 void GameEngine::signalEmergencyStopEnabled (bool *enableEmergencyStop*) [signal]

signalEmergencyStopEnabled, if this signal is emited, the actor low level will be stopped immediately.

Parameters

in	*enable-EmergencyStop*	(bool)

7.10.4.4 void GameEngine::signalStartDetection (bool *startGameEngine*) [signal]

signalStartDetection will be emited to start the orientation state in sensor-highlevel.

Parameters

in	*start*	(bool) if the detection should be started.

7.10.4.5 void GameEngine::signalStartGame ( ) [signal]

signalStartGame. This signal will be emited to init the state-machine of the game-obj.

7.10.4.6 void GameEngine::slotDetectionFinished (CamColor *color*) [slot]

slotDetectionFinished will get the right color from angelina after 1:30 mins. => save it to our static map obj.

Parameters

in	*color*	(CamColor)

Todo : Was tun wenn wir keine Frabe erkennen?

Definition at line 95 of file gameengine.cpp.

7.10.4.7 void GameEngine::slotRefConnected ( ) [private], [slot]

slotRefConnected to refree signal

Definition at line 167 of file gameengine.cpp.

7.10.4.8 void GameEngine::slotRefConnectFailed ( ) [private], [slot]

slotRefConnectFailed to refree signal

Definition at line 179 of file gameengine.cpp.

7.10.4.9 void GameEngine::slotRefDetectionStart ( ) [private], [slot]

slotRefDetectionStart to refree signal

Definition at line 190 of file gameengine.cpp.

7.10.4.10 void GameEngine::slotRefDisconnected ( ) [private], [slot]

slotRefDisconnected to refree signal

Definition at line 184 of file gameengine.cpp.

7.10.4.11 void GameEngine::slotRefGameOver ( ) [private], [slot]

slotRefGameOver to refree signal

Definition at line 236 of file gameengine.cpp.

7.10.4.12 void GameEngine::slotRefGameStart ( ) [private], [slot]

slotRefGameStart to refree signal

Definition at line 214 of file gameengine.cpp.

7.10.4.13 void GameEngine::slotRefStopMovement ( ) [private], [slot]

slotRefStopMovement to refree signal

Definition at line 226 of file gameengine.cpp.

7.10.4.14 void GameEngine::slotRefTrueColorOfTeam (TeamColor *color*) [private], [slot]

slotRefTrueColorOfTeam to refree signal

Definition at line 201 of file gameengine.cpp.

7.10.4.15 void GameEngine::slotReportGoal ( ) [slot]

slotReportGoal, if we scored a goal-> tell it angelina.

Definition at line 132 of file gameengine.cpp.

7.10.4.16 void GameEngine::slotTimerAlive ( ) [private], [slot]

slotTimerAlive, which is needed to send in defined time slots a keep alive signal to angelina to avoid disqualification.

Definition at line 139 of file gameengine.cpp.

7.10.4.17 void GameEngine::slotTimerAnnoy ( ) [private], [slot]

slotTimerAnnoy until the timer will set the game state to annoy foe

Definition at line 161 of file gameengine.cpp.

7.10.4.18 void GameEngine::slotTimerEgoPos ( ) [private], [slot]

slotTimerEgoPos, will tell angelina our determined position.

Definition at line 146 of file gameengine.cpp.

7.10.4.19 void GameEngine::startGameEngine ( )

startGameEngine. This method will be invoked by robot-thread for starting the game engine.

Definition at line 79 of file gameengine.cpp.

7.10.5 Member Data Documentation

7.10.5.1 Referee* GameEngine::referee [private]

refree-obj

Definition at line 133 of file gameengine.h.

7.10.5.2 StateNameEnum GameEngine::state [private]

current state of refree states

Definition at line 139 of file gameengine.h.

7.10.5.3 QTimer* GameEngine::timerAlive [private]

timer to tell angelina that our robot is still alive

Definition at line 135 of file gameengine.h.

7.10.5.4 QTimer * GameEngine::timerEgoPos [private]

timer to tell angelina were we are

Definition at line 135 of file gameengine.h.

7.10.5.5 QTimer * GameEngine::timerTillAnnoyEnemy [private]

timer to switch to annoy enemy

Definition at line 135 of file gameengine.h.

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

- gameengine.h
- gameengine.cpp

7.11 PathPlanning::GridPoint Class Reference

Collaboration diagram for PathPlanning::GridPoint:

Public Types

- enum PointType { GRID, TARGET, ROBOT }

Public Member Functions

- GridPoint ()

  GridPoint constructor of private member class representing a single grid point.
- GridPoint (PathPlanning *pathPlanning, int gridA, int gridB)

  GridPoint overloaded constructor of private member class.
- void init (GridPoint &point)

  init // does the initialization for a grid point, e.g. intrinsic cost calculation
- void calculateIntrinsicCost ()

calculateIntrinsicCost will calculate the intrinsic costs for this grid point

- double getDistance (const GridPoint *point)

  getDistance returns the distance to given Gridpoint

- QList< GridPoint * > getNeighbors (bool includeOutsideArena=false)

  getNeighbors will return the neighbour grid points for this grid point

- GridPoint * getGradientPoint ()

  getGradientPoint will return the neighbour with the smallest value

Public Attributes

- PathPlanning * pathPlanning
- double positionX
- double positionY
- int gridA
- int gridB
- double intrinsicCost
- double value
- enum PathPlanning::GridPoint::PointType type
- bool isOutsideArena
- bool active

Private Member Functions

- double constrainAngle (const double in)

  constrainAngle will return the adjusted angle (0-360° in rad)

7.11.1 Detailed Description

Definition at line 173 of file pathplanning.h.

7.11.2 Member Enumeration Documentation

7.11.2.1 enum PathPlanning::GridPoint::PointType

Enumerator:

GRID

TARGET

ROBOT

Definition at line 194 of file pathplanning.h.

7.11.3 Constructor & Destructor Documentation

7.11.3.1 PathPlanning::GridPoint::GridPoint ( )

GridPoint constructor of private member class representing a single grid point.

Definition at line 429 of file pathplanning.cpp.

7.11.3.2 PathPlanning::GridPoint::GridPoint (PathPlanning * *pathPlanning*, int *gridA*, int *gridB*)

GridPoint overloaded constructor of private member class.

Parameters

in	*pathPlanning*	(pointer on the current pathplanning obj)
in	*gridA*	(int)
in	*gridB*	(int)

Definition at line 432 of file pathplanning.cpp.

7.11.4 Member Function Documentation

7.11.4.1 void PathPlanning::GridPoint::calculateIntrinsicCost ( )

calculateIntrinsicCost will calculate the intrinsic costs for this grid point

Definition at line 462 of file pathplanning.cpp.

7.11.4.2 double PathPlanning::GridPoint::constrainAngle (const double *in*) [private]

constrainAngle will return the adjusted angle (0-360° in rad)

Parameters

in	(double) angle in rad

Returns

(double)

Definition at line 519 of file pathplanning.cpp.

7.11.4.3 double PathPlanning::GridPoint::getDistance (const GridPoint * *point*)

getDistance returns the distance to given Gridpoint

Parameters

in	*point*	(pointer to GridPoint)

Returns

the distance to given gridpoint

Definition at line 527 of file pathplanning.cpp.

7.11.4.4 PathPlanning::GridPoint * PathPlanning::GridPoint::getGradientPoint ( )

getGradientPoint will return the neighbour with the smallest value

Returns

pointer to neighbour with lowest value counter

Definition at line 595 of file pathplanning.cpp.

7.11.4.5 QList< PathPlanning::GridPoint * > PathPlanning::GridPoint::getNeighbors (bool *includeOutsideArena* = false)

getNeighbors will return the neighbour grid points for this grid point

Parameters

in	*includeOutside-Arena*	(bool) if area from outer arena should be included

Returns

a qlist of gridpoint pointers.

Definition at line 534 of file pathplanning.cpp.

7.11.4.6 void PathPlanning::GridPoint::init (GridPoint & *point*)

init // does the initialization for a grid point, e.g. intrinsic cost calculation

Parameters

point	

Definition at line 446 of file pathplanning.cpp.

7.11.5 Member Data Documentation

7.11.5.1 bool PathPlanning::GridPoint::active

bool if the pathplanning is active

Definition at line 196 of file pathplanning.h.

7.11.5.2 int PathPlanning::GridPoint::gridA

Definition at line 190 of file pathplanning.h.

7.11.5.3 int PathPlanning::GridPoint::gridB

Definition at line 191 of file pathplanning.h.

7.11.5.4 double PathPlanning::GridPoint::intrinsicCost

intrinisc costs for this grid point

Definition at line 192 of file pathplanning.h.

7.11.5.5 bool PathPlanning::GridPoint::isOutsideArena

bool if the gridpoint is outside the arena

Definition at line 195 of file pathplanning.h.

7.11.5.6 PathPlanning* PathPlanning::GridPoint::pathPlanning

member variable of parent obj adress

Definition at line 187 of file pathplanning.h.

7.11.5.7 double PathPlanning::GridPoint::positionX

position representation in x (carthesian space)

Definition at line 188 of file pathplanning.h.

7.11.5.8 double PathPlanning::GridPoint::positionY

position representation in y (carthesian space)

Definition at line 189 of file pathplanning.h.

7.11.5.9 enum PathPlanning::GridPoint::PointType PathPlanning::GridPoint::type

represent which type this gridpoint is

7.11.5.10 double PathPlanning::GridPoint::value

overall costs

Definition at line 193 of file pathplanning.h.

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

- pathplanning.h
- pathplanning.cpp

7.12 Hermes Class Reference

`#include <hermes.h>`

Collaboration diagram for Hermes:

Public Member Functions

- Hermes (QObject *parent=0)
- ~Hermes ()

Private Attributes

- Referee * referee
- int myTeamID
- int messageSize

7.12.1 Detailed Description

Definition at line 30 of file hermes.h.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 Hermes::Hermes (QObject * *parent* = 0)

Definition at line 24 of file hermes.cpp.

7.12.2.2 Hermes::~Hermes ( )

Definition at line 29 of file hermes.cpp.

7.12.3 Member Data Documentation

7.12.3.1 int Hermes::messageSize [private]

Definition at line 42 of file hermes.h.

7.12.3.2 int Hermes::myTeamID [private]

Definition at line 41 of file hermes.h.

7.12.3.3 Referee* Hermes::referee [private]

Definition at line 40 of file hermes.h.

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

- hermes.h
- hermes.cpp

7.13 LaserPlotData Struct Reference

The LaserPlotData struct is the Datapacket for plotting the laser data.

`#include <laserplotdata.h>`

Collaboration diagram for LaserPlotData:

Public Types

- enum DataTypeEnum {
  RAW, REDUCED, AVERAGE, MEDIAN,
  OBJECTS }

Public Member Functions

- LaserPlotData ()

  LaserPlotData.
- LaserPlotData (QVector< double > data, DataTypeEnum dataType)

  LaserPlotData will initialise the struct with data and dataType.
- LaserPlotData (QVector< double > keys, QVector< double > values, DataTypeEnum dataType)

  LaserPlotData will initialise the struct with keys, values and dataType.

Public Attributes

- QVector< double > angles
- QVector< double > sizes
- QVector< double > data
- DataTypeEnum dataType

7.13.1 Detailed Description

The LaserPlotData struct is the Datapacket for plotting the laser data.

Definition at line 10 of file laserplotdata.h.

7.13.2 Member Enumeration Documentation

7.13.2.1 enum LaserPlotData::DataTypeEnum

Enumerator:

RAW RAW datatype

REDUCED REDUCED datatype

AVERAGE AVERAGE datatype

MEDIAN MEDIAN datatype

OBJECTS OBJECTS datatype

Definition at line 12 of file laserplotdata.h.

7.13.3 Constructor & Destructor Documentation

7.13.3.1 LaserPlotData::LaserPlotData ( ) [inline]

LaserPlotData.

Definition at line 22 of file laserplotdata.h.

7.13.3.2 LaserPlotData::LaserPlotData (QVector< double > *data*, DataTypeEnum *dataType*) [inline]

LaserPlotData will initialise the struct with data and dataType.

Parameters

in	*data*	(QVector<double>)
in	*dataType*	(DataTypeEnum)

Definition at line 29 of file laserplotdata.h.

7.13.3.3 LaserPlotData::LaserPlotData (QVector< double > *keys*, QVector< double > *values*, DataTypeEnum *dataType*) [inline]

LaserPlotData will initialise the struct with keys, values and dataType.

Parameters

in	*keys*	(QVector<double>)

in	*values*	(QVector<double>)
in	*dataType*	(DataTypeEnum)

Definition at line 37 of file laserplotdata.h.

7.13.4 Member Data Documentation

7.13.4.1 QVector<double> LaserPlotData::angles

angles of data

Definition at line 40 of file laserplotdata.h.

7.13.4.2 QVector<double> LaserPlotData::data

Databody

Definition at line 44 of file laserplotdata.h.

7.13.4.3 DataTypeEnum LaserPlotData::dataType

type enumeration

Definition at line 45 of file laserplotdata.h.

7.13.4.4 QVector<double> LaserPlotData::sizes

sizes of data

Definition at line 41 of file laserplotdata.h.

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

- laserplotdata.h

7.14 Log Class Reference

The Log class.

`#include <log.h>`

Collaboration diagram for Log:

Static Public Member Functions

- static void setMainWindowReference (MainWindow *mainWindow)
- static void customLogger (QtMsgType type, const QMessageLogContext &context, const QString &msg)

  customLogger is a MessageHandler which redirect the qDebug(), qWarning(), qCritical() and qFatal() into our GUI.

Static Public Attributes

- static std::atomic_bool streamLogEnabled
- static LogParams logParams

Private Member Functions

- Log ()

Static Private Attributes

- static MainWindow * mainWindow = nullptr

7.14.1 Detailed Description

The Log class.

See Also

http://qt-project.org/doc/qt-5/qtglobal.html#qInstallMessageHandler

Definition at line 16 of file log.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Log::Log ( ) [private]

7.14.3 Member Function Documentation

7.14.3.1 void Log::customLogger (QtMsgType *type*, const QMessageLogContext & *context*, const QString & *msg*) [static]

customLogger is a MessageHandler which redirect the qDebug(), qWarning(), qCritical() and qFatal() into our GUI.

Parameters

in	*type*	(QtMsgType) one of QtDebugMsg, QtWarningMsg, QtCriticalMsg, QtFatalMsg
in	*context*	(QMessageLogContext) the messages origin
in	*msg*	(QString) the message.

Definition at line 28 of file log.cpp.

7.14.3.2 void Log::setMainWindowReference (MainWindow * *mainWindow*) [static]

Definition at line 17 of file log.cpp.

7.14.4 Member Data Documentation

7.14.4.1 LogParams Log::logParams [static]

Definition at line 20 of file log.h.

7.14.4.2 MainWindow * Log::mainWindow = nullptr [static], [private]

Definition at line 26 of file log.h.

7.14.4.3 std::atomic_bool Log::streamLogEnabled [static]

Definition at line 19 of file log.h.

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

- log.h
- log.cpp

7.15 LogParams Struct Reference

The LogParams struct describes the current logging level.

`#include <LogParams.h>`

Collaboration diagram for LogParams:

Public Types

- enum logLevelEnum {
  DEBUG = 0, WARNING = 1, CRITICAL = 2, FATAL = 3,
  DEBUG = 0, WARNING = 1, CRITICAL = 2, FATAL = 3 }
- enum logLevelEnum {
  DEBUG = 0, WARNING = 1, CRITICAL = 2, FATAL = 3,
  DEBUG = 0, WARNING = 1, CRITICAL = 2, FATAL = 3 }

  The logLevelEnum enum determining the current log level.

Public Attributes

- enum LogParams::logLevelEnum logLevel
- bool logActor
- bool logAI
- bool logData
- bool logPlots
- bool logSensor
- bool logOthers
- bool logMain

7.15.1 Detailed Description

The LogParams struct describes the current logging level.

Definition at line 7 of file LogParams.h.

7.15.2 Member Enumeration Documentation

7.15.2.1 enum LogParams::logLevelEnum

Enumerator:

DEBUG

WARNING

CRITICAL

FATAL

DEBUG

WARNING

CRITICAL

FATAL

Definition at line 9 of file LogParams.h.

7.15.2.2 enum LogParams::logLevelEnum

The logLevelEnum enum determining the current log level.

Enumerator:

DEBUG

WARNING

CRITICAL

FATAL

DEBUG

WARNING

CRITICAL

FATAL

Definition at line 12 of file logparams.h.

7.15.3 Member Data Documentation

7.15.3.1 bool LogParams::logActor

Definition at line 10 of file LogParams.h.

7.15.3.2 bool LogParams::logAI

Definition at line 10 of file LogParams.h.

7.15.3.3 bool LogParams::logData

Definition at line 10 of file LogParams.h.

7.15.3.4 enum LogParams::logLevelEnum LogParams::logLevel

7.15.3.5 bool LogParams::logMain

check if log level was activated

Definition at line 14 of file logparams.h.

7.15.3.6 bool LogParams::logOthers

Definition at line 10 of file LogParams.h.

7.15.3.7 bool LogParams::logPlots

Definition at line 10 of file LogParams.h.

7.15.3.8 bool LogParams::logSensor

Definition at line 10 of file LogParams.h.

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

- LogParams.h
- logparams.h

7.16 MainWindow Class Reference

The MainWindow class creates the GUI and connect user actions with programm functionalities for displaying and recording gathered data.

`#include <mainwindow.h>`

Collaboration diagram for MainWindow:

Public Slots

- void slotLog (QString html)

  slotLog will display logging messages
- void slotLaserDisplay (LaserPlotData laserData)

  slotLaserDisplay will display the gathered laserData
- void updatePathDisplay (PathPlotData dataPacket)

  updatePathDisplay will update the path information
- void slotDisplayFrame (cv::Mat mat)

  slotDisplayFrame will show given frames gathered by the cam
- void slotPIDPlot (PIDPlotData d)

  slotPIDPlot will plot gathered PID-informations
- void slotRestartTimerDisplay ()

  slotRestartTimerDisplay
- void slotUpdateColorLabel (CamColor color)

  slotUpdateColorLabel
- void slotLog (QString html)

  slotLog will display logging messages
- void slotLaserDisplay (LaserPlotData laserData)

  slotLaserDisplay will display the gathered laserData
- void updatePathDisplay (PathPlotData dataPacket)

  updatePathDisplay will update the path information
- void slotDisplayFrame (cv::Mat mat)

  slotDisplayFrame will show given frames gathered by the cam
- void slotPIDPlot (PIDPlotData d)

  slotPIDPlot will plot gathered PID-informations

Signals

- void robotRemoteControllUpdate (double velocity, double degreeturn)

  robotRemoteControllUpdate will send remote control parameter to lowLevelActor
- void signalStartOrientation (bool change)

  signalStartOrientation will emulate the signal from refree to start orientation
- void updateRemoteOdometry (Position)

  updateRemoteOdometry
- void signalChangeCamParams (CameraParams cp)

  signalChangeCamParams will send the changed cam parameters
- void signalChangeFilterParams (FilterParams cp)

  signalChangeFilterParams will emit the changed filter params
- void signalChangePIDParams (PIDParams p)

  signalChangePIDParams will emit the changed PID params
- void signalTestPuckRelease ()

  signalTestPuckRelease emit the signal for releasing the pucks
- void signalTestStartGame ()

  signalTestStartGame emit the signal to start the game (refree signal)
- void signalTestColorDetect ()

  signalTestColorDetect emit the signal for color detection
- void robotRemoteControllUpdate (double velocity, double degreeturn)

  robotRemoteControllUpdate will send remote control parameter to lowLevelActor
- void signalStartOrientation (bool change)
- void updateRemoteOdometry (Position)

  updateRemoteOdometry
- void signalChangeCamParams (CameraParams cp)

  signalChangeCamParams will send the changed cam parameters
- void signalChangeFilterParams (FilterParams cp)

  signalChangeFilterParams will emit the changed filter params
- void signalChangePIDParams (PIDParams p)

  signalChangePIDParams will emit the changed PID params

Public Member Functions

- MainWindow (QWidget ∗parent=0)
- ∼MainWindow ()
- void setup ()

  setup will transmit the GUI params to created RoboThread
- MainWindow (QWidget ∗parent=0)
- ∼MainWindow ()
- void setup ()

  setup will transmit the GUI params to created RoboThread

Private Slots

- void slotSimulationDetect ()

  slotSimulationDetect if the we noticed that we are in simulation
- void refresh ()

  refresh the GUI
- void forward ()

  forward button of remote control

- void back ()

  back button of remote control
- void left ()

  left button of remote control
- void right ()

  right button of remote control
- void strongleft ()

  strongleft button of remote control
- void strongright ()

  strongright button of remote control
- void stop ()

  stop button of remote control
- void mousePressEvent (QMouseEvent ∗event)

  mousePressEvent if user clicked
- void clearTargets ()

  clearTargets will reset the target list of waypoints
- void on_cbStream_stateChanged (int arg1)

  on_cbStream_stateChanged emit the changed arguments for cam
- void on_camSourceSpin_valueChanged (int arg1)

  on_camSourceSpin_valueChanged emit the changed arguments for cam
- void on_updateSpinner_valueChanged (int arg1)

  on_updateSpinner_valueChanged emit the changed arguments for cam
- void on_sizeX_textChanged (const QString &arg1)

  on_sizeX_textChanged emit the changed arguments for cam
- void on_sizeY_textChanged (const QString &arg1)

  on_sizeY_textChanged emit the changed arguments for cam
- void on_logCBOther_stateChanged (int arg1)

  on_logCBOther_stateChanged if Other combobox changed -> emit the changed logging values
- void on_logCBActor_stateChanged (int arg1)

  on_logCBActor_stateChanged if Actor combobox changed -> emit the changed logging values
- void on_logCBAI_stateChanged (int arg1)

  on_logCBAI_stateChanged if AI combobox changed -> emit the changed logging values
- void on_logCBData_stateChanged (int arg1)

  on_logCBData_stateChanged if Data combobox changed -> emit the changed logging values
- void on_logCBSensor_stateChanged (int arg1)

  on_logCBSensor_stateChanged if Sensor combobox changed -> emit the changed logging values
- void on_logCBPlot_stateChanged (int arg1)

  on_logCBPlot_stateChanged if Plot combobox changed -> emit the changed logging values
- void on_logLevel_currentIndexChanged (int index)

  on_logLevel_currentIndexChanged if logLevel changed -> emit the changed logging values
- void changeFilterParams ()

  sendFilterParams, emit the gathered filter data
- void on_spinPIDAP_valueChanged (double arg1)

  on_spinPIDAP_valueChanged if angle PID P value changed -> emit the changed PID values
- void on_spinPIDAI_valueChanged (double arg1)

  on_spinPIDAI_valueChanged if angle PID I value changed -> emit the changed PID values
- void on_spinPIDAD_valueChanged (double arg1)

  on_spinPIDAD_valueChanged if angle PID D value changed -> emit the changed PID values
- void on_spinPIDVP_valueChanged (double arg1)

  on_spinPIDVP_valueChanged if velocity PID P value changed -> emit the changed PID values
- void on_spinPIDVI_valueChanged (double arg1)

on_spinPIDVI_valueChanged if velocity PID I value changed -> emit the changed PID values

- void on_spinPIDVD_valueChanged (double arg1)

  on_spinPIDVD_valueChanged if velocity PID D value changed -> emit the changed PID values
- void on_streamSensor_stateChanged (int arg1)

  on_streamSensor_stateChanged check box if we want to see the sensor stream
- void on_streamPath_stateChanged (int arg1)

  on_streamPath_stateChanged check box if we want to see the path stream
- void on_streamPID_stateChanged (int arg1)

  on_streamPID_stateChanged check box if we want to see the PID-stream
- void on_streamLog_stateChanged (int arg1)

  on_streamLog_stateChanged check box if we want to see the Log-stream
- void on_btn_ReleasePuck_clicked ()

  on_btn_ReleasePuck_clicked if button released the puck
- void on_btn_StartGame_clicked ()

  on_btn_StartGame_clicked if botton to start the game is clicked
- void on_spinBox_kernel_valueChanged (int arg1)

  on_spinBox_kernel_valueChanged if kernel size of spin box has changed
- void on_refreshTime_valueChanged (int arg1)

  on_refreshTime_valueChanged if refresh time spin box changed.
- void on_pushButton_StartOrientation_clicked ()

  on_pushButton_StartOrientation_clicked if start orientation was clicked
- void on_btnDetectColor_clicked ()

  on_btnDetectColor_clicked if color detection started
- void refresh ()

  refresh the GUI
- void setrefreshrate ()

  setrefreshrate set the refresh cycle
- void forward ()

  forward button of remote control
- void back ()

  back button of remote control
- void left ()

  left button of remote control
- void right ()

  right button of remote control
- void strongleft ()

  strongleft button of remote control
- void strongright ()

  strongright button of remote control
- void stop ()

  stop button of remote control
- void mousePressEvent (QMouseEvent ∗event)

  mousePressEvent if user clicked
- void orientationSetup ()

  orientationSetup creates the orientation for pucks and poles
- void clearTargets ()

  clearTargets will reset the target list of waypoints
- void on_cbStream_stateChanged (int arg1)

  on_cbStream_stateChanged emit the changed arguments for cam
- void on_camSourceSpin_valueChanged (int arg1)

  on_camSourceSpin_valueChanged emit the changed arguments for cam

- void on_updateSpinner_valueChanged (int arg1)

  on_updateSpinner_valueChanged emit the changed arguments for cam
- void on_sizeX_textChanged (const QString &arg1)

  on_sizeX_textChanged emit the changed arguments for cam
- void on_sizeY_textChanged (const QString &arg1)

  on_sizeY_textChanged emit the changed arguments for cam
- void on_logCBOther_stateChanged (int arg1)

  on_logCBOther_stateChanged if Other combobox changed -> emit the changed logging values
- void on_logCBActor_stateChanged (int arg1)

  on_logCBActor_stateChanged if Actor combobox changed -> emit the changed logging values
- void on_logCBAI_stateChanged (int arg1)

  on_logCBAI_stateChanged if AI combobox changed -> emit the changed logging values
- void on_logCBData_stateChanged (int arg1)

  on_logCBData_stateChanged if Data combobox changed -> emit the changed logging values
- void on_logCBSensor_stateChanged (int arg1)

  on_logCBSensor_stateChanged if Sensor combobox changed -> emit the changed logging values
- void on_logCBPlot_stateChanged (int arg1)

  on_logCBPlot_stateChanged if Plot combobox changed -> emit the changed logging values
- void on_logLevel_currentIndexChanged (int index)

  on_logLevel_currentIndexChanged if logLevel changed -> emit the changed logging values
- void changeFilterParams ()

  sendFilterParams, emit the gathered filter data
- void on_spinPIDAP_valueChanged (double arg1)

  on_spinPIDAP_valueChanged if angle PID P value changed -> emit the changed PID values
- void on_spinPIDAI_valueChanged (double arg1)

  on_spinPIDAI_valueChanged if angle PID I value changed -> emit the changed PID values
- void on_spinPIDAD_valueChanged (double arg1)

  on_spinPIDAD_valueChanged if angle PID D value changed -> emit the changed PID values
- void on_spinPIDVP_valueChanged (double arg1)

  on_spinPIDVP_valueChanged if velocity PID P value changed -> emit the changed PID values
- void on_spinPIDVI_valueChanged (double arg1)

  on_spinPIDVI_valueChanged if velocity PID I value changed -> emit the changed PID values
- void on_spinPIDVD_valueChanged (double arg1)

  on_spinPIDVD_valueChanged if velocity PID D value changed -> emit the changed PID values
- void on_streamSensor_stateChanged (int arg1)
- void on_streamPath_stateChanged (int arg1)
- void on_streamPID_stateChanged (int arg1)
- void on_streamLog_stateChanged (int arg1)

Private Member Functions

- void drawMap ()

  DrawMap will draw the data from map to GUI.
- int convertX (float x)

  convertX the x values to screen resolution
- int convertY (float y)

  convertY the y values to screen resolution
- void changeCamParams ()

  sendCamParams send the cam params if button is pressed
- void changeLogParams ()

  changeLogParams chages the logging params if GUI values changed

- void changePIDParams ()

  changePIDParams will emit the new params for the PID controler
- void drawMap ()

  DrawMap will draw the data from map to GUI.
- int convertX (float x)

  convertX the x values to screen resolution
- int convertY (float y)

  convertY the y values to screen resolution
- void changeCamParams ()

  sendCamParams send the cam params if button is pressed
- void changeLogParams ()

  changeLogParams chages the logging params if GUI values changed
- void changePIDParams ()

  changePIDParams will emit the new params for the PID controler

Private Attributes

- Ui::MainWindow * ui
- QGraphicsScene * map
- QTimer * refreshtimer
- QElapsedTimer timer

  Für Anzeige vergangener zeit.
- QList< QPair< int, int > > TrackingMe
- QList< QPair< int, int > > TrackingYou
- QCPGraph * graphLaserRaw
- QCPGraph * graphLaserReduced
- QCPGraph * graphLaserMedian
- QCPGraph * graphLaserObjects
- QCPScatterStyle * scatterLaserObjects
- QCPColorMap * colorMap
- QCPScatterStyle * scatterRobotStyle
- QCPScatterStyle * scatterWPStyle
- QCPGraph * graphRobotScatter
- QCPGraph * graphWPScatter
- QCPCurve * graphSplineCurve
- QList< QColor > colors
- QCPGraph * graphPIDASoll
- QCPGraph * graphPIDAIst
- QCPGraph * graphPIDDSoll
- QCPGraph * graphPIDDIst
- bool m_changeOrientation
- QGraphicsScene * graphicsScene_1
- QGraphicsScene * graphicsScene_2
- QGraphicsScene * graphicsScene_3
- QGraphicsScene * graphicsScene_4
- QGraphicsScene * graphicsScene_5
- QGraphicsScene * graphicsScene_6

7.16.1 Detailed Description

The MainWindow class creates the GUI and connect user actions with programm functionalities for displaying and recording gathered data.

Definition at line 41 of file GUI/mainwindow.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

Definition at line 23 of file GUI/mainwindow.cpp.

7.16.2.2 MainWindow::~MainWindow ( )

Todo segmentation fault when trying to call delete on the pointer

Definition at line 194 of file GUI/mainwindow.cpp.

7.16.2.3 MainWindow::MainWindow (QWidget * *parent* = 0) [explicit]

7.16.2.4 MainWindow::~MainWindow ( )

7.16.3 Member Function Documentation

7.16.3.1 void MainWindow::back ( ) [private], [slot]

back button of remote control

7.16.3.2 void MainWindow::back ( ) [private], [slot]

back button of remote control

Definition at line 880 of file GUI/mainwindow.cpp.

7.16.3.3 void MainWindow::changeCamParams ( ) [private]

sendCamParams send the cam params if button is pressed

7.16.3.4 void MainWindow::changeCamParams ( ) [private]

sendCamParams send the cam params if button is pressed

Definition at line 1079 of file GUI/mainwindow.cpp.

7.16.3.5 void MainWindow::changeFilterParams ( ) [private], [slot]

sendFilterParams, emit the gathered filter data

7.16.3.6 void MainWindow::changeFilterParams ( ) [private], [slot]

sendFilterParams, emit the gathered filter data

Definition at line 1118 of file GUI/mainwindow.cpp.

7.16.3.7 void MainWindow::changeLogParams ( ) [private]

changeLogParams chages the logging params if GUI values changed

7.16.3.8 void MainWindow::changeLogParams ( ) [private]

changeLogParams chages the logging params if GUI values changed

Definition at line 1105 of file GUI/mainwindow.cpp.

7.16.3.9 void MainWindow::changePIDParams ( ) [private]

changePIDParams will emit the new params for the PID controler

7.16.3.10 void MainWindow::changePIDParams ( ) [private]

changePIDParams will emit the new params for the PID controler

Definition at line 1132 of file GUI/mainwindow.cpp.

7.16.3.11 void MainWindow::clearTargets ( ) [private], [slot]

clearTargets will reset the target list of waypoints

7.16.3.12 void MainWindow::clearTargets (void) [private], [slot]

clearTargets will reset the target list of waypoints

Definition at line 1068 of file GUI/mainwindow.cpp.

7.16.3.13 int MainWindow::convertX (float *x*) [private]

convertX the x values to screen resolution

Parameters

in	*x*	(double)

Returns

the converted x values

7.16.3.14 int MainWindow::convertX (float *x*) [private]

convertX the x values to screen resolution

Parameters

in	*x*	(double)

Returns

the converted x values

Definition at line 686 of file GUI/mainwindow.cpp.

7.16.3.15 int MainWindow::convertY (float *y*) [private]

convertY the y values to screen resolution

Parameters

in	*y*	(double)

Returns

the converted y values

7.16.3.16 int MainWindow::convertY (float *y*) [private]

convertY the y values to screen resolution

Parameters

in	*y*	(double)

Returns

the converted y values

Definition at line 695 of file GUI/mainwindow.cpp.

7.16.3.17 void MainWindow::drawMap ( ) [private]

DrawMap will draw the data from map to GUI.

7.16.3.18 void MainWindow::drawMap ( ) [private]

DrawMap will draw the data from map to GUI.

Definition at line 307 of file GUI/mainwindow.cpp.

7.16.3.19 void MainWindow::forward ( ) [private], [slot]

forward button of remote control

7.16.3.20 void MainWindow::forward ( ) [private], [slot]

forward button of remote control

Definition at line 872 of file GUI/mainwindow.cpp.

7.16.3.21 void MainWindow::left ( ) [private], [slot]

left button of remote control

7.16.3.22 void MainWindow::left ( ) [private], [slot]

left button of remote control

Definition at line 888 of file GUI/mainwindow.cpp.

7.16.3.23 void MainWindow::mousePressEvent (QMouseEvent * *event*) [private], [slot]

mousePressEvent if user clicked

Parameters

in	*event*	(QMouseEvent)

7.16.3.24 void MainWindow::mousePressEvent (QMouseEvent * *event*) [private], [slot]

mousePressEvent if user clicked

Parameters

in	*event*	(QMouseEvent)

Definition at line 929 of file GUI/mainwindow.cpp.

7.16.3.25 void MainWindow::on_btn_ReleasePuck_clicked ( ) [private], [slot]

on_btn_ReleasePuck_clicked if button released the puck

Definition at line 1191 of file GUI/mainwindow.cpp.

7.16.3.26 void MainWindow::on_btn_StartGame_clicked ( ) [private], [slot]

on_btn_StartGame_clicked if botton to start the game is clicked

Definition at line 1196 of file GUI/mainwindow.cpp.

7.16.3.27 void MainWindow::on_btnDetectColor_clicked ( ) [private], [slot]

on_btnDetectColor_clicked if color detection started

Definition at line 1214 of file GUI/mainwindow.cpp.

7.16.3.28 void MainWindow::on_camSourceSpin_valueChanged (int *arg1*) [private], [slot]

on_camSourceSpin_valueChanged emit the changed arguments for cam

Parameters

in	*arg1(int)*	

7.16.3.29 void MainWindow::on_camSourceSpin_valueChanged (int *arg1*) [private], [slot]

on_camSourceSpin_valueChanged emit the changed arguments for cam

Parameters

in	*arg1(int)*	

Definition at line 1075 of file GUI/mainwindow.cpp.

7.16.3.30 void MainWindow::on_cbStream_stateChanged (int *arg1*) [private], [slot]

on_cbStream_stateChanged emit the changed arguments for cam

Parameters

in	*arg1*	(int)

7.16.3.31 void MainWindow::on_cbStream_stateChanged (int *arg1*) [private], [slot]

on_cbStream_stateChanged emit the changed arguments for cam

Parameters

in	*arg1*	(int)

Definition at line 1150 of file GUI/mainwindow.cpp.

7.16.3.32 void MainWindow::on_logCBActor_stateChanged (int *arg1*) [private], [slot]

on_logCBActor_stateChanged if Actor combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.33 void MainWindow::on_logCBActor_stateChanged (int *arg1*) [private], [slot]

on_logCBActor_stateChanged if Actor combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1098 of file GUI/mainwindow.cpp.

7.16.3.34 void MainWindow::on_logCBAI_stateChanged (int *arg1*) [private], [slot]

on_logCBAI_stateChanged if AI combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.35 void MainWindow::on_logCBAI_stateChanged (int *arg1*) [private], [slot]

on_logCBAI_stateChanged if AI combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1099 of file GUI/mainwindow.cpp.

7.16.3.36 void MainWindow::on_logCBData_stateChanged (int *arg1*) [private], [slot]

on_logCBData_stateChanged if Data combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.37 void MainWindow::on_logCBData_stateChanged (int *arg1*) [private], [slot]

on_logCBData_stateChanged if Data combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1100 of file GUI/mainwindow.cpp.

7.16.3.38 void MainWindow::on_logCBOther_stateChanged (int *arg1*) [private], [slot]

on_logCBOther_stateChanged if Other combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.39 void MainWindow::on_logCBOther_stateChanged (int *arg1*) [private], [slot]

on_logCBOther_stateChanged if Other combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1097 of file GUI/mainwindow.cpp.

7.16.3.40 void MainWindow::on_logCBPlot_stateChanged (int *arg1*) [private], [slot]

on_logCBPlot_stateChanged if Plot combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.41 void MainWindow::on_logCBPlot_stateChanged (int *arg1*) [private], [slot]

on_logCBPlot_stateChanged if Plot combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1102 of file GUI/mainwindow.cpp.

7.16.3.42 void MainWindow::on_logCBSensor_stateChanged (int *arg1*) [private], [slot]

on_logCBSensor_stateChanged if Sensor combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

7.16.3.43 void MainWindow::on_logCBSensor_stateChanged (int *arg1*) [private], [slot]

on_logCBSensor_stateChanged if Sensor combobox changed -> emit the changed logging values

Parameters

in	*arg1(int)*	

Definition at line 1101 of file GUI/mainwindow.cpp.

7.16.3.44 void MainWindow::on_logLevel_currentIndexChanged (int *index*) [private], [slot]

on_logLevel_currentIndexChanged if logLevel changed -> emit the changed logging values

Parameters

in	*index(int)*	

7.16.3.45 void MainWindow::on_logLevel_currentIndexChanged (int *index*) [private], [slot]

on_logLevel_currentIndexChanged if logLevel changed -> emit the changed logging values

Parameters

in	*index(int)*	

Definition at line 1103 of file GUI/mainwindow.cpp.

7.16.3.46 void MainWindow::on_pushButton_StartOrientation_clicked ( ) [private], [slot]

on_pushButton_StartOrientation_clicked if start orientation was clicked

Definition at line 1208 of file GUI/mainwindow.cpp.

7.16.3.47 void MainWindow::on_refreshTime_valueChanged (int *arg1*) [private], [slot]

on_refreshTime_valueChanged if refresh time spin box changed.

Parameters

in	*arg1(int)*	

Definition at line 1203 of file GUI/mainwindow.cpp.

7.16.3.48 void MainWindow::on_sizeX_textChanged (const QString & *arg1*) [private], [slot]

on_sizeX_textChanged emit the changed arguments for cam

Parameters

in	*arg1(QString)*	

7.16.3.49 void MainWindow::on_sizeX_textChanged (const QString & *arg1*) [private], [slot]

on_sizeX_textChanged emit the changed arguments for cam

Parameters

in	*arg1(QString)*	

Definition at line 1077 of file GUI/mainwindow.cpp.

7.16.3.50 void MainWindow::on_sizeY_textChanged (const QString & *arg1*) [private], [slot]

on_sizeY_textChanged emit the changed arguments for cam

Parameters

in	*arg1(QString)*	

7.16.3.51 void MainWindow::on_sizeY_textChanged (const QString & *arg1*) [private], [slot]

on_sizeY_textChanged emit the changed arguments for cam

Parameters

in	*arg1(QString)*	

Definition at line 1078 of file GUI/mainwindow.cpp.

7.16.3.52 void MainWindow::on_spinBox_kernel_valueChanged (int *arg1*) [private], [slot]

on_spinBox_kernel_valueChanged if kernel size of spin box has changed

Parameters

in	*arg1(int)*	

Definition at line 1117 of file GUI/mainwindow.cpp.

7.16.3.53 void MainWindow::on_spinPIDAD_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAD_valueChanged if angle PID D value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.54 void MainWindow::on_spinPIDAD_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAD_valueChanged if angle PID D value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1127 of file GUI/mainwindow.cpp.

7.16.3.55 void MainWindow::on_spinPIDAI_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAI_valueChanged if angle PID I value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.56 void MainWindow::on_spinPIDAI_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAI_valueChanged if angle PID I value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1126 of file GUI/mainwindow.cpp.

7.16.3.57 void MainWindow::on_spinPIDAP_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAP_valueChanged if angle PID P value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.58 void MainWindow::on_spinPIDAP_valueChanged (double *arg1*) [private], [slot]

on_spinPIDAP_valueChanged if angle PID P value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1125 of file GUI/mainwindow.cpp.

7.16.3.59 void MainWindow::on_spinPIDVD_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVD_valueChanged if velocity PID D value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.60 void MainWindow::on_spinPIDVD_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVD_valueChanged if velocity PID D value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1130 of file GUI/mainwindow.cpp.

7.16.3.61 void MainWindow::on_spinPIDVI_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVI_valueChanged if velocity PID I value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.62 void MainWindow::on_spinPIDVI_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVI_valueChanged if velocity PID I value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1129 of file GUI/mainwindow.cpp.

7.16.3.63 void MainWindow::on_spinPIDVP_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVP_valueChanged if velocity PID P value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

7.16.3.64 void MainWindow::on_spinPIDVP_valueChanged (double *arg1*) [private], [slot]

on_spinPIDVP_valueChanged if velocity PID P value changed -> emit the changed PID values

Parameters

in	*arg1*	(double)

Definition at line 1128 of file GUI/mainwindow.cpp.

7.16.3.65 void MainWindow::on_streamLog_stateChanged (int *arg1*) [private], [slot]

7.16.3.66 void MainWindow::on_streamLog_stateChanged (int *arg1*) [private], [slot]

on_streamLog_stateChanged check box if we want to see the Log-stream

Parameters

in	*arg1(int)*	

Definition at line 1182 of file GUI/mainwindow.cpp.

7.16.3.67 void MainWindow::on_streamPath_stateChanged (int *arg1*) [private], [slot]

7.16.3.68 void MainWindow::on_streamPath_stateChanged (int *arg1*) [private], [slot]

on_streamPath_stateChanged check box if we want to see the path stream

Parameters

in	*arg1(int)*	

Definition at line 1166 of file GUI/mainwindow.cpp.

7.16.3.69 void MainWindow::on_streamPID_stateChanged (int *arg1*) [private], [slot]

7.16.3.70 void MainWindow::on_streamPID_stateChanged (int *arg1*) [private], [slot]

on_streamPID_stateChanged check box if we want to see the PID-stream

Parameters

in	*arg1*	(int)

Definition at line 1174 of file GUI/mainwindow.cpp.

7.16.3.71 void MainWindow::on_streamSensor_stateChanged (int *arg1*) [private], [slot]

7.16.3.72 void MainWindow::on_streamSensor_stateChanged (int *arg1*) [private], [slot]

on_streamSensor_stateChanged check box if we want to see the sensor stream

Parameters

in	*arg1*	(int)

Definition at line 1158 of file GUI/mainwindow.cpp.

7.16.3.73 void MainWindow::on_updateSpinner_valueChanged (int *arg1*) [private], [slot]

on_updateSpinner_valueChanged emit the changed arguments for cam

Parameters

in	*arg1(int)*	

7.16.3.74 void MainWindow::on_updateSpinner_valueChanged (int *arg1*) [private], [slot]

on_updateSpinner_valueChanged emit the changed arguments for cam

Parameters

in	*arg1(int)*	

Definition at line 1076 of file GUI/mainwindow.cpp.

7.16.3.75 void MainWindow::orientationSetup ( ) [private], [slot]

orientationSetup creates the orientation for pucks and poles

Definition at line 1013 of file mainwindow.cpp.

7.16.3.76 void MainWindow::refresh ( ) [private], [slot]

refresh the GUI

7.16.3.77 void MainWindow::refresh ( ) [private], [slot]

refresh the GUI

Definition at line 272 of file GUI/mainwindow.cpp.

7.16.3.78 void MainWindow::right ( ) [private], [slot]

right button of remote control

7.16.3.79 void MainWindow::right ( ) [private], [slot]

right button of remote control

Definition at line 896 of file GUI/mainwindow.cpp.

7.16.3.80 void MainWindow::robotRemoteControllUpdate (double *velocity,* double *degreeturn*) [signal]

robotRemoteControllUpdate will send remote control parameter to lowLevelActor

Parameters

in	*velocity*	in m/s
in	*degreeturn*	in rad/s

7.16.3.81 void MainWindow::robotRemoteControllUpdate (double *velocity,* double *degreeturn*) [signal]

robotRemoteControllUpdate will send remote control parameter to lowLevelActor

Parameters

in	*velocity*	in m/s
in	*degreeturn*	in rad/s

7.16.3.82 void MainWindow::setrefreshrate ( ) [private], [slot]

setrefreshrate set the refresh cycle

Definition at line 646 of file mainwindow.cpp.

7.16.3.83 void MainWindow::setup ( )

setup will transmit the GUI params to created RoboThread

7.16.3.84 void MainWindow::setup ( )

setup will transmit the GUI params to created RoboThread

Definition at line 216 of file GUI/mainwindow.cpp.

7.16.3.85 void MainWindow::signalChangeCamParams (CameraParams *cp*) [signal]

signalChangeCamParams will send the changed cam parameters

Parameters

in	*cp*	(CameraParams)

7.16.3.86 void MainWindow::signalChangeCamParams (CameraParams *cp*) [signal]

signalChangeCamParams will send the changed cam parameters

Parameters

in	*cp*	(CameraParams)

7.16.3.87 void MainWindow::signalChangeFilterParams (FilterParams *cp*) [signal]

signalChangeFilterParams will emit the changed filter params

Parameters

in	*cp*	(FilterParams)

7.16.3.88 void MainWindow::signalChangeFilterParams (FilterParams *cp*) [signal]

signalChangeFilterParams will emit the changed filter params

Parameters

in	*cp*	(FilterParams)

7.16.3.89 void MainWindow::signalChangePIDParams (PIDParams *p*) [signal]

signalChangePIDParams will emit the changed PID params

Parameters

in	*p*	(PIDParams-Enum)

7.16.3.90 void MainWindow::signalChangePIDParams (PIDParams *p*) [signal]

signalChangePIDParams will emit the changed PID params

Parameters

in	*p*	(PIDParams-Enum)

7.16.3.91 void MainWindow::signalStartOrientation (bool *change*) [signal]

7.16.3.92 void MainWindow::signalStartOrientation (bool *change*) [signal]

signalStartOrientation will emulate the signal from refree to start orientation

Parameters

in	*change*	(bool)

7.16.3.93 void MainWindow::signalTestColorDetect ( ) [signal]

signalTestColorDetect emit the signal for color detection

7.16.3.94 void MainWindow::signalTestPuckRelease ( ) [signal]

signalTestPuckRelease emit the signal for releasing the pucks

7.16.3.95 void MainWindow::signalTestStartGame ( ) [signal]

signalTestStartGame emit the signal to start the game (refree signal)

7.16.3.96 void MainWindow::slotDisplayFrame (cv::Mat *mat*) [slot]

slotDisplayFrame will show given frames gathered by the cam

Parameters

in	*mat*	

Definition at line 841 of file GUI/mainwindow.cpp.

7.16.3.97 void MainWindow::slotDisplayFrame (cv::Mat *mat*) [slot]

slotDisplayFrame will show given frames gathered by the cam

Parameters

in	*mat*	

7.16.3.98 void MainWindow::slotLaserDisplay (LaserPlotData *laserData*) [slot]

slotLaserDisplay will display the gathered laserData

Parameters

in	*laserData*	

Definition at line 705 of file GUI/mainwindow.cpp.

7.16.3.99 void MainWindow::slotLaserDisplay (LaserPlotData *laserData*) [slot]

slotLaserDisplay will display the gathered laserData

Parameters

in	*laserData*	

7.16.3.100 void MainWindow::slotLog (QString *html*) [slot]

slotLog will display logging messages

Parameters

in	*html*	

Definition at line 211 of file GUI/mainwindow.cpp.

7.16.3.101 void MainWindow::slotLog (QString *html*) [slot]

slotLog will display logging messages

Parameters

in	*html*	

7.16.3.102 void MainWindow::slotPIDPlot (PIDPlotData *d*) [slot]

slotPIDPlot will plot gathered PID-informations

Parameters

in	*d*	

Definition at line 846 of file GUI/mainwindow.cpp.

7.16.3.103 void MainWindow::slotPIDPlot (PIDPlotData *d*) [slot]

slotPIDPlot will plot gathered PID-informations

Parameters

in	*d*	

7.16.3.104 void MainWindow::slotRestartTimerDisplay ( ) [slot]

slotRestartTimerDisplay

Definition at line 237 of file GUI/mainwindow.cpp.

7.16.3.105 void MainWindow::slotSimulationDetect ( ) [private], [slot]

slotSimulationDetect if the we noticed that we are in simulation

Definition at line 1145 of file GUI/mainwindow.cpp.

7.16.3.106 void MainWindow::slotUpdateColorLabel (CamColor *color*) [slot]

slotUpdateColorLabel

Parameters

in	*color*	

Definition at line 242 of file GUI/mainwindow.cpp.

7.16.3.107 void MainWindow::stop ( ) [private], [slot]

stop button of remote control

7.16.3.108 void MainWindow::stop ( ) [private], [slot]

stop button of remote control

Definition at line 920 of file GUI/mainwindow.cpp.

7.16.3.109 void MainWindow::strongleft ( ) [private], [slot]

strongleft button of remote control

7.16.3.110 void MainWindow::strongleft ( ) [private], [slot]

strongleft button of remote control

Definition at line 904 of file GUI/mainwindow.cpp.

7.16.3.111 void MainWindow::strongright ( ) [private], [slot]

strongright button of remote control

7.16.3.112 void MainWindow::strongright ( ) [private], [slot]

strongright button of remote control

Definition at line 912 of file GUI/mainwindow.cpp.

7.16.3.113 void MainWindow::updatePathDisplay (PathPlotData *dataPacket*) [slot]

updatePathDisplay will update the path information

Parameters

in	*dataPacket*	

Definition at line 759 of file GUI/mainwindow.cpp.

7.16.3.114 void MainWindow::updatePathDisplay (PathPlotData *dataPacket*) [slot]

updatePathDisplay will update the path information

Parameters

in	*dataPacket*	

7.16.3.115 void MainWindow::updateRemoteOdometry (Position) [signal]

updateRemoteOdometry

7.16.3.116 void MainWindow::updateRemoteOdometry (Position) [signal]

updateRemoteOdometry

7.16.4 Member Data Documentation

7.16.4.1 QCPColorMap * MainWindow::colorMap [private]

create the colormap for heat map of path planning

Definition at line 189 of file GUI/mainwindow.h.

7.16.4.2 QList< QColor > MainWindow::colors [private]

coler of the cam tabbed view

Definition at line 197 of file GUI/mainwindow.h.

7.16.4.3 QGraphicsScene* MainWindow::graphicsScene_1 [private]

Definition at line 168 of file mainwindow.h.

7.16.4.4 QGraphicsScene* MainWindow::graphicsScene_2 [private]

Definition at line 169 of file mainwindow.h.

7.16.4.5 QGraphicsScene* MainWindow::graphicsScene_3 [private]

Definition at line 170 of file mainwindow.h.

7.16.4.6 QGraphicsScene* MainWindow::graphicsScene_4 [private]

Definition at line 171 of file mainwindow.h.

7.16.4.7 QGraphicsScene* MainWindow::graphicsScene_5 [private]

Definition at line 172 of file mainwindow.h.

7.16.4.8 QGraphicsScene* MainWindow::graphicsScene_6 [private]

Definition at line 173 of file mainwindow.h.

7.16.4.9 QCPGraph * MainWindow::graphLaserMedian [private]

median filtered laser data

Definition at line 184 of file GUI/mainwindow.h.

7.16.4.10 QCPGraph * MainWindow::graphLaserObjects [private]

recognised laser objects

Definition at line 185 of file GUI/mainwindow.h.

7.16.4.11 QCPGraph * MainWindow::graphLaserRaw [private]

raw graph of laser data

Definition at line 182 of file GUI/mainwindow.h.

7.16.4.12 QCPGraph * MainWindow::graphLaserReduced [private]

reduced graph laser data

Definition at line 183 of file GUI/mainwindow.h.

7.16.4.13 QCPGraph * MainWindow::graphPIDAIst [private]

current PID angel

Definition at line 214 of file GUI/mainwindow.h.

7.16.4.14 QCPGraph * MainWindow::graphPIDASoll [private]

desired PID angel

Definition at line 214 of file GUI/mainwindow.h.

7.16.4.15 QCPGraph * MainWindow::graphPIDDIst [private]

current PID velocity

Definition at line 214 of file GUI/mainwindow.h.

7.16.4.16 QCPGraph * MainWindow::graphPIDDSoll [private]

desired PID velocity

Definition at line 214 of file GUI/mainwindow.h.

7.16.4.17 QCPGraph * MainWindow::graphRobotScatter [private]

scatter graph for robot

Definition at line 192 of file GUI/mainwindow.h.

7.16.4.18 QCPCurve * MainWindow::graphSplineCurve [private]

create spline curve

Definition at line 194 of file GUI/mainwindow.h.

7.16.4.19 QCPGraph * MainWindow::graphWPScatter [private]

scatter graphf for waypoints

Definition at line 193 of file GUI/mainwindow.h.

7.16.4.20 bool MainWindow::m_changeOrientation [private]

Definition at line 128 of file mainwindow.h.

7.16.4.21 QGraphicsScene * MainWindow::map [private]

graphic for displaying the robot with pucks and poles

Definition at line 153 of file GUI/mainwindow.h.

7.16.4.22 QTimer * MainWindow::refreshtimer [private]

Timer to refresh the GUI

Definition at line 154 of file GUI/mainwindow.h.

7.16.4.23 QCPScatterStyle * MainWindow::scatterLaserObjects [private]

Laser object as scatterplot

Definition at line 186 of file GUI/mainwindow.h.

7.16.4.24 QCPScatterStyle * MainWindow::scatterRobotStyle [private]

Create the robot as scatter plot

Definition at line 190 of file GUI/mainwindow.h.

7.16.4.25 QCPScatterStyle * MainWindow::scatterWPStyle [private]

Waypoints as scatter plot

Definition at line 191 of file GUI/mainwindow.h.

7.16.4.26 QElapsedTimer MainWindow::timer [private]

Für Anzeige vergangener zeit.

Definition at line 156 of file GUI/mainwindow.h.

7.16.4.27 QList< QPair< int, int > > MainWindow::TrackingMe [private]

tracking own robot

Definition at line 178 of file GUI/mainwindow.h.

7.16.4.28 QList< QPair< int, int > > MainWindow::TrackingYou [private]

of foe bot

Definition at line 179 of file GUI/mainwindow.h.

7.16.4.29 Ui::MainWindow * MainWindow::ui [private]

main user interface

Definition at line 152 of file GUI/mainwindow.h.

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

- GUI/mainwindow.h
- mainwindow.h
- GUI/mainwindow.cpp
- mainwindow.cpp

7.17 MapData Class Reference

The MapData class is a static class for inter-thread communication and saving information for other parts of the programm.

`#include <mapdata.h>`

Collaboration diagram for MapData:

Static Public Member Functions

- static void setActualColor (const TeamColor color)

  setActualColor , sets the current team color
- static void setProbableColor (const CamColor color)

  setProbableColor
- static CamColor getTeamColor ()

  getTeamColor
- static QList< Obstacle > getObstacle (const ObstacleType &type)

  getObstacle return the obstacleList of given type
- static QList< Obstacle > getObstacle (const ObstacleType &type, const ObstacleColor &color)

  getObstacle return the obstacleList for given type and color
- static QList< Obstacle > getObstacle (const ObstacleType &type, const ObstacleStatus &status)

  getObstacle return the obstacleList for given type and color

- static QList< Obstacle > getObstacle (const ObstacleType &type, const ObstacleColor &color, const ObstacleStatus &status)

  getObstacle return the obstacleList of given type, color and status
- static bool setObstacle (const QList< Obstacle > &list)

  setObstacle will write a QList to merging process
- static bool setObstacle (const QList< Obstacle > &list, const Position &currentRoboPos)

  setObstacle
- static bool setObstacle (const Obstacle &value)

  setObstacle will write a single value to merging process
- static bool setObstacle (const Obstacle &value, const Position &currentRoboPos)

  setObstacle will write a single value to merging process
- static Position getRobotPosition (ObstacleType type=ObstacleType::ME)

  getRobotPosition will return a Position struct with the current XYR values of a robot
- static bool deleteObstacle (ObstacleType type=ObstacleType::DUMMY)

  deleteObstacle
- static void clearTargets (void)

  clearTargets will remove all target waypoints
- static Obstacle getFirstTarget (void)

  getFirstTarget will return the first targetpoint
- static void deleteFirstTarget (void)

  deleteFirstTarget will delete the first item in targetliste
- static bool getDisableEmergency ()

  getDisableEmergency getter for disableEmergency
- static void setDisableEmergency (bool value)

  setDisableEmergency setter for disableEmergency
- static bool getSimulationDetected ()

  getSimulationDetected getter for simulationDetected
- static void setSimulationDetected (bool value)

  setSimulationDetected setter for simulationDetected
- static bool isPuckInFork ()

  getPuckInFork
- static void setPuckInFork (bool value)

  setPuckInFork
- static void setPointerToPathPlanner (PathPlanning *value)

  setPointerToPathPlanner
- static PathPlanning * getPointerToPathPlanner ()

  getPointerToPathPlanner
- static bool getTargetNearEnemy ()

  getTargetNearEnemy
- static void setTargetNearEnemy (bool value)

  setTargetNearEnemy

Static Public Attributes

- static QMutex mutexRobotME
- static QMutex mutexRobotOpponent
- static QMutex mutexRobotDummy
- static QMutex mutexTargets
- static QMutex mutexPucks
- static QMutex mutexPoles
- static QMutex mutexUnidentified

- static QMutex mutexDisableEmergency
- static QMutex mutexSimulationDetected
- static QMutex mutexTeamColor
- static QMutex mutexPuckInFork
- static QMutex mutexPointerToPathPlanner
- static QMutex mutexTargetNearEnemy
- static bool disableEmergency = false
- static bool simulationDetected = true
- static bool puckInFork = false
- static bool targetNearEnemy = false
- static CamColor teamColor = CamColor::NONE

Private Member Functions

- MapData ()

  MapData hidden constructor.
- ~MapData ()

  ~MapData hidden destructor
- MapData (const MapData &)

  MapData as hidden copy constructor.
- MapData & operator= (const MapData &)

  operator = we leave just the declarations, so the compiler will warn us if we try to use those two functions by accident

Static Private Member Functions

- static bool organizeObstacles (const Obstacle &constObstacle, const Position &currentRoboPos)

  organizeObstacles is responsible for merging different obstacles
- static bool compareObstacleTimestamps (Obstacle i, Obstacle j)

  MapData::compareObstacleTimes.
- static QList< Obstacle > getListByType (const ObstacleType &type)

  getListByType will return a QList of Obstacles for given ObstacleType
- static void cleanup (const Position &currentRoboPos, const Position &enemyRobotPosition)

  cleanup functions to tidy up the MapData
- static void checkForEnemyNearTraget (const Position &enemyRobotPosition)

  checkForEnemyNearTraget is verifiying if the enemy position is close to the target

Static Private Attributes

- static PathPlanning * pointerToPathPlanner = nullptr
- static QList< Obstacle > obstaclesPoles = QList<Obstacle>()
- static QList< Obstacle > obstaclesPucks = QList<Obstacle>()
- static QList< Obstacle > obstaclesUnidentified = QList<Obstacle>()
- static QList< Obstacle > obstaclesTargets = QList<Obstacle>()
- static Obstacle obstacleMe = Obstacle()
- static Obstacle obstacleOpponent = Obstacle()
- static Obstacle obstacleDummy = Obstacle()

7.17.1 Detailed Description

The MapData class is a static class for inter-thread communication and saving information for other parts of the programm.

Definition at line 19 of file mapdata.h.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 MapData::MapData ( ) [private]

MapData hidden constructor.

7.17.2.2 MapData::~MapData ( ) [private]

~MapData hidden destructor

7.17.2.3 MapData::MapData (const MapData &) [private]

MapData as hidden copy constructor.

7.17.3 Member Function Documentation

7.17.3.1 void MapData::checkForEnemyNearTraget (const Position & *enemyRobotPosition*) [static], [private]

checkForEnemyNearTraget is verifiying if the enemy position is close to the target

Parameters

in	*enemyRobot-Position*	(Position)

Definition at line 500 of file mapdata.cpp.

7.17.3.2 void MapData::cleanup (const Position & *currentRoboPos,* const Position & *enemyRobotPosition*) [static], [private]

cleanup functions to tidy up the MapData

Parameters

in	*currentRoboPos*	(Position)
in	*enemyRobot-Position*	(Position)

MUTEX POLE

Definition at line 397 of file mapdata.cpp.

7.17.3.3 void MapData::clearTargets (void) [static]

clearTargets will remove all target waypoints

Definition at line 353 of file mapdata.cpp.

7.17.3.4 bool MapData::compareObstacleTimestamps (Obstacle *i,* Obstacle *j*) [static], [private]

MapData::compareObstacleTimes.

Parameters

in	*i*	Obstacle 1
in	*j*	Obstacle 2

Returns

Definition at line 823 of file mapdata.cpp.

7.17.3.5 void MapData::deleteFirstTarget (void) [static]

deleteFirstTarget will delete the first item in targetliste

Definition at line 370 of file mapdata.cpp.

7.17.3.6 bool MapData::deleteObstacle (ObstacleType *type* = ObstacleType::DUMMY) [static]

deleteObstacle

Parameters

in	*type*	(ME, DUMMY, OPPONENT)

Returns

Definition at line 265 of file mapdata.cpp.

7.17.3.7 bool MapData::getDisableEmergency ( ) [static]

getDisableEmergency getter for disableEmergency

Returns

a boolen if emergency is enabled

Definition at line 103 of file mapdata.cpp.

7.17.3.8 Obstacle MapData::getFirstTarget (void) [static]

getFirstTarget will return the first targetpoint

Returns

an Obstacle being the first target point of target-list

Definition at line 360 of file mapdata.cpp.

7.17.3.9 QList< Obstacle > MapData::getListByType (const ObstacleType & *type*) [static], [private]

getListByType will return a QList of Obstacles for given ObstacleType

Parameters

in	*type*	(ObstacleType)

Returns

a QList of Obstacle for given ObstacleType

Todo warning scheimssen

Definition at line 45 of file mapdata.cpp.

7.17.3.10 QList< Obstacle > MapData::getObstacle (const ObstacleType & *type*) [static]

getObstacle return the obstacleList of given type

Parameters

in	*type*	(ObstacleType)

Returns

a QList of Obstacle for given type

Definition at line 195 of file mapdata.cpp.

7.17.3.11 QList< Obstacle > MapData::getObstacle (const ObstacleType & *type*, const ObstacleColor & *color*) [static]

getObstacle return the obstacleList for given type and color

Parameters

in	*type*	(ObstacleType)
in	*color*	(ObstacleColor)

Returns

a QList of Obstacle for given type and color

Definition at line 200 of file mapdata.cpp.

7.17.3.12 QList< Obstacle > MapData::getObstacle (const ObstacleType & *type*, const ObstacleStatus & *status*) [static]

getObstacle return the obstacleList for given type and color

Parameters

in	*type*	(ObstacleType)
in	*status*	(ObstacleStatus)

Returns

a QList of Obstacle for given type and status

Definition at line 214 of file mapdata.cpp.

7.17.3.13 QList< Obstacle > MapData::getObstacle (const ObstacleType & *type*, const ObstacleColor & *color*, const ObstacleStatus & *status*) [static]

getObstacle return the obstacleList of given type, color and status

Parameters

in	*type*	(ObstacleType)
in	*color*	(ObstacleColor)
in	*status*	(ObstacleStatus)

Returns

a QList of Obstacle for given type, color and status.

Definition at line 228 of file mapdata.cpp.

7.17.3.14 PathPlanning * MapData::getPointerToPathPlanner ( ) [static]

getPointerToPathPlanner

Returns

PathPlanning*

Definition at line 145 of file mapdata.cpp.

7.17.3.15 Position MapData::getRobotPosition (ObstacleType *type* = ObstacleType::ME) [static]

getRobotPosition will return a Position struct with the current XYR values of a robot

Parameters

in	*type*	(OPPONENT or ME)

Returns

Position of the desired robot

Definition at line 242 of file mapdata.cpp.

7.17.3.16 bool MapData::getSimulationDetected ( ) [static]

getSimulationDetected getter for simulationDetected

Returns

a boolen if emergency is enabled

Definition at line 115 of file mapdata.cpp.

7.17.3.17 bool MapData::getTargetNearEnemy ( ) [static]

getTargetNearEnemy

Returns

true if the target is near the enemy robot

Definition at line 153 of file mapdata.cpp.

7.17.3.18 CamColor MapData::getTeamColor ( ) [static]

getTeamColor

Returns

Definition at line 189 of file mapdata.cpp.

7.17.3.19 bool MapData::isPuckInFork ( ) [static]

getPuckInFork

Returns

Definition at line 127 of file mapdata.cpp.

7.17.3.20 MapData& MapData::operator= (const MapData &) [private]

operator = we leave just the declarations, so the compiler will warn us if we try to use those two functions by accident

Returns

assigned value

7.17.3.21 bool MapData::organizeObstacles (const Obstacle & *constObstacle,* const Position & *currentRoboPos*) [static], [private]

organizeObstacles is responsible for merging different obstacles

Adding items to the map according to their type.

Parameters

in	*constObstacle*	(Obstacle) will be stored in representing list<Obstacle>
in	*currentRoboPos*	(Position)

Returns

true if successful

Todo : klären wie verschiewdene obstacles gemerged werden

MUTEX POLE

ISINFORK

MUTEX PUCKS

MUTEX PUCKS

MUTEX PUCKS

Todo : warning für default state

Definition at line 523 of file mapdata.cpp.

7.17.3.22 void MapData::setActualColor (const TeamColor *color*) [static]

setActualColor , sets the current team color

Parameters

in	*color*	(TeamColor enumeration)

Definition at line 167 of file mapdata.cpp.

7.17.3.23 void MapData::setDisableEmergency (bool *value*) [static]

setDisableEmergency setter for disableEmergency

Parameters

in	*value*	(boolean)

Definition at line 109 of file mapdata.cpp.

7.17.3.24 bool MapData::setObstacle (const QList< Obstacle > & *list*) [static]

setObstacle will write a QList to merging process

Parameters

in	*list*	(QList of Obstacle)

Returns

true if successful

Definition at line 293 of file mapdata.cpp.

7.17.3.25 bool MapData::setObstacle (const QList< Obstacle > & *list,* const Position & *currentRoboPos*) [static]

setObstacle

Parameters

in	*list*	(QList of Obstacle)
in	*currentRoboPos*	

Returns

true if successful

Definition at line 312 of file mapdata.cpp.

7.17.3.26 bool MapData::setObstacle (const Obstacle & *value*) [static]

setObstacle will write a single value to merging process

Parameters

in	*value*	

Returns

true if successful

Definition at line 328 of file mapdata.cpp.

7.17.3.27 bool MapData::setObstacle (const Obstacle & *value,* const Position & *currentRoboPos*) [static]

setObstacle will write a single value to merging process

Parameters

in	*value*	
in	*currentRoboPos*	

Returns

true if successful

Definition at line 342 of file mapdata.cpp.

7.17.3.28 void MapData::setPointerToPathPlanner (PathPlanning * *value*) [static]

setPointerToPathPlanner

Parameters

in	*value*	(PathPlanning*)

Definition at line 139 of file mapdata.cpp.

7.17.3.29 void MapData::setProbableColor (const CamColor *color*) [static]

setProbableColor

Parameters

in	*color*	

Definition at line 183 of file mapdata.cpp.

7.17.3.30 void MapData::setPuckInFork (bool *value*) [static]

setPuckInFork

Parameters

in	*value*	(bool)

Definition at line 133 of file mapdata.cpp.

7.17.3.31 void MapData::setSimulationDetected (bool *value*) [static]

setSimulationDetected setter for simulationDetected

Parameters

in	*value*	(boolean)

Definition at line 121 of file mapdata.cpp.

7.17.3.32 void MapData::setTargetNearEnemy (bool *value*) [static]

setTargetNearEnemy

Parameters

in	*value*	(bool)

Definition at line 160 of file mapdata.cpp.

7.17.4 Member Data Documentation

7.17.4.1 bool MapData::disableEmergency = false [static]

Boolean if the emergency signal should be considered or not

Definition at line 76 of file mapdata.h.

7.17.4.2 QMutex MapData::mutexDisableEmergency [static]

Mutex to ensure the secure multi-threaded access on disable emergency variable

Definition at line 62 of file mapdata.h.

7.17.4.3 QMutex MapData::mutexPointerToPathPlanner [static]

Mutex to ensure the secure multi-threaded access on Pointer

Definition at line 62 of file mapdata.h.

7.17.4.4 QMutex MapData::mutexPoles [static]

Mutex to ensure the secure multi-threaded access on pole list

Definition at line 62 of file mapdata.h.

7.17.4.5 QMutex MapData::mutexPuckInFork [static]

Mutex to ensure the secure multi-threaded access on PuckInFork

Definition at line 62 of file mapdata.h.

7.17.4.6 QMutex MapData::mutexPucks [static]

Mutex to ensure the secure multi-threaded access on puck list

Definition at line 62 of file mapdata.h.

7.17.4.7 QMutex MapData::mutexRobotDummy [static]

Mutex to ensure the secure multi-threaded access on dummy robot object

Definition at line 62 of file mapdata.h.

7.17.4.8 QMutex MapData::mutexRobotME [static]

Mutex to ensure the secure multi-threaded access on my robot object

Definition at line 62 of file mapdata.h.

7.17.4.9 QMutex MapData::mutexRobotOpponent [static]

Mutex to ensure the secure multi-threaded access on enemy robot object

Definition at line 62 of file mapdata.h.

7.17.4.10 QMutex MapData::mutexSimulationDetected [static]

Mutex to ensure the secure multi-threaded access on simulation detected variable

Definition at line 62 of file mapdata.h.

7.17.4.11 QMutex MapData::mutexTargetNearEnemy [static]

Mutex to ensure the secure multi-threaded access on TargetNearEnemy

Definition at line 62 of file mapdata.h.

7.17.4.12 QMutex MapData::mutexTargets [static]

Mutex to ensure the secure multi-threaded access on target list

Definition at line 62 of file mapdata.h.

7.17.4.13 QMutex MapData::mutexTeamColor [static]

Mutex to ensure the secure multi-threaded access on team color enum

Definition at line 62 of file mapdata.h.

7.17.4.14 QMutex MapData::mutexUnidentified [static]

Mutex to ensure the secure multi-threaded access on unidentified list

Definition at line 62 of file mapdata.h.

7.17.4.15 Obstacle MapData::obstacleDummy = Obstacle() [static], [private]

An obstacle object for testing

Definition at line 57 of file mapdata.h.

7.17.4.16 Obstacle MapData::obstacleMe = Obstacle() [static], [private]

The own robot is an obstacle too

Definition at line 57 of file mapdata.h.

7.17.4.17 Obstacle MapData::obstacleOpponent = Obstacle() [static], [private]

The foe robot is also an obstacle

Definition at line 57 of file mapdata.h.

7.17.4.18 QList< Obstacle > MapData::obstaclesPoles = QList<Obstacle>() [static], [private]

QList of obstacles identified as poles

Definition at line 52 of file mapdata.h.

7.17.4.19 QList< Obstacle > MapData::obstaclesPucks = QList<Obstacle>() [static], [private]

QList of obstacles identified as pucks

Definition at line 52 of file mapdata.h.

7.17.4.20 QList< Obstacle > MapData::obstaclesTargets = QList<Obstacle>() [static], [private]

QList of obstacles which are movement targets

Definition at line 52 of file mapdata.h.

7.17.4.21 QList< Obstacle > MapData::obstaclesUnidentified = QList<Obstacle>() [static], [private]

QList of obstacles being undefined

Definition at line 52 of file mapdata.h.

7.17.4.22 PathPlanning * MapData::pointerToPathPlanner = nullptr [static], [private]

For invoke method

Definition at line 51 of file mapdata.h.

7.17.4.23 bool MapData::puckInFork = false [static]

Boolean if a Puck is in the fork

Definition at line 78 of file mapdata.h.

7.17.4.24 bool MapData::simulationDetected = true [static]

Boolean if a simulation is detected or not

Definition at line 77 of file mapdata.h.

7.17.4.25 bool MapData::targetNearEnemy = false [static]

Boolean if the current target is nex to the enemy robot

Definition at line 79 of file mapdata.h.

7.17.4.26 CamColor MapData::teamColor = CamColor::NONE [static]

Enumeration of team colors <blue>,<yellow>

Definition at line 81 of file mapdata.h.

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

- mapdata.h
- mapdata.cpp

7.18 MedianFilter Class Reference

The MedianFilter class will filter data with an median filter which will return the centered value of a given set.

`#include <medianfilter.h>`

Collaboration diagram for MedianFilter:

Static Public Member Functions

- static std::vector< double > filter (const std::vector< double > &in, const int kernelsize=3)

  filter will filter a given input with an given kernelsize

7.18.1 Detailed Description

The MedianFilter class will filter data with an median filter which will return the centered value of a given set.

Definition at line 10 of file medianfilter.h.

7.18.2 Member Function Documentation

7.18.2.1 std::vector< double > MedianFilter::filter (const std::vector< double > & *in*, const int *kernelsize* = 3) [static]

filter will filter a given input with an given kernelsize

MedianFilter::filter fuehrt einen rolling median mit bestimmter kernelsize ueber dem eingangsvektor aus.

Parameters

in	*in*	(reference of std::vector<double>)
	kernelsize	(kernelsize with an odd number)

Returns

the filtered std::vector<double>

Parameters

in	Eingangsvektor
kernelsize	Aus wievielen Werte soll der Median einer Stelle im Vektor berechnet werden. Muss ungerade und >= 3 sein

Returns

Median-gefilterter Vektor

Definition at line 12 of file medianfilter.cpp.

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

- medianfilter.h
- medianfilter.cpp

7.19 Obstacle Class Reference

The Obstacle class describes all objects on field as obstacles, which can be distinguish by type.

`#include <obstacle.h>`

Collaboration diagram for Obstacle:

Public Member Functions

- Obstacle ()

  Obstacle.
- Obstacle (Position m_Position)

  Obstacle.
- Obstacle (Position m_Position, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords and enumType.
- Obstacle (Position m_Position, ObstacleColor m_enumColor, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords, enumColor and enumType.
- Obstacle (Position m_Position, ObstacleStatus m_enumStatus, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords, enumStatus and enumType.
- Obstacle (Position m_Position, ObstacleColor m_enumColor, ObstacleType m_enumType, ObstacleStatus m_enumStatus)

  Obstacle create an obstacle by setting qpairCoords, enumColor, enumType and enumStatus.
- Obstacle (QPair< double, double > m_qpairCoords)

  Obstacle create an obstacle by setting qpairCoords.
- Obstacle (QPair< double, double > m_qpairCoords, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords and enumType.
- Obstacle (QPair< double, double > m_qpairCoords, ObstacleColor m_enumColor, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords, enumColor and enumType.
- Obstacle (QPair< double, double > m_qpairCoords, ObstacleStatus m_enumStatus, ObstacleType m_enumType)

  Obstacle create an obstacle by setting qpairCoords, enumStatus and enumType.

- Obstacle (QPair< double, double > m_qpairCoords, ObstacleColor m_enumColor, ObstacleType m_enumType, ObstacleStatus m_enumStatus)

  Obstacle create an obstacle by setting qpairCoords, enumColor, enumType and enumStatus.
- Obstacle (QPair< double, double > m_qpairCoords, ObstacleColor m_enumColor, ObstacleType m_enumType, ObstacleStatus m_enumStatus, double m_dOrientation)

  Obstacle create an obstacle by setting qpairCoords, enumColor, enumType, enumStatus, dOrientation.
- virtual ~Obstacle ()

  ~Obstacle
- bool operator== (const Obstacle &b) const

  operator ==
- bool operator< (const Obstacle &b) const

  operator <
- void mergeWith (const Obstacle &newObst)

  mergeWith will merge two obstacles if possible
- ObstacleColor getColor () const

  getColor
- void setColor (const ObstacleColor &value)

  setColor will set the obstacleColor to the given value.
- ObstacleType getType () const

  getType
- void setType (const ObstacleType &value)

  setType of ObstacleType enum
- ObstacleStatus getStatus () const

  getStatus
- void setStatus (const ObstacleStatus &value)

  setStatus the ObstacleStatus enum
- QPair< double, double > getCoords () const

  getCoords
- void setCoords (const QPair< double, double > &value)

  setCoords
- bool getInitialized () const

  getInitialized
- void setInitialized (bool value)

  setInitialized of bInitialized
- QTime getLastUpdate () const

  getLastUpdate-time
- void setLastUpdate (const QTime &value)

  setLastUpdate time
- double getOrientation () const

  getOrientation
- void setOrientation (double value)

  setOrientation
- double getDistanceTo (const Obstacle &obstacle) const

  getDistanceTo
- Position getPosition () const

  getCPosition
- void setPosition (const Position &value)

  setCPosition
- bool isInSpecifiedArea (FieldArea area) const

  isInGoalArea

Private Attributes

- ObstacleColor enumColor
- ObstacleType enumType
- ObstacleStatus enumStatus
- Position cPosition
- bool bInitialized
- QTime cLastUpdate

7.19.1 Detailed Description

The Obstacle class describes all objects on field as obstacles, which can be distinguish by type.

Definition at line 71 of file obstacle.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Obstacle::Obstacle ( )

Obstacle.

Definition at line 222 of file obstacle.cpp.

7.19.2.2 Obstacle::Obstacle (Position *m_Position*)

Obstacle.

Parameters

in	*m_Position*	(class Position)

Definition at line 232 of file obstacle.cpp.

7.19.2.3 Obstacle::Obstacle (Position *m_Position*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords and enumType.

Parameters

in	*m_Position*	(class Position)
in	*m_enumType*	(Type of ObstacleType enum)

Definition at line 242 of file obstacle.cpp.

7.19.2.4 Obstacle::Obstacle (Position *m_Position*, ObstacleColor *m_enumColor*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords, enumColor and enumType.

Parameters

in	*m_Position*	(class Position)
in	*m_enumColor*	(Type of ObstacleType enum)
in	*m_enumType*	(Type from ObstacleColor enum)

Definition at line 249 of file obstacle.cpp.

7.19.2.5 Obstacle::Obstacle (Position *m_Position*, ObstacleStatus *m_enumStatus*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords, enumStatus and enumType.

Parameters

in	*m_Position*	(class Position)
in	*m_enumStatus*	(Type of ObstacleStatus enum)
in	*m_enumType*	(Type from ObstacleColor enum)

Definition at line 258 of file obstacle.cpp.

7.19.2.6 Obstacle::Obstacle (Position *m_Position*, ObstacleColor *m_enumColor*, ObstacleType *m_enumType*, ObstacleStatus *m_enumStatus*)

Obstacle create an obstacle by setting qpairCoords, enumColor, enumType and enumStatus.

Parameters

in	*m_Position*	(class Position)
in	*m_enumColor*	(Type of ObstacleType enum)
in	*m_enumType*	(Type from ObstacleColor enum)
in	*m_enumStatus*	(Type of ObstacleStatus enum)

Definition at line 267 of file obstacle.cpp.

7.19.2.7 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*)

Obstacle create an obstacle by setting qpairCoords.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)

Definition at line 278 of file obstacle.cpp.

7.19.2.8 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords and enumType.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)
in	*m_enumType*	(Type of ObstacleType enum)

Definition at line 288 of file obstacle.cpp.

7.19.2.9 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*, ObstacleColor *m_enumColor*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords, enumColor and enumType.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)
in	*m_enumColor*	(Type of ObstacleType enum)
in	*m_enumType*	(Type from ObstacleColor enum)

Definition at line 301 of file obstacle.cpp.

7.19.2.10 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*, ObstacleStatus *m_enumStatus*, ObstacleType *m_enumType*)

Obstacle create an obstacle by setting qpairCoords, enumStatus and enumType.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)
in	*m_enumStatus*	(Type of ObstacleStatus enum)
in	*m_enumType*	(Type from ObstacleColor enum)

Definition at line 310 of file obstacle.cpp.

7.19.2.11 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*, ObstacleColor *m_enumColor*, ObstacleType *m_enumType*, ObstacleStatus *m_enumStatus*)

Obstacle create an obstacle by setting qpairCoords, enumColor, enumType and enumStatus.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)
in	*m_enumColor*	(Type of ObstacleType enum)
in	*m_enumType*	(Type from ObstacleColor enum)
in	*m_enumStatus*	(Type of ObstacleStatus enum)

Definition at line 319 of file obstacle.cpp.

7.19.2.12 Obstacle::Obstacle (QPair< double, double > *m_qpairCoords*, ObstacleColor *m_enumColor*, ObstacleType *m_enumType*, ObstacleStatus *m_enumStatus*, double *m_dOrientation*)

Obstacle create an obstacle by setting qpairCoords, enumColor, enumType, enumStatus, dOrientation.

Parameters

in	*m_qpairCoords*	(QPair<double,double>)
in	*m_enumColor*	(Type of ObstacleType enum)
in	*m_enumType*	(Type from ObstacleColor enum)
in	*m_enumStatus*	(Type of ObstacleStatus enum)
in	*m_dOrientation*	(double)

Definition at line 330 of file obstacle.cpp.

7.19.2.13 Obstacle::~Obstacle ( ) [virtual]

~Obstacle

Definition at line 343 of file obstacle.cpp.

7.19.3 Member Function Documentation

7.19.3.1 ObstacleColor Obstacle::getColor ( ) const

getColor

Returns

the colorEnum

Definition at line 4 of file obstacle.cpp.

7.19.3.2 QPair< double, double > Obstacle::getCoords ( ) const

getCoords

Returns

a QPair<double,double> with the current coordinates

Definition at line 34 of file obstacle.cpp.

7.19.3.3 double Obstacle::getDistanceTo (const Obstacle & *obstacle*) const

getDistanceTo

Parameters

in	*obstacle*	

Returns

the distance to the obstacle

Definition at line 78 of file obstacle.cpp.

7.19.3.4 bool Obstacle::getInitialized ( ) const

getInitialized

Returns

the value of bInitialized

Definition at line 46 of file obstacle.cpp.

7.19.3.5 QTime Obstacle::getLastUpdate ( ) const

getLastUpdate-time

Returns

the last update time (QTime)

Definition at line 56 of file obstacle.cpp.

7.19.3.6 double Obstacle::getOrientation ( ) const

getOrientation

Returns

the current orientation (double)

position.rot(value) constrains radian between 0 and 2∗PI

Definition at line 67 of file obstacle.cpp.

7.19.3.7 Position Obstacle::getPosition ( ) const

getCPosition

Returns

Definition at line 85 of file obstacle.cpp.

7.19.3.8 ObstacleStatus Obstacle::getStatus ( ) const

getStatus

Returns

the ObstacleStatus enum

Definition at line 24 of file obstacle.cpp.

7.19.3.9 ObstacleType Obstacle::getType ( ) const

getType

Returns

the type Enum

Definition at line 14 of file obstacle.cpp.

7.19.3.10 bool Obstacle::isInSpecifiedArea (FieldArea *area*) const

isInGoalArea

Parameters

in	*myGoalArea*	(FieldArea)

Returns

true if the obstacle is in the specified area

Definition at line 95 of file obstacle.cpp.

7.19.3.11 void Obstacle::mergeWith (const Obstacle & *newObst*)

mergeWith will merge two obstacles if possible

Parameters

b	(reference on an Obstacle)

Todo merge conditions are a topic for discussion

Todo average of coordinates

Definition at line 414 of file obstacle.cpp.

7.19.3.12 bool Obstacle::operator< (const Obstacle & *b*) const

operator <

Parameters

b	

Returns

which of the given obstacles is greater

Todo check condition

Definition at line 392 of file obstacle.cpp.

7.19.3.13 bool Obstacle::operator== (const Obstacle & *b*) const

operator ==

Parameters

b	

Returns

if two obstacles are equal

Definition at line 347 of file obstacle.cpp.

7.19.3.14 void Obstacle::setColor (const ObstacleColor & *value*)

setColor will set the obstacleColor to the given value.

Parameters

in	*value*	(reference of ObstacleColor)

Definition at line 9 of file obstacle.cpp.

7.19.3.15 void Obstacle::setCoords (const QPair< double, double > & *value*)

setCoords

Parameters

in	*value*	(QPair<double,double>)

Definition at line 39 of file obstacle.cpp.

7.19.3.16 void Obstacle::setInitialized (bool *value*)

setInitialized of bInitialized

Parameters

in	*value*	(bool)

Definition at line 51 of file obstacle.cpp.

7.19.3.17 void Obstacle::setLastUpdate (const QTime & *value*)

setLastUpdate time

Parameters

in	*value*	(reference of QTime)

radian between 0 and 2∗PI

Definition at line 61 of file obstacle.cpp.

7.19.3.18 void Obstacle::setOrientation (double *value*)

setOrientation

Parameters

in	*value*	

Definition at line 73 of file obstacle.cpp.

7.19.3.19 void Obstacle::setPosition (const Position & *value*)

setCPosition

Parameters

value	

Definition at line 90 of file obstacle.cpp.

7.19.3.20 void Obstacle::setStatus (const ObstacleStatus & *value*)

setStatus the ObstacleStatus enum

Parameters

in	*value*	(reference ObstacleStatus)

Definition at line 29 of file obstacle.cpp.

7.19.3.21 void Obstacle::setType (const ObstacleType & *value*)

setType of ObstacleType enum

Parameters

in	*value*	(reference ObstacleType)

Definition at line 19 of file obstacle.cpp.

7.19.4 Member Data Documentation

7.19.4.1 bool Obstacle::bInitialized [private]

represents whether the class is initialised

Definition at line 78 of file obstacle.h.

7.19.4.2 QTime Obstacle::cLastUpdate [private]

represents the last update time

Definition at line 79 of file obstacle.h.

7.19.4.3 Position Obstacle::cPosition [private]

coordinates in x,y, orientation in radian

Definition at line 77 of file obstacle.h.

7.19.4.4 ObstacleColor Obstacle::enumColor [private]

member variable of color enum

Definition at line 74 of file obstacle.h.

7.19.4.5 ObstacleStatus Obstacle::enumStatus [private]

member variable of status enum

Definition at line 76 of file obstacle.h.

7.19.4.6 ObstacleType Obstacle::enumType [private]

member variable of type enum

Definition at line 75 of file obstacle.h.

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

- obstacle.h
- obstacle.cpp

7.20 Orientation Class Reference

The Orientation class try to compute the position and the orientation of the robot due to distance values and angles from sensor data.

`#include <orientierung.h>`

Collaboration diagram for Orientation:

Static Public Member Functions

- static Position beginOrientation (QList< Position > &objects)

  beginOrientation will start the orientation phase and try to find known distances for localisation
- static double distancePolar (const double &angleA, const double &depthA, const double &angleB, const double &depthB)

  distancePolar
- static double distancePolar (const double &depthA, const double &depthB, const double &angleBetweenAB)

  distancePolar
- static double angleBetweenAB (const double &depthA, const double &depthB, const double &distC)

  angleBetweenAB
- static Position getGlobalPolarPosition (const Position &relativePosition, const Position &robotPosition)

  getGlobalPolarPosition
- static bool checkObjectsOnLine (const Position &erstesObjekt, const Position &zweitesObjekt, const Position &drittesObjekt)

  checkObjectsOnLine

Private Member Functions

- Orientation ()

  Orientation hidden constructor.
- ~Orientation ()

  ~Orientation hidden destructor
- Orientation (const Orientation &)

  Orientation as hidden copy constructor.
- Orientation & operator= (const Orientation &)

  operator = we leave just the declarations, so the compiler will warn us if we try to use those two functions by accident

Static Private Member Functions

- static Position getGlobalPosition (const Position &poleA, const Position &poleB, const Position &poleC)

  getGlobalPosition

7.20.1 Detailed Description

The Orientation class try to compute the position and the orientation of the robot due to distance values and angles from sensor data.

Definition at line 12 of file orientierung.h.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 Orientation::Orientation ( ) [private]

Orientation hidden constructor.

7.20.2.2 Orientation::~Orientation ( ) [private]

~Orientation hidden destructor

7.20.2.3 Orientation::Orientation (const Orientation &) [private]

Orientation as hidden copy constructor.

7.20.3 Member Function Documentation

7.20.3.1 double Orientation::angleBetweenAB (const double & *depthA*, const double & *depthB*, const double & *distC*) [static]

angleBetweenAB

Parameters

in	*depthA*	(double&)
in	*depthB*	(double&)
in	*distC*	(double&)

Returns

the angle between the distance a and distance b

Definition at line 187 of file orientierung.cpp.

7.20.3.2 Position Orientation::beginOrientation (QList< Position > & *objects*) [static]

beginOrientation will start the orientation phase and try to find known distances for localisation

Parameters

in	*QList<Position>*	&objects

Returns

a position with a certain degree of ensureance.

Definition at line 45 of file orientierung.cpp.

7.20.3.3 bool Orientation::checkObjectsOnLine (const Position & *erstesObjekt*, const Position & *zweitesObjekt*, const Position & *drittesObjekt*) [static]

checkObjectsOnLine

Parameters

in	*erstesObjekt*	(Position&)
in	*zweitesObjekt*	(Position&)
in	*drittesObjekt*	(Position&)

Returns

true if the three objects are on a straight line within a small error

Definition at line 14 of file orientierung.cpp.

7.20.3.4 double Orientation::distancePolar (const double & *angleA*, const double & *depthA*, const double & *angleB*, const double & *depthB*) [static]

distancePolar

Parameters

in	*depthA*	(double&)
in	*angleA*	(double&)
in	*depthB*	(double&)
in	*angleB*	(double&)

Returns

the distance between two polar coordinates

Definition at line 167 of file orientierung.cpp.

7.20.3.5 double Orientation::distancePolar (const double & *depthA*, const double & *depthB*, const double & *angleBetweenAB*) [static]

distancePolar

Parameters

in	*depthA*	(double&)
in	*depthB*	(double&)
in	*angleBetweenA-B*	(double&)

Returns

the distance between two polar coordinates

Definition at line 178 of file orientierung.cpp.

7.20.3.6 Position Orientation::getGlobalPolarPosition (const Position & *relativePosition*, const Position & *robotPosition*) [static]

getGlobalPolarPosition

Parameters

in	*relativePosition*	(realDepth, realAngle, radius, sizeType)
in	*robotPosition*	(x, y, rot)

Returns

a Position value which is transformed from the robot perspective to the global coordinate system

Definition at line 195 of file orientierung.cpp.

7.20.3.7 Position Orientation::getGlobalPosition (const Position & *poleA*, const Position & *poleB*, const Position & *poleC*) [static], [private]

getGlobalPosition

Parameters

poleA	Position Pole 1 (depth (m), angle (rad))
poleB	Position Pole 2 (depth (m), angle (rad))
poleC	Position Pole 3 (depth (m), angle (rad))

Returns

Roboterposition

Definition at line 123 of file orientierung.cpp.

7.20.3.8 Orientation& Orientation::operator= (const Orientation &) [private]

operator = we leave just the declarations, so the compiler will warn us if we try to use those two functions by accident

Returns

assigned value

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

- orientierung.h
- orientierung.cpp

7.21 PathPlanning Class Reference

`#include <pathplanning.h>`

Collaboration diagram for PathPlanning:

Classes

- class GridPoint

Public Slots

- void planPath ()

  planPath this method is called in the robot-thread and will plan the new path

Signals

- void sendUpdatedWaypoints (QList< QPair< double, double > > waypoints)

  sendUpdatedWaypoints: This signal will emit the newest generated waypoints to the pathrealizer
- void pathDisplay (PathPlotData dataPacket)

  pathDisplay, signal for displaying the path planning data in GUI-tab

Public Member Functions

- PathPlanning ()

  PathPlanning constructor of wavefront pathplanning.
- ~PathPlanning ()

Static Public Member Functions

- static const QPair< double, double > grid2XY (const int a, const int b, const double gridRotation, const double gridSpacing)

  grid2XY calculate real-world X,Y coordinates for any given grid rotation and spacing
- static const QPair< double, double > grid2AB (const double x, const double y, const double gridRotation, const double gridSpacing)

  grid2AB // calculate grid coordinates (A,B) from XY for any given grid rotation and spacing
- static bool getEnabled ()

  getEnabled return if the pathplanning is enabled
- static void setEnabled (const bool &value)

  setEnabled: set if the pathplanning should be enabled
- static bool getAvoidRestOfField ()

  getAvoidRestOfField determines if points outside the field should be ignord.
- static void setAvoidRestOfField (const bool &value)

  setAvoidRestOfField simple setter of avoidRestOfField-member
- static bool getIgnorePucks ()
- static void setIgnorePucks (const bool &value)

Static Public Attributes

- static std::atomic_bool streamPathEnabled

  streamPathEnabled determines, if the stream in GUI is enabled.

Private Member Functions

- void generateGrid ()

  generateGrid will generate a new grid with updated obstacle distances
- void calculatePathCosts ()

  calculatePathCosts calculate the path costs as sum of intrinsic and extrinsic costs.
- QPair< QList< QPair< double, double > >, QList< QPair < double, double > > > calculateWaypoints ()

  calculateWaypoints will calculate the new waypoints from path
- const double getGridRotation ()

  getGridRotation
- int gridIndex (int a, int b)

  gridIndex returns the 2D-mesh index by given a and b index

- const QPair< double, double > grid2XY (const int a, const int b)

  grid2XY converte a given a,b-grid to xy-grid
- const QPair< double, double > grid2AB (const double x, const double y)

  grid2AB converte a given xy-grid to a,b-grid

Static Private Member Functions

- static void initGridPoint (GridPoint &point)

  initGridPoint // does the initialization for any grid point, e.g. intrinsic cost calculation

Private Attributes

- Obstacle robot
- double robotX
- double robotY
- double robotRot
- int robotA
- int robotB
- double targetX
- double targetY
- double targetRot
- int targetA
- int targetB
- double gridRotation
- double gridSpacing
- int gridSizeA
- int gridSizeB
- double minA
- double maxA
- double minB
- double maxB
- QVector< QVector< GridPoint > > grid
- QElapsedTimer * timer
- QList< Obstacle > obstaclesPuck
- QList< Obstacle > obstaclesPole
- QList< Obstacle > obstaclesEnemy

Static Private Attributes

- static std::atomic_bool enabled
- static std::atomic_bool avoidRestOfField
- static std::atomic_bool ignorePucks

7.21.1 Detailed Description

Definition at line 14 of file pathplanning.h.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 PathPlanning::PathPlanning ( )

PathPlanning constructor of wavefront pathplanning.

PathPlanning::PathPlanning Constructor.

Definition at line 33 of file pathplanning.cpp.

7.21.2.2 PathPlanning::~PathPlanning ( )

Definition at line 39 of file pathplanning.cpp.

7.21.3 Member Function Documentation

7.21.3.1 void PathPlanning::calculatePathCosts ( ) [private]

calculatePathCosts calculate the path costs as sum of intrinsic and extrinsic costs.

Definition at line 279 of file pathplanning.cpp.

7.21.3.2 QPair< QList< QPair< double, double > >, QList< QPair< double, double > > > PathPlanning::calculateWaypoints ( ) [private]

calculateWaypoints will calculate the new waypoints from path

Returns

a qpair of grid- and waypoints

Definition at line 322 of file pathplanning.cpp.

7.21.3.3 void PathPlanning::generateGrid ( ) [private]

generateGrid will generate a new grid with updated obstacle distances

Todo : Maybe remove a puck from the list if it is the target?

Definition at line 171 of file pathplanning.cpp.

7.21.3.4 bool PathPlanning::getAvoidRestOfField ( ) [static]

getAvoidRestOfField determines if points outside the field should be ignord.

Returns

bool

Definition at line 392 of file pathplanning.cpp.

7.21.3.5 bool PathPlanning::getEnabled ( ) [static]

getEnabled return if the pathplanning is enabled

Returns

if is enabled

Definition at line 413 of file pathplanning.cpp.

7.21.3.6 const double PathPlanning::getGridRotation ( ) [private]

getGridRotation

Returns

the grid rotation

Definition at line 270 of file pathplanning.cpp.

7.21.3.7 bool PathPlanning::getIgnorePucks ( ) [static]

Definition at line 402 of file pathplanning.cpp.

7.21.3.8 const QPair< double, double > PathPlanning::grid2AB (const double *x*, const double *y*, const double *gridRotation*, const double *gridSpacing*) [static]

grid2AB // calculate grid coordinates (A,B) from XY for any given grid rotation and spacing

Parameters

x	(const double):component in carthesian space
y	(const double):component in carthesian space
gridRotation	(const double): rotation of the grid
gridSpacing	(const double): space among mesh cells.

Returns

the grid in a;b-components

Definition at line 379 of file pathplanning.cpp.

7.21.3.9 const QPair< double, double > PathPlanning::grid2AB (const double *x*, const double *y*) [private]

grid2AB converte a given xy-grid to a,b-grid

Parameters

a	
b	

Returns

a qpair(double,double) of the converted xy grid

Definition at line 366 of file pathplanning.cpp.

7.21.3.10 const QPair< double, double > PathPlanning::grid2XY (const int *a,* const int *b,* const double *gridRotation,* const double *gridSpacing*) [static]

grid2XY calculate real-world X,Y coordinates for any given grid rotation and spacing

Parameters

a	(const int): component in a-space
b	(const int): component in a-space
gridRotation	(const double): rotation of the given grid
gridSpacing,:	(const double) distance between mesh-cells

Returns

the transformed grid.

Definition at line 371 of file pathplanning.cpp.

7.21.3.11 const QPair< double, double > PathPlanning::grid2XY (const int *a,* const int *b*) [private]

grid2XY converte a given a,b-grid to xy-grid

Parameters

a	
b	

Returns

a qpair(double,double) of the converted ab grid

Definition at line 361 of file pathplanning.cpp.

7.21.3.12 int PathPlanning::gridIndex (int *a,* int *b*) [private]

gridIndex returns the 2D-mesh index by given a and b index

Parameters

in	*a*	(int)
in	*b*	(int)

Returns

the index in 2D-Mesh of give a and b index

7.21.3.13 void PathPlanning::initGridPoint (PathPlanning::GridPoint & *point*) [static], [private]

initGridPoint // does the initialization for any grid point, e.g. intrinsic cost calculation

Parameters

point	(reference of a grid-point)

Definition at line 386 of file pathplanning.cpp.

7.21.3.14 void PathPlanning::pathDisplay (PathPlotData *dataPacket*) [signal]

pathDisplay, signal for displaying the path planning data in GUI-tab

Parameters

in	*dataPacket*	(PathPlotData)

7.21.3.15 void PathPlanning::planPath ( ) [slot]

planPath this method is called in the robot-thread and will plan the new path

Definition at line 49 of file pathplanning.cpp.

7.21.3.16 void PathPlanning::sendUpdatedWaypoints (QList< QPair< double, double > > *waypoints*) [signal]

sendUpdatedWaypoints: This signal will emit the newest generated waypoints to the pathrealizer

Parameters

out	*waypoints*	(QList of QPair(double,double)): list with the generated waypoints

7.21.3.17 void PathPlanning::setAvoidRestOfField (const bool & *value*) [static]

setAvoidRestOfField simple setter of avoidRestOfField-member

Parameters

in	*value*	(const bool)

Definition at line 397 of file pathplanning.cpp.

7.21.3.18 void PathPlanning::setEnabled (const bool & *value*) [static]

setEnabled: set if the pathplanning should be enabled

Parameters

in	*value*	(bool)

Definition at line 418 of file pathplanning.cpp.

7.21.3.19 void PathPlanning::setIgnorePucks (const bool & *value*) [static]

Definition at line 407 of file pathplanning.cpp.

7.21.4 Member Data Documentation

7.21.4.1 std::atomic_bool PathPlanning::avoidRestOfField [static], [private]

shall the outer side of the field be avoided.

Definition at line 123 of file pathplanning.h.

7.21.4.2 std::atomic_bool PathPlanning::enabled [static],[private]

should the pathplanning algorithm be enabled

Definition at line 122 of file pathplanning.h.

7.21.4.3 QVector<QVector<GridPoint> > PathPlanning::grid [private]

grid for calculate the path

Definition at line 116 of file pathplanning.h.

7.21.4.4 double PathPlanning::gridRotation [private]

current grid rotation

Definition at line 111 of file pathplanning.h.

7.21.4.5 int PathPlanning::gridSizeA [private]

number of cells in a-direction

Definition at line 113 of file pathplanning.h.

7.21.4.6 int PathPlanning::gridSizeB [private]

number of cells in b-direction

Definition at line 114 of file pathplanning.h.

7.21.4.7 double PathPlanning::gridSpacing [private]

current spacing amoung mesh-cells

Definition at line 112 of file pathplanning.h.

7.21.4.8 std::atomic_bool PathPlanning::ignorePucks [static],[private]

should pucks be ignored when planning a path?

Definition at line 124 of file pathplanning.h.

7.21.4.9 double PathPlanning::maxA [private]

Definition at line 115 of file pathplanning.h.

7.21.4.10 double PathPlanning::maxB [private]

Definition at line 115 of file pathplanning.h.

7.21.4.11 double PathPlanning::minA [private]

Definition at line 115 of file pathplanning.h.

7.21.4.12 double PathPlanning::minB [private]

Definition at line 115 of file pathplanning.h.

7.21.4.13 QList<Obstacle> PathPlanning::obstaclesEnemy [private]

list of foes -> handeled as obstacle

Definition at line 120 of file pathplanning.h.

7.21.4.14 QList<Obstacle> PathPlanning::obstaclesPole [private]

list of pole-> handeled as obstacle

Definition at line 119 of file pathplanning.h.

7.21.4.15 QList<Obstacle> PathPlanning::obstaclesPuck [private]

list of puck -> handeled as obstacle

Definition at line 118 of file pathplanning.h.

7.21.4.16 Obstacle PathPlanning::robot [private]

own robot obstacle

Definition at line 98 of file pathplanning.h.

7.21.4.17 int PathPlanning::robotA [private]

a pos of robot->transformed

Definition at line 104 of file pathplanning.h.

7.21.4.18 int PathPlanning::robotB [private]

b pos of robot->transformed

Definition at line 105 of file pathplanning.h.

7.21.4.19 double PathPlanning::robotRot [private]

robots orientation in rad

Definition at line 103 of file pathplanning.h.

7.21.4.20 double PathPlanning::robotX [private]

x pos in m of robot

Definition at line 101 of file pathplanning.h.

7.21.4.21 double PathPlanning::robotY [private]

y pos in m of robot

Definition at line 102 of file pathplanning.h.

7.21.4.22 std::atomic_bool PathPlanning::streamPathEnabled [static]

streamPathEnabled determines, if the stream in GUI is enabled.

Definition at line 48 of file pathplanning.h.

7.21.4.23 int PathPlanning::targetA [private]

target position in a

Definition at line 109 of file pathplanning.h.

7.21.4.24 int PathPlanning::targetB [private]

target position in b

Definition at line 110 of file pathplanning.h.

7.21.4.25 double PathPlanning::targetRot [private]

target orientation in rad

Definition at line 108 of file pathplanning.h.

7.21.4.26 double PathPlanning::targetX [private]

target position in x in m

Definition at line 106 of file pathplanning.h.

7.21.4.27 double PathPlanning::targetY [private]

target position in x in m

Definition at line 107 of file pathplanning.h.

7.21.4.28 QElapsedTimer* PathPlanning::timer [private]

timer, how long the algorithm needed for path calculation

Definition at line 117 of file pathplanning.h.

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

- pathplanning.h
- pathplanning.cpp

7.22 PathPlotData Struct Reference

The PathPlotData struct is the data holder for plotting the path.

`#include <pathplotdata.h>`

Collaboration diagram for PathPlotData:

Classes

- struct Point

  The Point struct represent Waypoints data coming from PathPlanning.

Public Types

- enum DataTypeEnum { WAYPOINTS, SPLINE, WAYPOINTS, SPLINE }

  The DataTypeEnum enum for waypoints or spline.
- enum DataTypeEnum { WAYPOINTS, SPLINE, WAYPOINTS, SPLINE }

  The DataTypeEnum enum for waypoints or spline, enum showing the source of the data.

Public Member Functions

- PathPlotData ()

  PathPlotData is Default constructor (empty vectors etc).
- PathPlotData ()

  PathPlotData is Default constructor (empty vectors etc).

Public Attributes

- enum PathPlotData::DataTypeEnum dataType
- QVector< Point > data
- int dataSizeX
- int dataSizeY
- QList< QPair< double, double > > waypoints
- QPair< double, double > robot
- QPair< double, double > target
- QVector< double > splineX
- QVector< double > splineY
- double splineLength

7.22.1 Detailed Description

The PathPlotData struct is the data holder for plotting the path.

Definition at line 11 of file Plots/pathplotdata.h.

7.22.2 Member Enumeration Documentation

7.22.2.1 enum PathPlotData::DataTypeEnum

The DataTypeEnum enum for waypoints or spline, enum showing the source of the data.

Enumerator:

WAYPOINTS

SPLINE

WAYPOINTS

SPLINE

Definition at line 16 of file Structs/pathplotdata.h.

7.22.2.2 enum PathPlotData::DataTypeEnum

The DataTypeEnum enum for waypoints or spline.

Enumerator:

WAYPOINTS

SPLINE

WAYPOINTS

SPLINE

Definition at line 17 of file Plots/pathplotdata.h.

7.22.3 Constructor & Destructor Documentation

7.22.3.1 PathPlotData::PathPlotData ( ) [inline]

PathPlotData is Default constructor (empty vectors etc).

Definition at line 25 of file Plots/pathplotdata.h.

7.22.3.2 PathPlotData::PathPlotData ( ) [inline]

PathPlotData is Default constructor (empty vectors etc).

Definition at line 23 of file Structs/pathplotdata.h.

7.22.4 Member Data Documentation

7.22.4.1 QVector< Point > PathPlotData::data

grid points containing the travel cost values

Definition at line 44 of file Plots/pathplotdata.h.

7.22.4.2 int PathPlotData::dataSizeX

size in x direction

Definition at line 45 of file Plots/pathplotdata.h.

7.22.4.3 int PathPlotData::dataSizeY

size in y direction

Definition at line 45 of file Plots/pathplotdata.h.

7.22.4.4 enum PathPlotData::DataTypeEnum PathPlotData::dataType

7.22.4.5 QPair< double, double > PathPlotData::robot

position of the robot (at path planning start)

Definition at line 49 of file Plots/pathplotdata.h.

7.22.4.6 double PathPlotData::splineLength

length of the spline QVectors

Definition at line 55 of file Plots/pathplotdata.h.

7.22.4.7 QVector< double > PathPlotData::splineX

spline in x direction

Definition at line 53 of file Plots/pathplotdata.h.

7.22.4.8 QVector< double > PathPlotData::splineY

spline in y direction

Definition at line 53 of file Plots/pathplotdata.h.

7.22.4.9 QPair< double, double > PathPlotData::target

target position (at path planning start)

Definition at line 50 of file Plots/pathplotdata.h.

7.22.4.10 QList< QPair< double, double > > PathPlotData::waypoints

caluclated waypoints in X/Y space

Definition at line 48 of file Plots/pathplotdata.h.

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

- Plots/pathplotdata.h
- Structs/pathplotdata.h

7.23 PathRealizer Class Reference

The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

`#include <pathRealizer.h>`

Collaboration diagram for PathRealizer:

Public Types

- enum StatePathProcessing { STOP, RUNNING }

  The StatePathProcessing enum internal state-machine.

Public Slots

- void slotUpdateWaypoints (QList< QPair< double, double > > waypoints)

  slotUpdateWaypoints receive computed waypoints vom AI/pathplanning. Will hard reset the waypoints if new waypoints are available.
- void slotMotionControl ()

  slotMotionControl => This method represents the internal state-machine implementing a PID-controller for motion control.
- void slotChangePIDParams (PIDParams p)

  slotChangePIDParams => This method will change the PID-values caused by changing the values in the GUI.

Signals

- void signalSendRobotControlParams (double velocity, double turnangle)

  signalSendRobotControlParams => This method will emit the new velocity and turnrate to the actorLowLevel class.
- void signalSplinePlot (PathPlotData pathPlotData)

  signalSplinePlot this method will emit a data-struct, to display the path in GUI
- void signalPIDPlot (PIDPlotData d)

  signalPIDPlot this method will emit a data-struct to display the current PID-values

Public Member Functions

- PathRealizer ()

  PathRealizer => default constructor intialising member variables with default values.
- ~PathRealizer ()

  ~PathRealizer => default destructor which will clear the heap and delete other objects

Static Public Attributes

- static std::atomic_bool streamPIDEnabled

Private Slots

- void slotTimerSendPIDPlot ()

  slotTimerSendPIDPlot this method will update the data in GUI.

Private Member Functions

- QVector< double > lowPass (QVector< double > in, double alpha=0)

  lowPass => This method will lowpass filter the given data
- QVector< double > getVelocityProfile ()

  getVelocityProfile this method will return the internal velocity profile(weaviness) of a spline.
- QVector< double > splineToQVector (tkqt::spline spline, QVector< double > metric)

  splineToQVector => This method will convert an tkqt::spline Obj into an QVector
- double getDistance (QPair< double, double > a, QPair< double, double > b)

  getDistance => This function will calculate the distance between two points given as QPairs.

Static Private Member Functions

- static const double constrainAngle (const double inRad)

  constrainAngle => This method is responsible to prohibit a phase-shift.
- static QVector< double > takeDimension (const QVector< QPair< double, double > > in, const int dimension)

  takeDimension => This method will reduce the dimensions of a given QVector of points.

Private Attributes

- StatePathProcessing state
- Obstacle robotObstacle
- QList< QPair< double, double > > internalWP
- int numWP
- tkqt::spline splineX
- tkqt::spline splineY
- tkqt::spline velProfile
- QTimer timerMotionControl
- QMutex pidHistMutex
- QTimer timerPIDPlot
- qint64 timeOfStart
- QList< double > pidHistTime
- QList< double > pidHistWinkelSoll
- QList< double > pidHistWinkelIst
- QList< double > pidHistDistIst
- QList< double > pidHistDistSoll
- double splineProgress
- double maxWaviness
- double integrationTime
- QElapsedTimer * elapsedTime
- double periodMotionControl
- double PID_A_P
- double PID_A_I
- double PID_A_D
- double PID_V_P
- double PID_V_I
- double PID_V_D
- double lastDeltaA
- double iDeltaA
- double lastDeltaL
- double iDeltaL

7.23.1 Detailed Description

The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

Definition at line 23 of file pathRealizer.h.

7.23.2 Member Enumeration Documentation

7.23.2.1 enum PathRealizer::StatePathProcessing

The StatePathProcessing enum internal state-machine.

Enumerator:

STOP first enum value, represents the stop state

RUNNING second enum value, represents the running and therefore pathrealising state

Definition at line 43 of file pathRealizer.h.

7.23.3 Constructor & Destructor Documentation

7.23.3.1 PathRealizer::PathRealizer ( )

PathRealizer => default constructor intialising member variables with default values.

7.23.3.2 PathRealizer::~PathRealizer ( )

~PathRealizer => default destructor which will clear the heap and delete other objects

7.23.4 Member Function Documentation

7.23.4.1 static const double PathRealizer::constrainAngle (const double *inRad*) [inline], [static], [private]

constrainAngle => This method is responsible to prohibit a phase-shift.

Parameters

in	*inRad*	(double) angle in rad which should be checked

Returns

the corrected angle in rad as double

Definition at line 175 of file pathRealizer.h.

7.23.4.2 double PathRealizer::getDistance (QPair< double, double > *a*, QPair< double, double > *b*) [inline], [private]

getDistance => This function will calculate the distance between two points given as QPairs.

Parameters

a	(QPair<double,double>) first point in x and y
b	(QPair<double,double>) second point in x and y

Returns

the distance between a and b

Definition at line 163 of file pathRealizer.h.

7.23.4.3 QVector<double> PathRealizer::getVelocityProfile ( ) [private]

getVelocityProfile this method will return the internal velocity profile(weaviness) of a spline.

Returns

QVector<double> the velocity profile of a spline

7.23.4.4 QVector<double> PathRealizer::lowPass (QVector< double > *in*, double *alpha* = 0) [private]

lowPass => This method will lowpass filter the given data

Parameters

in	*in*	(QVector<double>) vector of values
in	*alpha*	(double) weighting of previous values during iteration

Returns

QVector<double> as filtered values

7.23.4.5 void PathRealizer::signalPIDPlot (PIDPlotData *d*) [signal]

signalPIDPlot this method will emit a data-struct to display the current PID-values

Parameters

in	*d*	(struct of QList<double>) an struct with data for the GUI plot of PID-values.

7.23.4.6 void PathRealizer::signalSendRobotControlParams (double *velocity*, double *turnangle*) [signal]

signalSendRobotControlParams => This method will emit the new velocity and turnrate to the actorLowLevel class.

Parameters

in	*velocity*	(double) m/s
in	*turnangle*	(double) rad/s

7.23.4.7 void PathRealizer::signalSplinePlot (PathPlotData *pathPlotData*) [signal]

signalSplinePlot this method will emit a data-struct, to display the path in GUI

Parameters

in	*pathPlotData*	(PathPlotData) an struct which included the values of the GUI tab to display the path.

7.23.4.8 void PathRealizer::slotChangePIDParams (PIDParams *p*) [slot]

slotChangePIDParams => This method will change the PID-values caused by changing the values in the GUI.

Parameters

in	*p*	(struct of double values) for changing the PID-controler values.

7.23.4.9 void PathRealizer::slotMotionControl ( ) [slot]

slotMotionControl => This method represents the internal state-machine implementing a PID-controller for motion control.

7.23.4.10 void PathRealizer::slotTimerSendPIDPlot ( ) [private], [slot]

slotTimerSendPIDPlot this method will update the data in GUI.

7.23.4.11 void PathRealizer::slotUpdateWaypoints (QList< QPair< double, double > > *waypoints*) [slot]

slotUpdateWaypoints receive computed waypoints vom AI/pathplanning. Will hard reset the waypoints if new waypoints are available.

Parameters

in	*waypoints*	(QList of QPair of <double,double>) representing the given waypoints in (x,y).

7.23.4.12 QVector<double> PathRealizer::splineToQVector (tkqt::spline *spline*, QVector< double > *metric*) [private]

splineToQVector => This method will convert an tkqt::spline Obj into an QVector

Parameters

in	*spline*	(tkqt::spline) spline based on a metric
in	*metric*	(QVector<double>) as distance measurement of the spline

Returns

QVector<double> the converted spline

7.23.4.13 static QVector<double> PathRealizer::takeDimension (const QVector< QPair< double, double > > *in*, const int *dimension*) [static], [private]

takeDimension => This method will reduce the dimensions of a given QVector of points.

Parameters

in	*in*	(QVector<QPair<double,double>> multi-dimensional array of points
in	*dimension*	(int) the desired dimension which should returned.

Returns

an empty QVector<double> if length of QVector is equal to zero, otherwise the flatten (one dimensional) QVector<double>

7.23.5 Member Data Documentation

7.23.5.1 QElapsedTimer* PathRealizer::elapsedTime [private]

member object for determination of new time delta

Definition at line 120 of file pathRealizer.h.

7.23.5.2 double PathRealizer::iDeltaA [private]

desired angle derivation

Definition at line 131 of file pathRealizer.h.

7.23.5.3 double PathRealizer::iDeltaL [private]

desiered length derivation

Definition at line 133 of file pathRealizer.h.

7.23.5.4 double PathRealizer::integrationTime [private]

time integration

Definition at line 119 of file pathRealizer.h.

7.23.5.5 QList< QPair<double,double> > PathRealizer::internalWP [private]

internal member-variable to save the received waypoints

Definition at line 98 of file pathRealizer.h.

7.23.5.6 double PathRealizer::lastDeltaA [private]

previous desired angle derivation

Definition at line 130 of file pathRealizer.h.

7.23.5.7 double PathRealizer::lastDeltaL [private]

pervious desired length derivation

Definition at line 132 of file pathRealizer.h.

7.23.5.8 double PathRealizer::maxWaviness [private]

maximal value of waviness

Definition at line 118 of file pathRealizer.h.

7.23.5.9 int PathRealizer::numWP [private]

length of the internal waypoints

Definition at line 99 of file pathRealizer.h.

7.23.5.10 double PathRealizer::periodMotionControl [private]

Time of one period => internal heartbeat

Definition at line 122 of file pathRealizer.h.

7.23.5.11 double PathRealizer::PID_A_D [private]

D-part of the angle-PID

Definition at line 125 of file pathRealizer.h.

7.23.5.12 double PathRealizer::PID_A_I [private]

I-part of the angle-PID

Definition at line 124 of file pathRealizer.h.

7.23.5.13 double PathRealizer::PID_A_P [private]

P-part of the angle-PID

Definition at line 123 of file pathRealizer.h.

7.23.5.14 double PathRealizer::PID_V_D [private]

D-part of the velocity-PID

Definition at line 128 of file pathRealizer.h.

7.23.5.15 double PathRealizer::PID_V_I [private]

I-part of the velocity-PID

Definition at line 127 of file pathRealizer.h.

7.23.5.16 double PathRealizer::PID_V_P [private]

P-part of the velocity-PID

Definition at line 126 of file pathRealizer.h.

7.23.5.17 QList<double> PathRealizer::pidHistDistIst [private]

Display the past n-secounds in GUI plot

Definition at line 111 of file pathRealizer.h.

7.23.5.18 QList<double> PathRealizer::pidHistDistSoll [private]

Display the past n-secounds in GUI plot

Definition at line 111 of file pathRealizer.h.

7.23.5.19 QMutex PathRealizer::pidHistMutex [private]

Mutex to prohibit racing condition

Definition at line 108 of file pathRealizer.h.

7.23.5.20 QList<double> PathRealizer::pidHistTime [private]

Display the past n-secounds in GUI plot

Definition at line 111 of file pathRealizer.h.

7.23.5.21 QList<double> PathRealizer::pidHistWinkelIst [private]

Display the past n-secounds in GUI plot

Definition at line 111 of file pathRealizer.h.

7.23.5.22 QList<double> PathRealizer::pidHistWinkelSoll [private]

Display the past n-secounds in GUI plot

Definition at line 111 of file pathRealizer.h.

7.23.5.23 Obstacle PathRealizer::robotObstacle [private]

current pose of the robot including, position in x, in y and orientation

Definition at line 96 of file pathRealizer.h.

7.23.5.24 double PathRealizer::splineProgress [private]

current position on spline

Definition at line 117 of file pathRealizer.h.

7.23.5.25 tkqt::spline PathRealizer::splineX [private]

cubic hermite spline created of received waypoints in x direction

Definition at line 101 of file pathRealizer.h.

7.23.5.26 tkqt::spline PathRealizer::splineY [private]

cubic hermite spline created of received waypoints in y direction

Definition at line 102 of file pathRealizer.h.

7.23.5.27 StatePathProcessing PathRealizer::state [private]

internal state-machine

Definition at line 95 of file pathRealizer.h.

7.23.5.28 std::atomic_bool PathRealizer::streamPIDEnabled [static]

Definition at line 37 of file pathRealizer.h.

7.23.5.29 qint64 PathRealizer::timeOfStart [private]

ms since epoch, used to display time since program start in pid plot

Definition at line 110 of file pathRealizer.h.

7.23.5.30 QTimer PathRealizer::timerMotionControl [private]

internal heartbeat to call the controler

Definition at line 105 of file pathRealizer.h.

7.23.5.31 QTimer PathRealizer::timerPIDPlot [private]

update timer for the PID-plot

Definition at line 109 of file pathRealizer.h.

7.23.5.32 tkqt::spline PathRealizer::velProfile [private]

from spline derived velocity profile caused by the waviness of the resulting spline

Definition at line 103 of file pathRealizer.h.

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

- pathRealizer.h

7.24 PIDParams Struct Reference

The PIDParams struct, values for the PID controler for angular PID and velocity PID.

`#include <pidparams.h>`

Collaboration diagram for PIDParams:

Public Attributes

- double PID_A_P
- double PID_A_I
- double PID_A_D
- double PID_V_P
- double PID_V_I
- double PID_V_D

7.24.1 Detailed Description

The PIDParams struct, values for the PID controler for angular PID and velocity PID.

Definition at line 9 of file Actor/pidparams.h.

7.24.2 Member Data Documentation

7.24.2.1 double PIDParams::PID_A_D

D of angular PID

Definition at line 13 of file Actor/pidparams.h.

7.24.2.2 double PIDParams::PID_A_I

I of angular PID

Definition at line 12 of file Actor/pidparams.h.

7.24.2.3 double PIDParams::PID_A_P

P of angular PID

Definition at line 11 of file Actor/pidparams.h.

7.24.2.4 double PIDParams::PID_V_D

D of velocity PID

Definition at line 16 of file Actor/pidparams.h.

7.24.2.5 double PIDParams::PID_V_I

I of velocity PID

Definition at line 15 of file Actor/pidparams.h.

7.24.2.6 double PIDParams::PID_V_P

P of velocity PID

Definition at line 14 of file Actor/pidparams.h.

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

- Actor/pidparams.h
- Structs/pidparams.h

7.25 PIDPlotData Struct Reference

The PIDPlotData struct represents the data of the PID controler of the last n-time steps.

`#include <pidplotdata.h>`

Collaboration diagram for PIDPlotData:

Public Attributes

- QList< double > time
- QList< double > winkelSoll
- QList< double > winkelIst

- QList< double > distanzSoll
- QList< double > distanzIst

7.25.1 Detailed Description

The PIDPlotData struct represents the data of the PID controler of the last n-time steps.

Definition at line 10 of file Plots/pidplotdata.h.

7.25.2 Member Data Documentation

7.25.2.1 QList< double > PIDPlotData::distanzIst

list of current distances

Definition at line 16 of file Plots/pidplotdata.h.

7.25.2.2 QList< double > PIDPlotData::distanzSoll

list of reference distances

Definition at line 15 of file Plots/pidplotdata.h.

7.25.2.3 QList< double > PIDPlotData::time

list of time

Definition at line 12 of file Plots/pidplotdata.h.

7.25.2.4 QList< double > PIDPlotData::winkelIst

list of current angles

Definition at line 14 of file Plots/pidplotdata.h.

7.25.2.5 QList< double > PIDPlotData::winkelSoll

list of reference angles

Definition at line 13 of file Plots/pidplotdata.h.

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

- Plots/pidplotdata.h
- Structs/pidplotdata.h

7.26 PlayerX Class Reference

The Player class this class contains the instance of the player client to access in 'global' scope.

`#include <player.h>`

Collaboration diagram for PlayerX:

Static Public Member Functions

- static void startPlayer ()

  startPlayer

- static void stopPlayerIfStarted ()

  stopPlayerIfStarted

- static PlayerCc::PlayerClient * getInstance ()

  getInstance

Static Private Member Functions

- static std::string getSelfpath ()

Static Private Attributes

- static bool didWeStartPlayerOurselves

7.26.1 Detailed Description

The Player class this class contains the instance of the player client to access in 'global' scope.

Definition at line 14 of file player.h.

7.26.2 Member Function Documentation

7.26.2.1 PlayerCc::PlayerClient * PlayerX::getInstance ( ) [static]

getInstance

Returns

the static instance of a playerclient object on heap

Definition at line 112 of file player.cpp.

7.26.2.2 static std::string PlayerX::getSelfpath ( ) [static], [private]

7.26.2.3 void PlayerX::startPlayer ( ) [static]

startPlayer

Definition at line 37 of file player.cpp.

7.26.2.4 void PlayerX::stopPlayerIfStarted ( ) [static]

stopPlayerIfStarted

Definition at line 95 of file player.cpp.

7.26.3 Member Data Documentation

7.26.3.1 bool PlayerX::didWeStartPlayerOurselves [static],[private]

Definition at line 32 of file player.h.

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

- player.h
- player.cpp

7.27 PathPlotData::Point Struct Reference

The Point struct represent Waypoints data coming from PathPlanning.

`#include <pathplotdata.h>`

Collaboration diagram for PathPlotData::Point:

Public Attributes

- double x
- double y
- double value

7.27.1 Detailed Description

The Point struct represent Waypoints data coming from PathPlanning.

Definition at line 39 of file Plots/pathplotdata.h.

7.27.2 Member Data Documentation

7.27.2.1 double PathPlotData::Point::value

scalar potential field

Definition at line 40 of file Plots/pathplotdata.h.

7.27.2.2 double PathPlotData::Point::x

x value in m

Definition at line 40 of file Plots/pathplotdata.h.

7.27.2.3 double PathPlotData::Point::y

y value in m

Definition at line 40 of file Plots/pathplotdata.h.

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

- Plots/pathplotdata.h
- Structs/pathplotdata.h

7.28 Position Class Reference

The Position struct will represent the current pose of the robot.

`#include <position.h>`

Collaboration diagram for Position:

Public Member Functions

- double x () const
  x
- void x (double value)
  x
- double y () const
  y
- void y (double value)
  y
- double rot () const
  rot
- void rot (double value)
  rot
- SizeType sizeType () const
  sizeType
- void sizeType (SizeType value)
  sizeType
- double certainty () const
  certainty
- void setCertainty (double certainty)
  setCertainty
- Position ()
  Position Default constructor.
- Position (double x, double y)
  Position.
- Position (QPair< double, double > qPairPosition)
  Position.
- Position (double x, double y, double rot)
  Position.
- Position (double x, double y, double rot, double certainty)
  Position constructor.
- Position (double x, double y, double rot, SizeType size)
  Position.
- double getDistanceTo (const Position &b) const
  getDistanceTo
- bool isConsimilarTo (const Position &b, const double &tolerance) const
  isSimilarPosition
- bool isConsimilarTo (const Position &b) const
  isSimilarPosition
- bool operator== (const Position &b) const
  operator ==
- bool isPositionInStartField () const
  isPositionInStartField

Private Attributes

- double m_x
- double m_y
- double m_rot
- SizeType m_size
- double m_certainty

7.28.1 Detailed Description

The Position struct will represent the current pose of the robot.

Definition at line 20 of file position.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 Position::Position ( )

Position Default constructor.

Definition at line 55 of file position.cpp.

7.28.2.2 Position::Position (double *x*, double *y*)

Position.

Parameters

in	*x*	(double)
in	*y*	(double)

Definition at line 64 of file position.cpp.

7.28.2.3 Position::Position (QPair< double, double > *qPairPosition*)

Position.

Parameters

in	*qPairPosition*	(QPair<double,double>)

Definition at line 72 of file position.cpp.

7.28.2.4 Position::Position (double *x*, double *y*, double *rot*)

Position.

Parameters

in	*x*	(double)
in	*y*	(double)
in	*rot*	(double)

all values will be stored in a range from 0 to 2∗M_PI

Definition at line 78 of file position.cpp.

7.28.2.5 Position::Position (double *x*, double *y*, double *rot*, double *certainty*)

Position constructor.

Parameters

in	*x*	(double)
in	*y*	(double)
in	*rot*	(double)
in	*certainty*	(double)

Definition at line 88 of file position.cpp.

7.28.2.6 Position::Position (double *x*, double *y*, double *rot*, SizeType *size*)

Position.

Parameters

in	*x*	(double)
in	*y*	(double)
in	*rot*	(double)
in	*size*	(SizeType)

Definition at line 94 of file position.cpp.

7.28.3 Member Function Documentation

7.28.3.1 double Position::certainty ( ) const

certainty

Returns

Definition at line 5 of file position.cpp.

7.28.3.2 double Position::getDistanceTo (const Position & *b*) const

getDistanceTo

Parameters

in	*b*	(Position&)

Returns

kartesian distance between both positions

Definition at line 116 of file position.cpp.

7.28.3.3 bool Position::isConsimilarTo (const Position & *b*, const double & *tolerance*) const

isSimilarPosition

Parameters

in	*b*	(Position&)
in	*tolerance*	(double&)

Returns

true if the distance is in tolerance range

Definition at line 125 of file position.cpp.

7.28.3.4 bool Position::isConsimilarTo (const Position & *b*) const

isSimilarPosition

Parameters

in	*b*	(Position&)

Returns

true if the distance is in tolerance range

Definition at line 133 of file position.cpp.

7.28.3.5 bool Position::isPositionInStartField ( ) const

isPositionInStartField

Returns

true if the position is within the start area

Definition at line 106 of file position.cpp.

7.28.3.6 bool Position::operator== (const Position & *b*) const

operator ==

Parameters

in	*b*	(Position&)

Returns

Todo we may change this condition here

Definition at line 101 of file position.cpp.

7.28.3.7 double Position::rot ( ) const

rot

Returns

Definition at line 32 of file position.cpp.

7.28.3.8 void Position::rot (double *value*)

rot

Parameters

in	*value*	(double)

all values will be stored in a range from 0 to 2∗M_PI

Definition at line 37 of file position.cpp.

7.28.3.9 void Position::setCertainty (double *certainty*)

setCertainty

Parameters

in	*value*	(certainty)

Definition at line 10 of file position.cpp.

7.28.3.10 SizeType Position::sizeType ( ) const

sizeType

Returns

Definition at line 45 of file position.cpp.

7.28.3.11 void Position::sizeType (SizeType *value*)

sizeType

Parameters

in	*value*	(SizeType)

Definition at line 50 of file position.cpp.

7.28.3.12 double Position::x ( ) const

x

Returns

Definition at line 14 of file position.cpp.

7.28.3.13 void Position::x (double *value*)

x

Parameters

in	*value*	(double)

Definition at line 19 of file position.cpp.

7.28.3.14 double Position::y ( ) const

y

Returns

Definition at line 23 of file position.cpp.

7.28.3.15 void Position::y (double *value*)

y

Parameters

in	*value*	(double)

Definition at line 28 of file position.cpp.

7.28.4 Member Data Documentation

7.28.4.1 double Position::m_certainty [private]

how reliable a position is

Definition at line 27 of file position.h.

7.28.4.2 double Position::m_rot [private]

radian between 0 and 2∗PI

Definition at line 25 of file position.h.

7.28.4.3 SizeType Position::m_size [private]

how big an object at a position is

Definition at line 26 of file position.h.

7.28.4.4 double Position::m_x [private]

position in x in m

Definition at line 23 of file position.h.

7.28.4.5 double Position::m_y [private]

position in y in m

Definition at line 24 of file position.h.

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

- position.h
- position.cpp

7.29 Referee Class Reference

Die Schiedsrichterklasse.

`#include <referee.h>`

Collaboration diagram for Referee:

Signals

- void disconnected ()

  Wird gesendet falls die Verbindung getrennt wird.
- void detectionStart ()

  Wird gesendet, wenn die Zeit zur Spielfelderkennung beginnt.
- void gameStart ()

  Wird gesendet wenn der Wettkampf beginnt.
- void gameOver ()

  Wird gesendet wenn der Wettkampf auf Serverseite beendet wird.
- void abValues (double a, double b)

  Gibt euch die richtigen Seitenlängen.
- void trueColorOfTeam (TeamColor color)

  Gibt euch die richtige Teamfarbe.
- void stopMovement ()

  Roboter muss seine Bewegung sofort stoppen.
- void connected ()

  Wird gesendet, falls die Verbindung zum Server erfolgreich war.
- void connectFailed ()

  Wird gesendet, falls die Verbindung zum Server nicht aufgebaut werden konnte.

Public Member Functions

- Referee (int teamID, QObject ∗parent=0)

  Erstellt einen Schiedsrichter.
- ∼Referee ()

  Destruktor.
- void setVerbose (bool enabled)

  Gibt Meldungen auf der Konsole aus.
- bool isVerbose ()

  Fragt ab, ob meldungen auf der Konsole ausgegeben werden.
- void connectToServer (const QString &ip, int port)

  Verbindet zum Server Angelina.
- void reportReady ()

Gibt das Zeichen, dass ihr bereit seid.

- void reportDone ()

  Gibt das Zeichen, dass ihr Fertig seid bzw. aufhören möchtet (DONE).
- void sendAlive ()

  Ruft diese Funktion mindestens alle 45 Sekunden auf um Angelina mitzuteilen, dass ihr noch funktioniert ;-).
- void tellAbRatio (double ratio)

  Sendet das Seitenverhältnis an Angelina.
- void tellTeamColor (TeamColor color)

  Sendet die eigene Farbe.
- void reportGoal ()

  Teilt Angelina mit, dass ein Tor geschossen wurde.
- void tellEgoPos (double posX, double posY)

  Sendet die eigene Position an Angelina (optional).
- bool isConnected ()

  Returns true, wenn die Verbindung zum Server existiert.

Private Slots

- void slotRead ()
- void slotConnected ()
- void slotDisconnected ()

Private Attributes

- Hermes * messengerOfTheGods
- unsigned int wLimit
- int myTeamID
- int messageSize
- bool connection
- bool testMode
- bool verbose
- bool ready

7.29.1 Detailed Description

Die Schiedsrichterklasse.

Die Schiedsrichterklasse kommuniziert mit Angelina und dient dazu dem Server die Ergebnisse mitzuteilen

Definition at line 45 of file referee.h.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 Referee::Referee (int *teamID*, QObject * *parent* = 0)

Erstellt einen Schiedsrichter.

Parameters

teamID	Eure teamID (einfach die Nummer eurer CPP Gruppe).
**parent*	Das Eltern-Objekt.

Definition at line 25 of file referee.cpp.

7.29.2.2 Referee::∼Referee ( )

Destruktor.

Definition at line 34 of file referee.cpp.

7.29.3 Member Function Documentation

7.29.3.1 void Referee::abValues (double *a,* double *b*) [signal]

Gibt euch die richtigen Seitenlängen.

7.29.3.2 void Referee::connected ( ) [signal]

Wird gesendet, falls die Verbindung zum Server erfolgreich war.

7.29.3.3 void Referee::connectFailed ( ) [signal]

Wird gesendet, falls die Verbindung zum Server nicht aufgebaut werden konnte.

7.29.3.4 void Referee::connectToServer (const QString & *ip,* int *port*)

Verbindet zum Server Angelina.

Parameters

&ip	Die IP-Adresse z.B. 127.0.0.1 (localhost) zum Testen an einem Computer.
port	Der Port z.B. 10000.

Definition at line 52 of file referee.cpp.

7.29.3.5 void Referee::detectionStart ( ) [signal]

Wird gesendet, wenn die Zeit zur Spielfelderkennung beginnt.

7.29.3.6 void Referee::disconnected ( ) [signal]

Wird gesendet falls die Verbindung getrennt wird.

7.29.3.7 void Referee::gameOver ( ) [signal]

Wird gesendet wenn der Wettkampf auf Serverseite beendet wird.

7.29.3.8 void Referee::gameStart ( ) [signal]

Wird gesendet wenn der Wettkampf beginnt.

7.29.3.9 bool Referee::isConnected ( )

Returns true, wenn die Verbindung zum Server existiert.

Definition at line 333 of file referee.cpp.

7.29.3.10 bool Referee::isVerbose ( )

Fragt ab, ob meldungen auf der Konsole ausgegeben werden.

Definition at line 47 of file referee.cpp.

7.29.3.11 void Referee::reportDone ( )

Gibt das Zeichen, dass ihr Fertig seid bzw. aufhören möchtet (DONE).

Definition at line 98 of file referee.cpp.

7.29.3.12 void Referee::reportGoal ( )

Teilt Angelina mit, dass ein Tor geschossen wurde.

Definition at line 212 of file referee.cpp.

7.29.3.13 void Referee::reportReady ( )

Gibt das Zeichen, dass ihr bereit seid.

Definition at line 80 of file referee.cpp.

7.29.3.14 void Referee::sendAlive ( )

Ruft diese Funktion mindestens alle 45 Sekunden auf um Angelina mitzuteilen, dass ihr noch funktioniert ;-).

Definition at line 116 of file referee.cpp.

7.29.3.15 void Referee::setVerbose (bool *enabled*)

Gibt Meldungen auf der Konsole aus.

Parameters

enabled	Der Ein- Aus-Schalter.

Definition at line 42 of file referee.cpp.

7.29.3.16 void Referee::slotConnected ( ) [private], [slot]

Definition at line 316 of file referee.cpp.

7.29.3.17 void Referee::slotDisconnected ( ) [private], [slot]

Definition at line 323 of file referee.cpp.

7.29.3.18 void Referee::slotRead ( ) [private], [slot]

Definition at line 231 of file referee.cpp.

7.29.3.19 void Referee::stopMovement ( ) [signal]

Roboter muss seine Bewegung sofort stoppen.

7.29.3.20 void Referee::tellAbRatio (double *ratio*)

Sendet das Seitenverhältnis an Angelina.

Parameters

ratio	Ergebnis von a/b z.B 0.5 (a und b in Metern).

Definition at line 135 of file referee.cpp.

7.29.3.21 void Referee::tellEgoPos (double *posX*, double *posY*)

Sendet die eigene Position an Angelina (optional).

Parameters

posX	Position in a-Richtung in Metern vom linken oberen Spielfeldrand
posY	Position in b-Richtung in Metern vom linken oberen Spielfeldrand

Definition at line 156 of file referee.cpp.

7.29.3.22 void Referee::tellTeamColor (TeamColor *color*)

Sendet die eigene Farbe.

Parameters

color	Die Farbe des eigenen Teams (yellow oder blue)

Definition at line 179 of file referee.cpp.

7.29.3.23 void Referee::trueColorOfTeam (TeamColor *color*) [signal]

Gibt euch die richtige Teamfarbe.

7.29.4 Member Data Documentation

7.29.4.1 bool Referee::connection [private]

Definition at line 148 of file referee.h.

7.29.4.2 int Referee::messageSize [private]

Definition at line 147 of file referee.h.

7.29.4.3 Hermes* Referee::messengerOfTheGods [private]

Definition at line 144 of file referee.h.

7.29.4.4 int Referee::myTeamID [private]

Definition at line 146 of file referee.h.

7.29.4.5 bool Referee::ready [private]

Definition at line 151 of file referee.h.

7.29.4.6 bool Referee::testMode [private]

Definition at line 149 of file referee.h.

7.29.4.7 bool Referee::verbose [private]

Definition at line 150 of file referee.h.

7.29.4.8 unsigned int Referee::wLimit [private]

Definition at line 145 of file referee.h.

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

- referee.h
- referee.cpp

7.30 RobotThread Class Reference

The RobotThread class is responsible for the communication of all classes and is the software representation of the robot, all threads are forked there and will joinen in the end. The class will move diverent tasks to different threads.

`#include <robotThread.h>`

Collaboration diagram for RobotThread:

Public Member Functions

- RobotThread (MainWindow *mainWindow)

  RobotThread will initialise all objects.
- ~RobotThread ()

  ~RobotTread will free the heap space after programs termination
- const PathPlanning * getPathPlanning ()

  getPathPlanning
- RobotThread ()

  RobotThread will initialise all objects.
- ~RobotThread ()

  ~RobotTread will free the heap space after programs termination
- const PathPlanning * getPathPlanning ()

  getPathPlanning

Private Attributes

- MainWindow * mainWindow
- ActorLowLevel * actorLowLevel
- ActorHighLevel * actorHighLevel
- SensorHighLevel * sensorHighLevel
- SensorLowLevel * sensorLowLevel
- PathPlanning * pathPlanner
- GameEngine * gameEngine
- Cam * cam
- Game * game
- QThread threadRobotLowLevel
- QThread threadActorHighLevel
- QThread threadSensorHighLevel
- QThread threadPathPlanner
- QThread threadCam
- QThread threadGameEngine
- QThread threadGame
- PathRealizer * pathRealizer
- QThread threadPathRealizer

7.30.1 Detailed Description

The RobotThread class is responsible for the communication of all classes and is the software representation of the robot, all threads are forked there and will joinen in the end. The class will move diverent tasks to different threads.

Definition at line 40 of file Main/robotThread.h.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 RobotThread::RobotThread (MainWindow * *mainWindow*)

RobotThread will initialise all objects.

RobotThread::RobotThread.

Definition at line 24 of file Main/robotThread.cpp.

7.30.2.2 RobotThread::~RobotThread ( )

~RobotTread will free the heap space after programs termination

Destructor

Definition at line 288 of file Main/robotThread.cpp.

7.30.2.3 RobotThread::RobotThread ( )

RobotThread will initialise all objects.

RobotThread::RobotThread.

Todo : Das ist erstmal nur zum Debuggen drinnen.

Definition at line 23 of file robotThread.cpp.

7.30.2.4 RobotThread::∼RobotThread ( )

∼RobotTread will free the heap space after programs termination

7.30.3 Member Function Documentation

7.30.3.1 const PathPlanning* RobotThread::getPathPlanning ( ) [inline]

getPathPlanning

Returns

a pointer to an pathPlanner

Definition at line 38 of file robotThread.h.

7.30.3.2 const PathPlanning* RobotThread::getPathPlanning ( ) [inline]

getPathPlanning

Returns

a pointer to an pathPlanner

Definition at line 59 of file Main/robotThread.h.

7.30.4 Member Data Documentation

7.30.4.1 ActorHighLevel* RobotThread::actorHighLevel [private]

the actor high level instance which is responsible for path realizing

Definition at line 66 of file Main/robotThread.h.

7.30.4.2 ActorLowLevel * RobotThread::actorLowLevel [private]

actor low level instance for interaction with the engines

Definition at line 65 of file Main/robotThread.h.

7.30.4.3 Cam * RobotThread::cam [private]

a cam object for gathering a video stream

Definition at line 71 of file Main/robotThread.h.

7.30.4.4 Game* RobotThread::game [private]

will be the Game

Definition at line 72 of file Main/robotThread.h.

7.30.4.5 GameEngine * RobotThread::gameEngine [private]

will communicate with the angelina serve

Definition at line 70 of file Main/robotThread.h.

7.30.4.6 MainWindow* RobotThread::mainWindow [private]

pointer to mainWindow for easy connecting of signals and slots

Definition at line 62 of file Main/robotThread.h.

7.30.4.7 PathPlanning * RobotThread::pathPlanner [private]

will calculate the best path based on a given target and recognised obstacles

Definition at line 69 of file Main/robotThread.h.

7.30.4.8 PathRealizer* RobotThread::pathRealizer [private]

the actor high level instance which is responsible for path realizing

Definition at line 43 of file robotThread.h.

7.30.4.9 SensorHighLevel * RobotThread::sensorHighLevel [private]

sensor high level is responsible for object recognition and position evaluation

Definition at line 67 of file Main/robotThread.h.

7.30.4.10 SensorLowLevel * RobotThread::sensorLowLevel [private]

will read the data from laser

Definition at line 68 of file Main/robotThread.h.

7.30.4.11 QThread RobotThread::threadActorHighLevel [private]

Pathrealising and motion controler is computed in an own thread

Definition at line 76 of file Main/robotThread.h.

7.30.4.12 QThread RobotThread::threadCam [private]

gathering cam data should not slow down other threads therefore it gets an own one

Definition at line 79 of file Main/robotThread.h.

7.30.4.13 QThread RobotThread::threadGame [private]

Definition at line 81 of file Main/robotThread.h.

7.30.4.14 QThread RobotThread::threadGameEngine [private]

the comunication with the game server have to be stable

Definition at line 80 of file Main/robotThread.h.

7.30.4.15 QThread RobotThread::threadPathPlanner [private]

Pathplanning will compute the path in own thread

Definition at line 78 of file Main/robotThread.h.

7.30.4.16 QThread RobotThread::threadPathRealizer [private]

Pathrealising and motion controler is computed in an own thread

Definition at line 52 of file robotThread.h.

7.30.4.17 QThread RobotThread::threadRobotLowLevel [private]

Thread in which the player instance is accessible

Definition at line 75 of file Main/robotThread.h.

7.30.4.18 QThread RobotThread::threadSensorHighLevel [private]

Object recognition is computed in an own thread

Thread for the High Level Sensor

Definition at line 77 of file Main/robotThread.h.

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

- Main/robotThread.h
- robotThread.h
- Main/robotThread.cpp
- robotThread.cpp

7.31 SensorHighLevel Class Reference

The SensorHighLevel class is responsible for the processing of the raw laser data and adds all objects to the MapData.

`#include <sensor.h>`

Collaboration diagram for SensorHighLevel:

Public Slots

- void getLaserData (QVector< double > sensorData)
- void getSonarData (QVector< double > sonarData)
- void slotSetFilterParams (FilterParams cameraParams)
- void slotStartDetection (bool start)
- void slotColorDetected (CamColor color)
- void slotGetLaserData (QVector< double > sensorData, Position positionSignal)
- void slotSetFilterParams (FilterParams cameraParams)
- void slotStartDetection (bool start)
- void slotColorDetected (CamColor color)

Signals

- void signalSendRobotControlParams (double velocity, double turnangle)
- void signalEmergencyStopEnabled (bool)
- void signalSendLaserData (LaserPlotData laserData)
- void sendOdometryData (Position)
- void signalSendTeamColor (CamColor color)
- void signalStartColorDetection ()
- void signalSendRobotControlParams (double velocity, double turnangle)
- void signalEmergencyStopEnabled (bool)
- void signalSendLaserData (LaserPlotData laserData)
- void signalSendOdometryData (Position finalPosition)
- void signalSendTeamColor (CamColor color)
- void signalStartColorDetection ()
- void signalPlanNewPath ()

Public Member Functions

- SensorHighLevel ()
- ~SensorHighLevel ()
- SensorStates getState () const
- void setState (const SensorStates &value)
- CamColor getTeamColor () const
- void setTeamColor (const CamColor &value)
- SensorHighLevel ()

  SensorHighLevel.
- ~SensorHighLevel ()
- SensorStates getState () const

  getState
- void setState (const SensorStates &value)

  setState
- CamColor getTeamColor () const

  getTeamColor
- void setTeamColor (const CamColor &value)

  setTeamColor

Static Public Attributes

- static std::atomic_bool streamSensorEnabled

Private Member Functions

- bool recognition (QVector< QVector3D > &objects)
- QVector< QVector3D > extractObjects (QPair< QVector< double >, QVector< double > > &anglesAndDepths)
- QPair< QVector< double >
  , QVector< double > > constrainData (const QVector< double > &rawDepthsVector)
- QPair< double, double > calculateObjCenter (QPair< double, double > firstCoordinate, QPair< double, double > lastCoordinate)
- QPair< double, double > convertPolToGlobalCoordinates (QPair< double, double > coordinates)
- double distanceKartesisch (QPair< double, double > point_a, QPair< double, double > point_b)
- double distancePolar (QPair< double, double > point_a, QPair< double, double > point_b)
- void avoideCollision (QVector< double > &sensorData)

- void avoideCollision (QVector< double > &rawDepthsVector)

  avoideCollision will only
- void recognition (QList< Position > &objects, Position &transmissionPosition)

  recognition
- QList< Position > extractObjects (const ConstrainedLaserData &constrainedData)

  extractObjects
- void constrainData (const QVector< double > &filteredDepthsVector, const QVector< double > &rawDepthsVector, ConstrainedLaserData &constrainedData)

  constrainData
- Position calculateObjCenter (const ConstrainedLaserData &constrainedData, int objectBeginn, int objectEnd, SensorStates &tempState)

  calculateObjCenter
- void driveToPreposition ()

  driveToPreposition
- void puckGrabbed ()

  puckGrabbed check if we grabbed the puck

Private Attributes

- bool hadEmergency
- bool targetsSet
- QElapsedTimer timeSinceStart
- FilterParams filterParameter
- SensorStates currentState
- Position previousPosition
- Position currentPosition
- int counter
- QVector< QVector< QVector3D > > collectorObj
- double minAngle
- double maxAngle
- CamColor teamColor
- bool isInSlowTurn
- bool prepositionInitialized
- double dummyAngleOffset
- QElapsedTimer * timerWaitForValidColorFrame
- QElapsedTimer * timerToAbondonColorDetection
- QMutex * mutexState
- QMutex * mutexFilterParameter
- QMutex * mutexTeamColor
- Position transmissionPosition
- Position dummyPosition
- ConstrainedLaserData constrainedData
- QVector< Position > cPolePositions
- QList< Position > previousObjects
- QList< Position > currentObjects

Static Private Attributes

- static QMutex mutexFilterParameter
- static QMutex mutexState
- static QMutex mutexTeamColor

7.31.1 Detailed Description

The SensorHighLevel class is responsible for the processing of the raw laser data and adds all objects to the MapData.

Definition at line 35 of file sensor.h.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 SensorHighLevel::SensorHighLevel ( )

Definition at line 22 of file sensorhighlevel.cpp.

7.31.2.2 SensorHighLevel::~SensorHighLevel ( )

Definition at line 66 of file sensorhighlevel.cpp.

7.31.2.3 SensorHighLevel::SensorHighLevel ( )

SensorHighLevel.

7.31.2.4 SensorHighLevel::~SensorHighLevel ( )

7.31.3 Member Function Documentation

7.31.3.1 void SensorHighLevel::avoideCollision (QVector< double > & *rawDepthsVector*) [private]

avoideCollision will only

Parameters

rawDepthsVector	

7.31.3.2 void SensorHighLevel::avoideCollision (QVector< double > & *sensorData*) [private]

Definition at line 477 of file sensorhighlevel.cpp.

7.31.3.3 QPair<double,double> SensorHighLevel::calculateObjCenter (QPair< double, double > *firstCoordinate,* QPair< double, double > *lastCoordinate*) [private]

7.31.3.4 Position SensorHighLevel::calculateObjCenter (const ConstrainedLaserData & *constrainedData,* int *objectBeginn,* int *objectEnd,* SensorStates & *tempState*) [private]

calculateObjCenter

Parameters

constrainedData	
objectBeginn	
objectEnd	
tempState	

Returns

Todo may be cheating here with the distAB

Definition at line 860 of file sensorhighlevel.cpp.

7.31.3.5 QPair<QVector<double>, QVector<double> > SensorHighLevel::constrainData (const QVector< double > & *rawDepthsVector*) [private]

7.31.3.6 void SensorHighLevel::constrainData (const QVector< double > & *filteredDepthsVector,* const QVector< double > & *rawDepthsVector,* ConstrainedLaserData & *constrainedData*) [private]

constrainData

Parameters

filteredDepths-Vector	
rawDepthsVector	
constrainedData	

Definition at line 705 of file sensorhighlevel.cpp.

7.31.3.7 QPair<double,double> SensorHighLevel::convertPolToGlobalCoordinates (QPair< double, double > *coordinates*) [private]

7.31.3.8 double SensorHighLevel::distanceKartesisch (QPair< double, double > *point_a,* QPair< double, double > *point_b*) [private]

7.31.3.9 double SensorHighLevel::distancePolar (QPair< double, double > *point_a,* QPair< double, double > *point_b*) [private]

7.31.3.10 void SensorHighLevel::driveToPreposition ( ) [private]

driveToPreposition

Parameters

setUp	

Definition at line 440 of file sensorhighlevel.cpp.

7.31.3.11 QVector<QVector3D> SensorHighLevel::extractObjects (QPair< QVector< double >, QVector< double > > & *anglesAndDepths*) [private]

7.31.3.12 QList< Position > SensorHighLevel::extractObjects (const ConstrainedLaserData & *constrainedData*) [private]

extractObjects

SensorHighLevel::extractObjects Geht durch die sensorData list und überprüft auf entfernungen zwischen zwei sensordatenpunkten.

Parameters

constrainedData	

Returns

Parameters

sensorData	Eingangsvektor
objects	Referenz auf Vektor für Rückgabe - QVector3D beinhaltet Tiefe - Winkel (rad) - Breite

Todo also ich mache es hier mal von der simulation abhängig, ob 2 oder 3 werte für ein object genügen!

Definition at line 605 of file sensorhighlevel.cpp.

7.31.3.13 void SensorHighLevel::getLaserData (QVector< double > *sensorData*) [slot]

7.31.3.14 void SensorHighLevel::getSonarData (QVector< double > *sonarData*) [slot]

7.31.3.15 SensorStates SensorHighLevel::getState ( ) const

Definition at line 846 of file sensorhighlevel.cpp.

7.31.3.16 SensorStates SensorHighLevel::getState ( ) const

getState

Returns

the current state of the HighLevelSensor

7.31.3.17 CamColor SensorHighLevel::getTeamColor ( ) const

Definition at line 509 of file sensorhighlevel.cpp.

7.31.3.18 CamColor SensorHighLevel::getTeamColor ( ) const

getTeamColor

Returns

7.31.3.19 void SensorHighLevel::puckGrabbed ( ) [private]

puckGrabbed check if we grabbed the puck

7.31.3.20 bool SensorHighLevel::recognition (QVector< QVector3D > & *objects*) [private]

7.31.3.21 void SensorHighLevel::recognition (QList< Position > & *objects*, Position & *transmissionPosition*) [private]

recognition

SensorHighLevel::recognition Soll Objekte, die sich im angemessenen Abstand befinden erkennen und ggf identifizieren.

Parameters

objects	
transmission-Position	
objects	QVector3Ds mit x=entfernung, y=winkel (0°=links vom roboter, 90°=vorm roboter), z=breite

Returns

Definition at line 527 of file sensorhighlevel.cpp.

7.31.3.22 void SensorHighLevel::sendOdometryData (Position) [signal]

7.31.3.23 void SensorHighLevel::setState (const SensorStates & *value*)

Definition at line 852 of file sensorhighlevel.cpp.

7.31.3.24 void SensorHighLevel::setState (const SensorStates & *value*)

setState

Parameters

in	*value*	(SensorStates&)

7.31.3.25 void SensorHighLevel::setTeamColor (const CamColor & *value*)

Definition at line 515 of file sensorhighlevel.cpp.

7.31.3.26 void SensorHighLevel::setTeamColor (const CamColor & *value*)

setTeamColor

Parameters

in	*value*	(CamColor)

7.31.3.27 void SensorHighLevel::signalEmergencyStopEnabled (bool) [signal]

7.31.3.28 void SensorHighLevel::signalEmergencyStopEnabled (bool) [signal]

7.31.3.29 void SensorHighLevel::signalPlanNewPath ( ) [signal]

7.31.3.30 void SensorHighLevel::signalSendLaserData (LaserPlotData *laserData*) [signal]

7.31.3.31 void SensorHighLevel::signalSendLaserData (LaserPlotData *laserData*) [signal]

7.31.3.32 void SensorHighLevel::signalSendOdometryData (Position *finalPosition*) [signal]

7.31.3.33 void SensorHighLevel::signalSendRobotControlParams (double *velocity,* double *turnangle*) [signal]

7.31.3.34 void SensorHighLevel::signalSendRobotControlParams (double *velocity,* double *turnangle*) [signal]

7.31.3.35 void SensorHighLevel::signalSendTeamColor (CamColor *color*) [signal]

7.31.3.36 void SensorHighLevel::signalSendTeamColor (CamColor *color*) [signal]

7.31.3.37 void SensorHighLevel::signalStartColorDetection ( ) [signal]

7.31.3.38 void SensorHighLevel::signalStartColorDetection ( ) [signal]

7.31.3.39 void SensorHighLevel::slotColorDetected (CamColor *color*) [slot]

Definition at line 805 of file sensorhighlevel.cpp.

7.31.3.40 void SensorHighLevel::slotColorDetected (CamColor *color*) [slot]

7.31.3.41 void SensorHighLevel::slotGetLaserData (QVector< double > *sensorData,* Position *positionSignal*) [slot]

Todo magic number for certainty here

Todo consider removing the counter and replace through = new QElapsedTimer;

Todo angle range?

Todo consider removing the counter and replace through = new QElapsedTimer;

Definition at line 81 of file sensorhighlevel.cpp.

7.31.3.42 void SensorHighLevel::slotSetFilterParams (FilterParams *cameraParams*) [slot]

Definition at line 782 of file sensorhighlevel.cpp.

7.31.3.43 void SensorHighLevel::slotSetFilterParams (FilterParams *cameraParams*) [slot]

7.31.3.44 void SensorHighLevel::slotStartDetection (bool *start*) [slot]

Definition at line 792 of file sensorhighlevel.cpp.

7.31.3.45 void SensorHighLevel::slotStartDetection (bool *start*) [slot]

7.31.4 Member Data Documentation

7.31.4.1 QVector<QVector<QVector3D> > SensorHighLevel::collectorObj [private]

Definition at line 98 of file sensor.h.

7.31.4.2 ConstrainedLaserData SensorHighLevel::constrainedData [private]

Sensor Data with out of field lieres removed

Definition at line 164 of file sensorhighlevel.h.

7.31.4.3 int SensorHighLevel::counter [private]

Counter to collect senor data

Definition at line 97 of file sensor.h.

7.31.4.4 QVector<Position> SensorHighLevel::cPolePositions [private]

Vector with Positions of the poles

Definition at line 165 of file sensorhighlevel.h.

7.31.4.5 QList<Position> SensorHighLevel::currentObjects [private]

A list which contains the objects from the current run

Definition at line 167 of file sensorhighlevel.h.

7.31.4.6 Position SensorHighLevel::currentPosition [private]

Calculated Position of the robot

Definition at line 95 of file sensor.h.

7.31.4.7 SensorStates SensorHighLevel::currentState [private]

saves the current state

Definition at line 92 of file sensor.h.

7.31.4.8 double SensorHighLevel::dummyAngleOffset [private]

Angle difference between dummy and (unset) robot

Definition at line 149 of file sensorhighlevel.h.

7.31.4.9 Position SensorHighLevel::dummyPosition [private]

Position of the Dummy. Set after first orientation and rotating with robot

Definition at line 163 of file sensorhighlevel.h.

7.31.4.10 FilterParams SensorHighLevel::filterParameter [private]

All parameters set by the GUI are stored into filterParameter

Definition at line 76 of file sensor.h.

7.31.4.11 bool SensorHighLevel::hadEmergency [private]

Is set by the old collision avoidance

Definition at line 70 of file sensor.h.

7.31.4.12 bool SensorHighLevel::isInSlowTurn [private]

Bool needed for orientation validation

Definition at line 143 of file sensorhighlevel.h.

7.31.4.13 double SensorHighLevel::maxAngle [private]

The max Angle the roboter should turn to

Definition at line 104 of file sensor.h.

7.31.4.14 double SensorHighLevel::minAngle [private]

The min Angle the roboter should turn to

Definition at line 103 of file sensor.h.

7.31.4.15 QMutex SensorHighLevel::mutexFilterParameter [static], [private]

Definition at line 75 of file sensor.h.

7.31.4.16 QMutex* SensorHighLevel::mutexFilterParameter [private]

Pointer to mutex to set parameters by the GUI

Definition at line 156 of file sensorhighlevel.h.

7.31.4.17 QMutex SensorHighLevel::mutexState [static], [private]

Definition at line 91 of file sensor.h.

7.31.4.18 QMutex* SensorHighLevel::mutexState [private]

Pointer to mutex to get the state form outside

Definition at line 154 of file sensorhighlevel.h.

7.31.4.19 QMutex SensorHighLevel::mutexTeamColor [static], [private]

Definition at line 107 of file sensor.h.

7.31.4.20 QMutex* SensorHighLevel::mutexTeamColor [private]

Pointer to mutex to cummunicate with cam

Definition at line 158 of file sensorhighlevel.h.

7.31.4.21 bool SensorHighLevel::prepositionInitialized [private]

Bool is true if the preposition position was initialized

Definition at line 146 of file sensorhighlevel.h.

7.31.4.22 QList<Position> SensorHighLevel::previousObjects [private]

A list which contains the objects from the last run

Definition at line 166 of file sensorhighlevel.h.

7.31.4.23 Position SensorHighLevel::previousPosition [private]

First Position detected, used for validation

Definition at line 94 of file sensor.h.

7.31.4.24 static std::atomic_bool SensorHighLevel::streamSensorEnabled [static]

Definition at line 44 of file sensor.h.

7.31.4.25 bool SensorHighLevel::targetsSet [private]

If the target is set on the color field

Definition at line 71 of file sensor.h.

7.31.4.26 CamColor SensorHighLevel::teamColor [private]

saves the color of the team

Definition at line 108 of file sensor.h.

7.31.4.27 QElapsedTimer* SensorHighLevel::timerToAbondonColorDetection [private]

Pointer to timer to abondon color checking

Definition at line 153 of file sensorhighlevel.h.

7.31.4.28 QElapsedTimer* SensorHighLevel::timerWaitForValidColorFrame [private]

Pointer to timer to recheck color frame

Definition at line 152 of file sensorhighlevel.h.

7.31.4.29 QElapsedTimer SensorHighLevel::timeSinceStart [private]

Definition at line 73 of file sensor.h.

7.31.4.30 Position SensorHighLevel::transmissionPosition [private]

The position set to vertify if seen objects are in the field

Definition at line 160 of file sensorhighlevel.h.

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

- sensor.h
- sensorhighlevel.h
- sensorhighlevel.cpp

7.32 SensorLowLevel Class Reference

The SensorLowLevel class is collecting odometry and laser data from the player client.

`#include <sensorLowLevel.h>`

Collaboration diagram for SensorLowLevel:

Public Slots

- void quit ()
  quit the internal endless loop for data collection
- void run ()
  run starts the internal endless loop for data collection

Signals

- void signalLaserDataReady (QVector< double >, Position)
  signalLaserDataReady signal which sends the current laser data along with the interpolated position
- void signalLaserPlotRaw (LaserPlotData laserPlotData)
  signalLaserPlotRaw signal for the raw laser data to show in the gui
- void signalSimulationDetect ()
  signalSimulationDetect signals when the LowLevelSensor has detected if it is a simulation or not

Public Member Functions

- SensorLowLevel ()
  SensorLowLevel.
- ~SensorLowLevel ()

Static Public Member Functions

- static double angleWeightedAverage (const double &angle1, const double &weight1, const double &angle2, const double &weight2)
  angleWeightedAverage will correctly interpolate between two angles of 0 to 2∗M_PI rad

Private Types

- enum SensorState { INIT, RUN }
  The SensorState enum for internal initialization of the class.

Private Member Functions

- void readSensorData ()
  readSensorData will be called endlessly and block till it received new data

Private Attributes

- SensorState state
- PlayerCc::PlayerClient * robot
- PlayerCc::Position2dProxy * positionProxy
- PlayerCc::RangerProxy * laserProxy
- QVector< double > laserData
- int laserArrayLength
- QElapsedTimer elapsedTimer
- Position previousOdometryPosition
- Position currentOdometryPosition
- double previousOdometryTime
- double currentOdometryTime
- double laserTime
- std::atomic_bool quitting

7.32.1 Detailed Description

The SensorLowLevel class is collecting odometry and laser data from the player client.

Definition at line 24 of file sensorLowLevel.h.

7.32.2 Member Enumeration Documentation

7.32.2.1 enum SensorLowLevel::SensorState [private]

The SensorState enum for internal initialization of the class.

Enumerator:

INIT

RUN

Definition at line 31 of file sensorLowLevel.h.

7.32.3 Constructor & Destructor Documentation

7.32.3.1 SensorLowLevel::SensorLowLevel ( )

SensorLowLevel.

Definition at line 15 of file sensorLowLevel.cpp.

7.32.3.2 SensorLowLevel::~SensorLowLevel ( )

Definition at line 47 of file sensorLowLevel.cpp.

7.32.4 Member Function Documentation

7.32.4.1 double SensorLowLevel::angleWeightedAverage (const double & *angle1*, const double & *weight1*, const double & *angle2*, const double & *weight2*) [static]

angleWeightedAverage will correctly interpolate between two angles of 0 to 2*M_PI rad

Parameters

in	*angle1*	(double&)
in	*weight1*	(double&)
in	*angle2*	(double&)
in	*weight2*	(double&)

Returns

the interpolated and weighted new angle value

See Also

http://stackoverflow.com/a/1687116

Definition at line 227 of file sensorLowLevel.cpp.

7.32.4.2 void SensorLowLevel::quit ( ) [slot]

quit the internal endless loop for data collection

Definition at line 56 of file sensorLowLevel.cpp.

7.32.4.3 void SensorLowLevel::readSensorData ( ) [private]

readSensorData will be called endlessly and block till it received new data

the sensor count of the real robot has 361 values

Definition at line 68 of file sensorLowLevel.cpp.

7.32.4.4 void SensorLowLevel::run ( ) [slot]

run starts the internal endless loop for data collection

Definition at line 60 of file sensorLowLevel.cpp.

7.32.4.5 void SensorLowLevel::signalLaserDataReady (QVector< double >, Position) [signal]

signalLaserDataReady signal which sends the current laser data along with the interpolated position

7.32.4.6 void SensorLowLevel::signalLaserPlotRaw (LaserPlotData *laserPlotData*) [signal]

signalLaserPlotRaw signal for the raw laser data to show in the gui

Parameters

laserPlotData	for the GUI

7.32.4.7 void SensorLowLevel::signalSimulationDetect ( ) [signal]

signalSimulationDetect signals when the LowLevelSensor has detected if it is a simulation or not

7.32.5 Member Data Documentation

7.32.5.1 Position SensorLowLevel::currentOdometryPosition [private]

current Odometry position

Definition at line 95 of file sensorLowLevel.h.

7.32.5.2 double SensorLowLevel::currentOdometryTime [private]

current Odometry timestamp

Definition at line 98 of file sensorLowLevel.h.

7.32.5.3 QElapsedTimer SensorLowLevel::elapsedTimer [private]

This timer is needed in order to interpolate the position

Definition at line 92 of file sensorLowLevel.h.

7.32.5.4 int SensorLowLevel::laserArrayLength [private]

the legth determines if it is real(>360) or simulation(360)

Definition at line 90 of file sensorLowLevel.h.

7.32.5.5 QVector<double> SensorLowLevel::laserData [private]

array which contains the received laser data

Definition at line 89 of file sensorLowLevel.h.

7.32.5.6 PlayerCc::RangerProxy* SensorLowLevel::laserProxy [private]

laser proxy for the laser data reading

Definition at line 87 of file sensorLowLevel.h.

7.32.5.7 double SensorLowLevel::laserTime [private]

laser data timestamp

Definition at line 99 of file sensorLowLevel.h.

7.32.5.8 PlayerCc::Position2dProxy* SensorLowLevel::positionProxy [private]

position proxy for odometry data reading

Definition at line 86 of file sensorLowLevel.h.

7.32.5.9 Position SensorLowLevel::previousOdometryPosition [private]

previous Odometry position

Definition at line 94 of file sensorLowLevel.h.

7.32.5.10 double SensorLowLevel::previousOdometryTime [private]

previous Odometry timestamp

Definition at line 97 of file sensorLowLevel.h.

7.32.5.11 std::atomic_bool SensorLowLevel::quitting [private]

is set to true via Signal&Slot if the no more laser data is needed

Definition at line 101 of file sensorLowLevel.h.

7.32.5.12 PlayerCc::PlayerClient* SensorLowLevel::robot [private]

pointer to the global static player client

Definition at line 85 of file sensorLowLevel.h.

7.32.5.13 SensorState SensorLowLevel::state [private]

holds the internal state of the LowLevelSensor

Definition at line 83 of file sensorLowLevel.h.

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

- sensorLowLevel.h
- sensorLowLevel.cpp

7.33 tkqt::spline Class Reference

The spline class => This class will create an cubic hermite spline from two given vectors.

`#include <spline.h>`

Collaboration diagram for tkqt::spline:

Public Member Functions

- void set_points (const QVector< double > &x, const QVector< double > &y, bool cubic_spline=true)

  set_points this method will compute the cubic hermite spline from given QVector<double> =x and QVector<double> =y
- double operator() (double x) const

  operator ()

Private Attributes

- QVector< double > m_x
- QVector< double > m_y
- QVector< double > m_a
- QVector< double > m_b
- QVector< double > m_c
- QVector< double > m_d

7.33.1 Detailed Description

The spline class => This class will create an cubic hermite spline from two given vectors.

Definition at line 126 of file spline.h.

7.33.2 Member Function Documentation

7.33.2.1 double tkqt::spline::operator() (double *x*) const

operator ()

Parameters

x	

Returns

Definition at line 231 of file spline.cpp.

7.33.2.2 void tkqt::spline::set_points (const QVector< double > & *x*, const QVector< double > & *y*, bool *cubic_spline* = `true`)

set_points this method will compute the cubic hermite spline from given QVector<double> =x and QVector<double> =y

Parameters

`in`	*x*	(QVector<double>) x-points of the spline
`in`	*y*	(QVector<double>) y-points of the spline
`in`	*cubic_spline*	(bool) if the spline should be a cubic spline or not

Warning

x,y have to be sorted ascendingly.

Todo : sort x and y, rather than returning an error

Definition at line 170 of file spline.cpp.

7.33.3 Member Data Documentation

7.33.3.1 QVector<double> tkqt::spline::m_a `[private]`

interpolation parameter m_a of f(x) = m_a∗(x-x_i)^3 + m_b∗(x-x_i)^2 + m_c∗(x-x_i) + m_d

Definition at line 134 of file spline.h.

7.33.3.2 QVector<double> tkqt::spline::m_b `[private]`

interpolation parameter m_b of f(x) = m_a∗(x-x_i)^3 + m_b∗(x-x_i)^2 + m_c∗(x-x_i) + m_d

Definition at line 134 of file spline.h.

7.33.3.3 QVector<double> tkqt::spline::m_c [private]

interpolation parameter m_c of f(x) = m_a∗(x-x_i)^3 + m_b∗(x-x_i)^2 + m_c∗(x-x_i) + m_d

Definition at line 134 of file spline.h.

7.33.3.4 QVector<double> tkqt::spline::m_d [private]

interpolation parameter m_d of f(x) = m_a∗(x-x_i)^3 + m_b∗(x-x_i)^2 + m_c∗(x-x_i) + m_d

Definition at line 134 of file spline.h.

7.33.3.5 QVector<double> tkqt::spline::m_x [private]

x coordinates of points

Definition at line 129 of file spline.h.

7.33.3.6 QVector<double> tkqt::spline::m_y [private]

y coordinates of points

Definition at line 129 of file spline.h.

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

- spline.h
- spline.cpp

Chapter 8

File Documentation

8.1 actorhighlevel.cpp File Reference

```
#include "actorhighlevel.h"
#include <QDebug>
#include <QElapsedTimer>
#include <QTimer>
#include <QMutexLocker>
#include <QCoreApplication>
#include <QThread>
#include "AI/pathplanning.h"
#include "Data/mapdata.h"
#include "Data/define.h"
#include "Sensor/sensorLowLevel.h"
#include "Structs/pathplotdata.h"
```

Include dependency graph for actorhighlevel.cpp:

8.2 actorhighlevel.h File Reference

```
#include "Data/obstacle.h"
#include "Structs/pidplotdata.h"
#include "Structs/pidparams.h"
#include "Actor/spline.h"
#include <QMutex>
#include "atomic"
```

Include dependency graph for actorhighlevel.h: This graph shows which files directly or indirectly include this file:

Classes

- class ActorHighLevel

  The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

Enumerations

- enum StatePathProcessing {
  STOP, RUNNING, RELEASE_PUCK, RELEASE_PUCK_FROM_PUSH,
  PUSH_AND_RELEASE_PUCK, GATHER_PUCK }

The StatePathProcessing enum internal state-machine.

8.2.1 Enumeration Type Documentation

8.2.1.1 enum StatePathProcessing

The StatePathProcessing enum internal state-machine.

Enumerator:

STOP first enum value, represents the stop state

RUNNING second enum value, represents the running and therefore pathrealising state

RELEASE_PUCK

RELEASE_PUCK_FROM_PUSH

PUSH_AND_RELEASE_PUCK

GATHER_PUCK

Definition at line 21 of file actorhighlevel.h.

8.3 actorLowLevel.cpp File Reference

```
#include "actorLowLevel.h"
#include "Data/define.h"
#include "Data/mapdata.h"
#include "Main/player.h"
#include "AI/pathplanning.h"
#include <QDebug>
#include <QTime>
#include <QThread>
#include <libplayerc++/playerc++.h>
```

Include dependency graph for actorLowLevel.cpp:

8.4 actorLowLevel.h File Reference

```
#include <QObject>
#include "Structs/position.h"
```

Include dependency graph for actorLowLevel.h: This graph shows which files directly or indirectly include this file:

Classes

- class ActorLowLevel

  The ActorLowLevel class Class for accessing the PlayerCc::Position2dProxy, as single point of access.

Namespaces

- namespace PlayerCc

8.5 cam.cpp File Reference

```
#include "cam.h"
#include "Data/define.h"
#include "Sensor/cameraparams.h"
#include <QColor>
#include <QDebug>
#include <QTimer>
#include <QMutex>
#include <QThread>
```

Include dependency graph for cam.cpp:

8.6 cam.h File Reference

```
#include <atomic>
#include "Sensor/cameraparams.h"
#include <opencv2/opencv.hpp>
```

Include dependency graph for cam.h: This graph shows which files directly or indirectly include this file:

Classes

- class Cam

  The Cam class will stream cam data to GUI-panal for recognition of poles color.

Namespaces

- namespace cv

Enumerations

- enum CamColor {
  GREEN, YELLOW, BLUE, NONE,
  ERROR }

8.6.1 Enumeration Type Documentation

8.6.1.1 enum CamColor

Enumerator:

GREEN recognised color

YELLOW recognised color

BLUE recognised color

NONE inititial color

ERROR error signalisation

Definition at line 19 of file cam.h.

8.7 cameraparams.h File Reference

```
#include <QColor>
```

Include dependency graph for Sensor/cameraparams.h: This graph shows which files directly or indirectly include this file:

Classes

- struct CameraParams

  The CameraParams struct represents the cam params for calibration.

8.8 cameraparams.h File Reference

```
#include <QColor>
```

Include dependency graph for Structs/cameraparams.h: This graph shows which files directly or indirectly include this file:

Classes

- struct CameraParams

  The CameraParams struct represents the cam params for calibration.

8.9 constrainedlaserdata.cpp File Reference

```
#include "constrainedlaserdata.h"
```

Include dependency graph for constrainedlaserdata.cpp:

8.10 constrainedlaserdata.h File Reference

```
#include <QList>
```

Include dependency graph for constrainedlaserdata.h: This graph shows which files directly or indirectly include this file:

Classes

- class ConstrainedLaserData

  The ConstrainedLaserData class is a Datapacket containing the processed data from the LowLevelSensor and is used for sharing references of the data through the SensorHighLevel.

8.11 cvimagewidget.h File Reference

```
#include <QWidget>
#include <QImage>
#include <QPainter>
#include <opencv2/opencv.hpp>
```

Include dependency graph for GUI/cvimagewidget.h:

Classes

- class CVImageWidget

  The CVImageWidget class will draw the video stream directy instead of bytewise.

8.12 cvimagewidget.h File Reference

```
#include <QWidget>
#include <QImage>
#include <QPainter>
#include <opencv2/opencv.hpp>
```

Include dependency graph for Plots/cvimagewidget.h:

Classes

- class CVImageWidget

  The CVImageWidget class will draw the video stream directy instead of bytewise.

8.13 define.h File Reference

```
#include <QString>
#include <QBrush>
#include <QPen>
#include "Structs/position.h"
```

Include dependency graph for define.h: This graph shows which files directly or indirectly include this file:

Namespaces

- namespace config

Macros

- #define _USE_MATH_DEFINES

Functions

- static const QPen config::guiPenOpponent (QBrush(QColor(201, 40, 27)), 4.0, Qt::SolidLine)
- static const QPen config::guiPenMe (QBrush(QColor(109, 191, 55)), 4.0, Qt::SolidLine)
- static const QBrush config::guiBrushMe (QBrush(QColor(109, 191, 55)))
- static const QPen config::guiPenOrient (QBrush(QColor(133, 100, 84)), 1.0, Qt::SolidLine)
- static const QPen config::guiPenDummy (QBrush(QColor(Qt::cyan)), 1.0, Qt::SolidLine)
- static const QPen config::guiPenTarget (QBrush(QColor(Qt::gray)), 1.0, Qt::SolidLine)
- static const QPen config::guiPenPole (QBrush(QColor(109, 191, 55)), 1.0, Qt::SolidLine)
- static const QBrush config::guiBrushPole (QBrush(QColor(109, 191, 55)))
- static const QPen config::guiPenPuckMe (QBrush(QColor(109, 191, 55)), 2.0, Qt::SolidLine)
- static const QPen config::guiPenPuckOpponent (QBrush(QColor(201, 40, 27)), 2.0, Qt::SolidLine)
- static const QPen config::guiPenPuckUndef (QBrush(QColor(127, 127, 127)), 2.0, Qt::SolidLine)
- static const QPen config::guiPenPuckMeOuter (QBrush(QColor(109, 191, 55, 127)), 2.0, Qt::SolidLine)
- static const QPen config::guiPenPuckOpponentOuter (QBrush(QColor(201, 40, 27, 127)), 2.0, Qt::SolidLine)
- static const QPen config::guiPenPuckUndefOuter (QBrush(QColor(127, 127, 127, 127)), 2.0, Qt::SolidLine)

- static const QPen config::guiPenFieldPrimary (QBrush(QColor(Qt::white)), 1.0, Qt::SolidLine)
- static const QPen config::guiPenFieldSecondary (QBrush(QColor(Qt::lightGray)), 1.0, Qt::SolidLine)
- static const QBrush config::guiBrushPuckMoving (QColor(QColor(215, 69, 232)))
- static const QBrush config::guiBrushPuckBlocked (QColor(QColor(Qt::black)))
- static const QBrush config::guiBrushGoalBlue (QColor(QColor(38, 66, 115)))
- static const QBrush config::guiBrushGoalYellow (QColor(QColor(255, 211, 36)))
- static const QBrush config::guiBrushGoalUndef (QColor(Qt::lightGray))
- static const Position config::gameGoalSlotL (geoGoalMarginSide-geoRobotForkCenterDist, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2.0, 0)
- static const Position config::gameGoalSlotM (geoGoalMarginSide+0.5 ∗geoGoalWidth, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight-geoRobotForkCenterDist, M_PI_2)
- static const Position config::gameGoalSlotR (geoGoalMarginSide+1.0 ∗geoGoalWidth+geoRobotForkCenterDist, geoFieldHeight-geoGoalMarginBottom-geoGoalHeight/2.0, M_PI)
- static const Position config::gameGoalSlotOutsideLeft (0.50 ∗geoGoalMarginSide, geoFieldHeight-geoGoalMarginBottom-2 ∗geoGoalHeight, 3.0/4.0 ∗M_PI)
- static const Position config::gameGoalSlotOutsideRight (geoFieldWidth-0.50 ∗geoGoalMarginSide, geoFieldHeight-geoGoalMarginBottom-2 ∗geoGoalHeight, 1.0/4.0 ∗M_PI)
- static const Position config::gameAnnoyPositionL (geoGoalMarginSide, geoGoalMarginBottom+geoGoalHeight/2.0, 0)
- static const Position config::gameAnnoyPositionR (geoGoalMarginSide+geoGoalWidth, geoGoalMarginBottom+geoGoalHeight/2.0, M_PI)

Variables

- static const double config::geoFieldWidth = 3.0
- static const double config::geoFieldHeight = 5.0
- static const double config::geoGoalWidth = 1.0
- static const double config::geoGoalHeight = (5.0/3.0)/4.0
- static const double config::geoGoalMarginBottom = (5.0/3.0)/4.0
- static const double config::geoGoalMarginSide = 1.0
- static const double config::geoRobotForkCenterDist = 0.20
- static const double config::geoPol_1_2 = (5.0/3.0)/4.0
- static const double config::geoPol_2_3 = 0.75∗(5.0/3.0)
- static const double config::geoPol_3_4 = (5.0/3.0)/2.0
- static const double config::geoPol_1_14 = geoFieldWidth
- static const double config::geoPoleRadiusReal = 0.03
- static const double config::geoPoleRadiusSim = 0.06
- static const double config::geoPuckRadiusTopReal = 0.02
- static const double config::geoPuckRadiusTopSim = 0.03525
- static const double config::geoPuckRadiusBottom = 0.13 / 2.0
- static const int config::teamID = 7
- static const double config::periodAlive = 1000
- static const double config::periodEgoPos = 250
- static const QString config::refIP = "localhost"
- static const int config::refPort = 10000
- static const bool config::refVerbose = false
- static const int config::periodTillAnnoy = 250000
- static const int config::GUIopponent = 0
- static const int config::GUIself = 1
- static const int config::GUIpuckOpponent = 2
- static const int config::GUIpuckSelf = 3
- static const int config::GUIpuckUndef = 4
- static const int config::GUItarget = 5
- static const double config::GUIREMOTEVELOCITY = 0.1

- static const double config::GUIREMOTETURNRATE = 0.174
- static const double config::guiPuckKlickTolerance = 10/100.0

  wie nah muss auf das Zentrum eines Pucks in der Map geklickt werden, um ihn als geklickt wahrzunehmen. In Meter, bzw 100px
- static const int config::ROBOSIZE_CM = 40
- static const int config::PUCKSIZE_INNER_CM = 5
- static const int config::PUCKSIZE_OUTER_CM = 13
- static const int config::POLESIZE_CM = 6
- static const int config::BORDERTOP_CM = 75
- static const int config::BORDERSIDE_CM = 75
- static const int config::FIELDHEIGHT_CM = geoFieldHeight∗100
- static const int config::FIELDWIDTH_CM = geoFieldWidth∗100
- static const int config::TARGETSIZE_CM = 20
- static const int config::PATH_SHOWRES = 160
- static const double config::SENSOR_OUT_OF_FIELD_TOLERANCE = 0.15
- static const double config::SENSOR_OBJECTWIDTH_ROBO = 0.25
- static const double config::SENSOR_MAX_DISTANCE_OF_OBJ = 0.08
- static const double config::SENSOR_MAX_RANGE_ORIENTATION = 3.40
- static const double config::SENSOR_MAX_RANGE_RECOGNITION
- static const double config::SENSOR_COLLISION_AT = 0.5
- static const double config::SENSOR_DELTA_ANGLE = M_PI/180∗10
- static const double config::SENSOR_RADIUS_ROBOT = 0.23
- static const int config::SENSOR_WAIT_COUNTER = 10
- static const double config::SENSOR_MEASUREMENT_DEVIATION = 0.15
- static const double config::CAM_ROI_WIDTH_RELATIVE = 0.2
- static const double config::CAM_ROI_HEIGHT_RELATIVE = 0.3
- static const double config::CAM_ROI_OFFSET_HORIZONTAL_RELATIVE = 0.5 - CAM_ROI_WIDTH_RELATIVE / 2
- static const double config::CAM_ROI_OFFSET_VERTICAL_RELATIVE = 1 - 0.05 - CAM_ROI_HEIGHT_RELATIVE
- static const double config::CAM_PERCENTAGE_NON_COLOR_DETECTION = 0.5
- static const double config::ORIENTATION_APPROXIMATION_VALUE = 0.05
- static const double config::ORIENTATION_SENSOR_ODOMETRIE_DELTA = 0.03
- static const double config::DIST_TO_PUCK_BEFORE_GATHERING_IT = 0.40
- static const double config::gameWieWeitMussDerGegnerVomZielEntferntSeinImAnnoyModus = 1.0
- static const double config::DUMP_SLOT_2_3 = 1.042
- static const double config::DUMP_SLOT_3_4 = 2.083
- static const double config::DUMP_SLOT_4_5 = 2.917
- static const double config::DUMP_SLOT_5_6 = 3.958
- static const double config::TARGET_POLE_VARIANCE = 0.13
- static const double config::DISTANCE_TO_WAITING_LINE = DIST_TO_PUCK_BEFORE_GATHERING_IT
- static const double config::gameBisZuWelchemAbstandWirdZielwackelnGefiltert = 0.10
- static const double config::gameZielwackelfilterTiefpassKoeffizient = 1e-1
- static const double config::gameWievielBesserMussEinPuckSeinUmDasZielZuWechseln = 0.1

  in Prozent
- static const double config::gameMinimumDistanceEnemyToDumpSlot = 1.0

  Wie weit muss der Gegner davon entfernt sein, damit ein Dump Slot ausgewählt werden kann.
- static const double config::gameMinimumDistanceToEnemyRobot = 1.5 ∗ SENSOR_RADIUS_ROBOT
- static const double config::puckIsCloseToPoleDistance = 0.50
- static const int config::pathMaxWPIterations = 10000
- static const double config::pathGridSpacingBase = 0.05
- static const double config::pathPlanningEnabledUpwardsOfThisDistance = 0.10
- static const double config::pathArenaMinX = 0
- static const double config::pathArenaMaxX = geoFieldWidth

- static const double config::pathArenaMinY = 0
- static const double config::pathArenaMaxY = geoFieldHeight
- static const double config::pathArenaFieldAvoidMaxY = geoPol_1_2 + geoPol_2_3
- static const double config::pathRobotRadius = 0.26
- static const double config::pathPoleCloseDist = pathRobotRadius + 0.10 + geoPoleRadiusReal
- static const double config::pathPoleCloseCost = 100.0
- static const double config::pathPoleStartCost = 5.0
- static const double config::pathPoleFarDist = pathPoleCloseDist + 0.2
- static const double config::pathEnemyCloseDist = pathRobotRadius + 0.10 + pathRobotRadius
- static const double config::pathEnemyCloseCost = 10.0
- static const double config::pathEnemyStartCost = 1.0
- static const double config::pathEnemyFarDist = pathEnemyCloseDist + 0.30
- static const double config::pathPuckCloseDist = pathRobotRadius + 0.01 + geoPuckRadiusBottom
- static const double config::pathPuckCloseCost = 0.1
- static const double config::pathPuckStartCost = 0.05
- static const double config::pathPuckFarDist = pathPuckCloseDist + 0.10
- static const double config::pathTargetApproachAngleInfluenceDistance = 0.0
- static const double config::pathTargetApproachAngleMaxDeviationWithoutFullCost = 15.0 ∗M_PI/180.0
- static const double config::pathTargetApproachAngleFullCost = 100.0
- static const double config::pathAdjacencyMultiplier = 50.0
- static const double config::actorWaypointReachedDistance = 0.08

  Theshold in m to destination is reached.
- static const double config::actorWaypointReachedDiffChange = 0
- static const double config::actorWaypointMaxAngleDeviation = 2.5 /180.0∗M_PI

  Wie weit darf der Roboterwinkel vom Zielwinkel abweichen um noch als erreicht zu gelten.
- static const double config::actorDistanceOfTargetOnSpline = 0.2/ pathGridSpacingBase

  Wie weit soll der PID-Sollpunkt dem der Roboter hinterherfährt auf dem Spline maximal entfernt sein (in #-Wegpunkten, muss keine ganze Zahl sein)
- static const double config::actorGatherPuckDistance

  Wie weit wird vorwärts gefahren um einen Puck aufzunehmen.
- static const double config::actorReleasePuckDistance = 0.25

  Wie weit wird beim Puck loslassen zurückgefahren.
- static const double config::actorPushPuckDistance = 0.20

  Wie weit wird der Puck aus der Arena gefahren.
- static const double config::actorPushAndReleaseAdditionalReverseDist = 0.025

  Wie weit wird mehr zurück gefahren als vor bei push and release.
- static const double config::actorPeriodMotionControl = 1000.0 / 200.0

  Wie schnell wird der PID Regler ausgeführt (1000.0 / x Hz)
- static const double config::actorWPLowPassAlpha = 1e-20

  Koeffizient bei Wegpunkt-Tiefpass. Sollte nahe, aber nicht 0 sein. Guter Wert: (1.0 / config::actorPeriodMotionControl) / ((1.0 / config::actorPeriodMotionControl) + (config::actorPeriodMotionControl+1))
- static const double config::actorMinAngleLimiter = 15.0 ∗M_PI/180.0
- static const double config::actorMaxAngleLimiter = 60.0 ∗M_PI/180.0
- static const double config::actorMaxI = 10

  Maximaler PID-I-Anteil.
- static const double config::actorLowPass = 1e-0

  Tiefpassfilterkoeffizient für Winkel (1e-10 = stark, 1e-0 = aus)
- static const double config::obstacleCoordinateTolerance = 0.10
- static const int config::obstacleNumberOfPucks = 6
- static const int config::obstacleNumberOfPoles = 14
- static const double config::mapPolePuckFusionDistance = geoPoleRadiusReal + geoPuckRadiusBottom + 0.20
- static const double config::mapPuckPuckFusionDistance = 3 ∗ geoPuckRadiusBottom

Abstand, bei dem zwei Pucks zu einem zusammengefasst werden (m). (Wenn der Abstand zwischen zwei Pucks exakt 2∗Puckradius ist, berühren sie sich bereits)

- static const double config::mapIgnorePuckInsideEnemyDistance = SENSOR_RADIUS_ROBOT + 0.5 ∗ geoPuckRadiusBottom

  Wenn ein Puck innerhalb diesen Abstands vom Gegner erkannt wird, ist es gar kein Puck.
- static const double config::mapAbstandRoboterZentrumZuGabel = 0.20
- static const double config::mapToleranzBisWohinEinPuckInDerGabelIstMIN = 0.17
- static const double config::mapToleranzBisWohinEinPuckInDerGabelIstMAX = 0.27
- static const bool config::enableDebugMapData = false
- static const bool config::enableDebugOrientation = false
- static const bool config::enableDebugActorLowLevel = false
- static const bool config::enableDebugActorHighLevel = false
- static const bool config::enableDebugSensorLowLevel = false
- static const bool config::enableDebugSensorHighLevel = false
- static const bool config::enableDebugPathPlanning = false
- static const bool config::enableDebugGame = false
- static const bool config::enableDebugMainwindow = false
- static const bool config::enableDebugCam = false

8.13.1 Macro Definition Documentation

8.13.1.1 #define _USE_MATH_DEFINES

Definition at line 4 of file define.h.

8.14 dynsections.js File Reference

Functions

- function toggleVisibility (linkObj)
- function updateStripes ()
- function toggleLevel (level)
- function toggleFolder (id)
- function toggleInherit (id)

8.14.1 Function Documentation

8.14.1.1 function toggleFolder (*id*)

Definition at line 48 of file dynsections.js.

8.14.1.2 function toggleInherit (*id*)

Definition at line 84 of file dynsections.js.

8.14.1.3 function toggleLevel (*level*)

Definition at line 27 of file dynsections.js.

8.14.1.4 function toggleVisibility (*linkObj*)

Definition at line 1 of file dynsections.js.

8.14.1.5 function updateStripes ( )

Definition at line 22 of file dynsections.js.

8.15 filterparams.h File Reference

```
#include <QObject>
```

Include dependency graph for filterparams.h: This graph shows which files directly or indirectly include this file:

Classes

- struct FilterParams

8.16 game.cpp File Reference

```
#include "game.h"
#include <QDebug>
#include <QCoreApplication>
#include <QThread>
#include <QElapsedTimer>
#include "AI/gameengine.h"
#include "AI/pathplanning.h"
#include "Data/mapdata.h"
#include "Sensor/cam.h"
#include "Sensor/sensorLowLevel.h"
#include "Data/define.h"
```

Include dependency graph for game.cpp:

8.17 game.h File Reference

```
#include <QObject>
#include <QList>
#include <QPair>
#include <atomic>
#include "Data/mapdata.h"
#include "Structs/position.h"
```

Include dependency graph for game.h: This graph shows which files directly or indirectly include this file:

Classes

- class Game

8.18 gameengine.cpp File Reference

```
#include "gameengine.h"
#include <QDebug>
#include <QThread>
#include <referee.h>
#include "Data/define.h"
#include "AI/game.h"
#include "AI/pathplanning.h"
#include "Data/mapdata.h"
#include "Sensor/cam.h"
#include "Structs/position.h"
#include "Actor/actorhighlevel.h"
```

Include dependency graph for gameengine.cpp:

8.19 gameengine.h File Reference

```
#include <QObject>
#include <QTimer>
#include <referee.h>
```

Include dependency graph for gameengine.h: This graph shows which files directly or indirectly include this file:

Classes

- class GameEngine

8.20 hermes.cpp File Reference

```
#include "hermes.h"
#include "referee.h"
```

Include dependency graph for hermes.cpp:

8.21 hermes.h File Reference

```
#include <QTcpSocket>
#include "hermescodes.h"
```

Include dependency graph for hermes.h: This graph shows which files directly or indirectly include this file:

Classes

- class Hermes

8.22 hermescodes.h File Reference

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

Macros

- #define HERMES_CONNECT 1

- #define HERMES_STATUS 2
- #define HERMES_LEFT_PLAYGROUND 3
- #define HERMES_GAME_START 4
- #define HERMES_GAME_OVER 5
- #define HERMES_READY 6
- #define HERMES_DONE 7
- #define HERMES_ERROR 8
- #define HERMES_KEEP_ALIVE 9
- #define HERMES_DATA_T1 11
- #define HERMES_DATA_T2 12
- #define HERMES_DATA_T3 13
- #define HERMES_DATA_T4 14
- #define HERMES_SCORE 21
- #define HERMES_A_B 22
- #define HERMES_LOOKINGFOR 31
- #define HERMES_ANGELINAFOUND 32
- #define HERMES_DETECTION_START 41
- #define HERMES_STOP_MOVEMENT 42
- #define HERMES_TEAMCOLOR 43

8.22.1 Macro Definition Documentation

8.22.1.1 #define HERMES_A_B 22

Definition at line 45 of file hermescodes.h.

8.22.1.2 #define HERMES_ANGELINAFOUND 32

Definition at line 48 of file hermescodes.h.

8.22.1.3 #define HERMES_CONNECT 1

Definition at line 26 of file hermescodes.h.

8.22.1.4 #define HERMES_DATA_T1 11

Definition at line 38 of file hermescodes.h.

8.22.1.5 #define HERMES_DATA_T2 12

Definition at line 39 of file hermescodes.h.

8.22.1.6 #define HERMES_DATA_T3 13

Definition at line 40 of file hermescodes.h.

8.22.1.7 #define HERMES_DATA_T4 14

Definition at line 41 of file hermescodes.h.

8.22.1.8 #define HERMES_DETECTION_START 41

Definition at line 50 of file hermescodes.h.

8.22.1.9 #define HERMES_DONE 7

Definition at line 33 of file hermescodes.h.

8.22.1.10 #define HERMES_ERROR 8

Definition at line 34 of file hermescodes.h.

8.22.1.11 #define HERMES_GAME_OVER 5

Definition at line 31 of file hermescodes.h.

8.22.1.12 #define HERMES_GAME_START 4

Definition at line 30 of file hermescodes.h.

8.22.1.13 #define HERMES_KEEP_ALIVE 9

Definition at line 35 of file hermescodes.h.

8.22.1.14 #define HERMES_LEFT_PLAYGROUND 3

Definition at line 29 of file hermescodes.h.

8.22.1.15 #define HERMES_LOOKINGFOR 31

Definition at line 47 of file hermescodes.h.

8.22.1.16 #define HERMES_READY 6

Definition at line 32 of file hermescodes.h.

8.22.1.17 #define HERMES_SCORE 21

Definition at line 44 of file hermescodes.h.

8.22.1.18 #define HERMES_STATUS 2

Definition at line 27 of file hermescodes.h.

8.22.1.19 #define HERMES_STOP_MOVEMENT 42

Definition at line 51 of file hermescodes.h.

8.22.1.20 #define HERMES_TEAMCOLOR 43

Definition at line 52 of file hermescodes.h.

8.23 laserplotdata.h File Reference

```
#include <QVector>
#include <QObject>
```

Include dependency graph for laserplotdata.h: This graph shows which files directly or indirectly include this file:

Classes

- struct LaserPlotData

  The LaserPlotData struct is the Datapacket for plotting the laser data.

8.24 log.cpp File Reference

```
#include "log.h"
#include <atomic>
#include <iostream>
#include <QMutex>
#include <QStringList>
#include <QDebug>
#include "GUI/mainwindow.h"
```

Include dependency graph for log.cpp:

8.25 log.h File Reference

```
#include <atomic>
#include <QtMsgHandler>
#include "Structs/logparams.h"
```

Include dependency graph for log.h: This graph shows which files directly or indirectly include this file:

Classes

- class Log

  The Log class.

8.26 logparams.h File Reference

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

Classes

- struct LogParams

  The LogParams struct describes the current logging level.

8.27 LogParams.h File Reference

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

Classes

- struct LogParams

  The LogParams struct describes the current logging level.

8.28 main.cpp File Reference

```
#include <QApplication>
#include <QDebug>
#include "Logging/log.h"
#include "GUI/mainwindow.h"
#include "Main/robotThread.h"
#include "Main/player.h"
```

Include dependency graph for main.cpp:

Functions

- int main (int argc, char ∗argv[])

  main will create the GUI and the RobotThread and waits until all components are finished

8.28.1 Function Documentation

8.28.1.1 int main (int *argc*, char ∗ *argv[]*)

main will create the GUI and the RobotThread and waits until all components are finished

Parameters

argc	
argv	

Returns

Definition at line 15 of file main.cpp.

8.29 mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include <QDebug>
#include <QTimer>
#include <QGraphicsScene>
#include "ui_mainwindow.h"
#include "Logging/log.h"
#include "Data/define.h"
#include "Data/mapdata.h"
#include "AI/pathplanning.h"
#include "Sensor/sensorhighlevel.h"
#include "Sensor/cam.h"
#include "Actor/actorhighlevel.h"
#include "Structs/pathplotdata.h"
#include "Structs/pidplotdata.h"
#include "Structs/laserplotdata.h"
#include "Structs/pidparams.h"
#include "Main/player.h"
#include <libplayerc++/playerc++.h>
```

Include dependency graph for GUI/mainwindow.cpp:

Variables

- QElapsedTimer timer

8.29.1 Variable Documentation

8.29.1.1 QElapsedTimer timer

Definition at line 21 of file GUI/mainwindow.cpp.

8.30 mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include <cmath>
#include <QDebug>
#include <QTimer>
#include <QGraphicsScene>
#include "Logging/log.h"
#include "Logging/LogParams.h"
#include "Plots/pathplotdata.h"
#include "Plots/pidplotdata.h"
#include "Plots/LaserPlotData.h"
#include "ui_mainwindow.h"
#include "Data/define.h"
#include "Data/mapdata.h"
#include "AI/pathplanning.h"
#include "Sensor/sensorhighlevel.h"
#include "Sensor/cam.h"
#include "Actor/actorhighlevel.h"
#include "Structs/pathplotdata.h"
#include "Structs/pidplotdata.h"
#include "Structs/laserplotdata.h"
#include "Structs/pidparams.h"
```

Include dependency graph for mainwindow.cpp:

8.31 mainwindow.h File Reference

```
#include <QMainWindow>
#include <QList>
#include <QPair>
#include <QElapsedTimer>
#include <opencv2/core/core.hpp>
#include "Structs/logparams.h"
#include "Structs/laserplotdata.h"
#include "Structs/cameraparams.h"
#include "Structs/filterparams.h"
```

Include dependency graph for GUI/mainwindow.h: This graph shows which files directly or indirectly include this file:

Classes

- class MainWindow

  The MainWindow class creates the GUI and connect user actions with programm functionalities for displaying and recording gathered data.

Namespaces

- namespace Ui

8.32 mainwindow.h File Reference

```
#include <QMainWindow>
#include <QList>
#include <QPair>
```

Include dependency graph for mainwindow.h: This graph shows which files directly or indirectly include this file:

Classes

- class MainWindow

  The MainWindow class creates the GUI and connect user actions with programm functionalities for displaying and recording gathered data.

Namespaces

- namespace cv
- namespace Ui

Variables

- LogParams logParams

8.32.1 Variable Documentation

8.32.1.1 LogParams logParams

8.33 mapdata.cpp File Reference

```
#include "mapdata.h"
#include "Data/define.h"
#include "Sensor/cam.h"
#include "Structs/position.h"
#include "Sensor/orientierung.h"
#include "AI/pathplanning.h"
#include <QDebug>
```

Include dependency graph for mapdata.cpp:

8.34 mapdata.h File Reference

```
#include "Data/obstacle.h"
#include <referee.h>
#include <QList>
#include <QMutex>
```

Include dependency graph for mapdata.h: This graph shows which files directly or indirectly include this file:

Classes

- class MapData

  The MapData class is a static class for inter-thread communication and saving information for other parts of the programm.

8.35 medianfilter.cpp File Reference

```
#include "medianfilter.h"
#include <algorithm>
#include <QDebug>
```

Include dependency graph for medianfilter.cpp:

8.36 medianfilter.h File Reference

```
#include <QVector>
```

Include dependency graph for medianfilter.h: This graph shows which files directly or indirectly include this file:

Classes

- class MedianFilter

  The MedianFilter class will filter data with an median filter which will return the centered value of a given set.

8.37 medianfilter_new.cpp File Reference

```
#include <memory.h>
#include "medianfilter_new.h"
```
Include dependency graph for medianfilter_new.cpp:

Namespaces

- namespace Filter

Functions

- void Filter::_medianfilter (const element *signal, element *result, int N)
- void Filter::medianfilter (element *signal, element *result, int N)
- void Filter::_medianfilter (const element *image, element *result, int N, int M)
- void Filter::medianfilter (element *image, element *result, int N, int M)

8.38 medianfilter_new.h File Reference

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

Namespaces

- namespace Filter

Typedefs

- typedef double Filter::element

Functions

- void Filter::medianfilter (element *signal, element *result, int N)
- void Filter::medianfilter (element *image, element *result, int N, int M)

8.39 obstacle.cpp File Reference

```
#include "obstacle.h"
#include "Data/define.h"
```
Include dependency graph for obstacle.cpp:

8.40 obstacle.h File Reference

```
#include "Structs/position.h"
#include <QPair>
#include <QTime>
```
Include dependency graph for obstacle.h: This graph shows which files directly or indirectly include this file:

Classes

- class Obstacle

  The Obstacle class describes all objects on field as obstacles, which can be distinguish by type.

Enumerations

- enum FieldArea {
  OUT_OF_AREA = 0, MY_AREA, MY_GOAL_AREA, MY_BEHIND_GOAL_AREA,
  ENEMY_AREA, ENEMY_GOAL_AREA, ENEMY_BEHINDGOAL_AREA, NEUTRAL_AREA,
  POLE_AREA, ENEMY_GOAL_INFLUENCE, GOAL_ZONE_LEFT, GOAL_ZONE_MID,
  GOAL_ZONE_RIGHT }

  The FieldArea enum defines the different areas of the field.
- enum ObstacleColor {
  NOMODIFY = 0, UNIDENTIFIED, POLE, ME,
  OPPONENT }

  The ObstacleColor enum will represent the color of the enum fields.
- enum ObstacleType {
  NOMODIFY = 0, UNIDENTIFIED, DUMMY, POLE,
  PUCK, OPPONENT, ME, TARGET }

  The ObstacleType enum which represents the obstacleType.
- enum ObstacleStatus {
  NOMODIFY = 0, UNDEFINED, UNBLOCKED, ISMOVING,
  BLOCKED, INMYGOAL, INENEMYGOAL }

  The ObstacleStatus enum.

8.40.1 Enumeration Type Documentation

8.40.1.1 enum FieldArea

The FieldArea enum defines the different areas of the field.

Enumerator:

OUT_OF_AREA position is not in the game area

MY_AREA Position is on my side of the field

MY_GOAL_AREA Position is in my goal area

MY_BEHIND_GOAL_AREA Position is behind my goal area

ENEMY_AREA Position is on the enemy side of the field

ENEMY_GOAL_AREA Position is in the enemy goal area

ENEMY_BEHINDGOAL_AREA Position is behind the enemy goal area

NEUTRAL_AREA Position is in the neutral area of the field

POLE_AREA Position in the area around the poles

ENEMY_GOAL_INFLUENCE Position in the area around the Goal

GOAL_ZONE_LEFT checking if an obstacle lies within a certain goal zone

GOAL_ZONE_MID checking if an obstacle lies within a certain goal zone

GOAL_ZONE_RIGHT checking if an obstacle lies within a certain goal zone

Definition at line 12 of file obstacle.h.

8.40.1.2 enum ObstacleColor

The ObstacleColor enum will represent the color of the enum fields.

Enumerator:

NOMODIFY enum which wont modified

UNIDENTIFIED If color is unidentified

POLE pole color

ME own color

OPPONENT foe color

Definition at line 32 of file obstacle.h.

8.40.1.3 enum ObstacleStatus

The ObstacleStatus enum.

Enumerator:

NOMODIFY enum which wont modified

UNDEFINED if the status is undefined

UNBLOCKED if status is unblocked

ISMOVING if the puck is moved by the robot

BLOCKED if status is blocked

INMYGOAL if puck is in my own goal

INENEMYGOAL if puck is in my enemy goal

Definition at line 57 of file obstacle.h.

8.40.1.4 enum ObstacleType

The ObstacleType enum which represents the obstacleType.

Enumerator:

NOMODIFY enum which wont modified

UNIDENTIFIED unidentified obstacle type

DUMMY dummy type for testing purpose

POLE pole type

PUCK puck type

OPPONENT type for our foe

ME type for own robot

TARGET type for target

Definition at line 43 of file obstacle.h.

8.41 orientierung.cpp File Reference

```
#include "orientierung.h"
#include <QtCore>
#include <QDebug>
#include <QLineF>
#include "Data/define.h"
#include "Data/mapdata.h"
#include "Structs/position.h"
#include "Sensor/trilateration.h"
```

Include dependency graph for orientierung.cpp:

8.42 orientierung.h File Reference

```
#include <QVector>
```

Include dependency graph for orientierung.h: This graph shows which files directly or indirectly include this file:

Classes

- class Orientation

  The Orientation class try to compute the position and the orientation of the robot due to distance values and angles from sensor data.

8.43 pathplanning.cpp File Reference

```
#include "pathplanning.h"
#include "Data/define.h"
#include <QtConcurrent/QtConcurrentMap>
#include <QDebug>
#include <QElapsedTimer>
#include <iostream>
#include "Data/mapdata.h"
```

Include dependency graph for pathplanning.cpp:

Functions

- double round (double r)

  round helper function for integer rounding

8.43.1 Function Documentation

8.43.1.1 double round (double *r*)

round helper function for integer rounding

Parameters

r	

Returns

the next integer value

Definition at line 23 of file pathplanning.cpp.

8.44 pathplanning.h File Reference

```
#include <QList>
#include <QPair>
#include <atomic>
#include "Data/obstacle.h"
#include "Structs/pathplotdata.h"
```

Include dependency graph for pathplanning.h: This graph shows which files directly or indirectly include this file:

Classes

- class PathPlanning
- class PathPlanning::GridPoint

8.45 pathplotdata.h File Reference

```
#include <QVector>
#include <QPair>
#include <QObject>
```

Include dependency graph for Plots/pathplotdata.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PathPlotData

  The PathPlotData struct is the data holder for plotting the path.
- struct PathPlotData::Point

  The Point struct represent Waypoints data coming from PathPlanning.

8.46 pathplotdata.h File Reference

```
#include <QVector>
#include <QPair>
#include <QObject>
```

Include dependency graph for Structs/pathplotdata.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PathPlotData

  The PathPlotData struct is the data holder for plotting the path.
- struct PathPlotData::Point

  The Point struct represent Waypoints data coming from PathPlanning.

8.47 pathRealizer.h File Reference

```
#include <QTimer>
#include <QMutex>
#include <atomic>
#include "Data/obstacle.h"
#include "Plots/pathplotdata.h"
#include "Plots/pidplotdata.h"
#include "Actor/pidparams.h"
#include "Actor/spline.h"
```

Include dependency graph for pathRealizer.h:

Classes

- class PathRealizer

  The PathRealizer class This class is responsible for realising a given path (array of points) It is required, that the robot did not drive backwards because it has not any sensors on its back. The internal state-machine is called with an heartbeat signal (TIMER)

8.48 pidparams.h File Reference

```
#include <QObject>
```

Include dependency graph for Actor/pidparams.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PIDParams

  The PIDParams struct, values for the PID controler for angular PID and velocity PID.

8.49 pidparams.h File Reference

```
#include <QObject>
```

Include dependency graph for Structs/pidparams.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PIDParams

  The PIDParams struct, values for the PID controler for angular PID and velocity PID.

8.50 pidplotdata.h File Reference

```
#include <QList>
#include <QObject>
```

Include dependency graph for Plots/pidplotdata.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PIDPlotData

  The PIDPlotData struct represents the data of the PID controler of the last n-time steps.

8.51 pidplotdata.h File Reference

```
#include <QList>
#include <QObject>
```

Include dependency graph for Structs/pidplotdata.h: This graph shows which files directly or indirectly include this file:

Classes

- struct PIDPlotData

  The PIDPlotData struct represents the data of the PID controler of the last n-time steps.

8.52 player.cpp File Reference

```
#include "player.h"
#include <string>
#include <unistd.h>
#include <QMutex>
#include <QMutexLocker>
#include <QGlobalStatic>
#include <QDebug>
#include <QElapsedTimer>
#include <QThread>
#include <libplayerc++/playerc++.h>
```

Include dependency graph for player.cpp:

8.53 player.h File Reference

```
#include <string>
```

Include dependency graph for player.h: This graph shows which files directly or indirectly include this file:

Classes

- class PlayerX

  The Player class this class contains the instance of the player client to access in 'global' scope.

Namespaces

- namespace PlayerCc

8.54 position.cpp File Reference

```
#include "position.h"
#include "Data/define.h"
```

Include dependency graph for position.cpp:

8.55 position.h File Reference

```
#include <QObject>
#include <QPair>
```

Include dependency graph for position.h: This graph shows which files directly or indirectly include this file:

Classes

- class Position

  The Position struct will represent the current pose of the robot.

Enumerations

- enum SizeType { UNKNOWN, ROBOT, PUCK, POLE }

  The SizeType enum.

8.55.1 Enumeration Type Documentation

8.55.1.1 enum SizeType

The SizeType enum.

Enumerator:

UNKNOWN

ROBOT

PUCK

POLE

Definition at line 10 of file position.h.

8.56 qcustomplot.cpp File Reference

```
#include "qcustomplot.h"
```

Include dependency graph for qcustomplot.cpp:

8.56.1 Detailed Description

Definition in file qcustomplot.cpp.

8.57 qcustomplot.h File Reference

```
#include <QObject>
#include <QPointer>
#include <QWidget>
#include <QPainter>
#include <QPaintEvent>
#include <QMouseEvent>
#include <QPixmap>
#include <QVector>
#include <QString>
#include <QDateTime>
#include <QMultiMap>
#include <QFlags>
#include <QDebug>
#include <QVector2D>
#include <QStack>
#include <QCache>
#include <QMargins>
#include <qmath.h>
#include <limits>
#include <QtNumeric>
#include <QtPrintSupport>
```

Include dependency graph for qcustomplot.h: This graph shows which files directly or indirectly include this file:

Functions

- Q_DECLARE_TYPEINFO (QCPScatterStyle, Q_MOVABLE_TYPE)
- Q_DECLARE_TYPEINFO (QCPRange, Q_MOVABLE_TYPE)
- const QCPRange operator+ (const QCPRange &range, double value)
- const QCPRange operator+ (double value, const QCPRange &range)
- const QCPRange operator- (const QCPRange &range, double value)
- const QCPRange operator∗ (const QCPRange &range, double value)
- const QCPRange operator∗ (double value, const QCPRange &range)
- const QCPRange operator/ (const QCPRange &range, double value)
- Q_DECLARE_TYPEINFO (QCPLineEnding, Q_MOVABLE_TYPE)
- Q_DECLARE_TYPEINFO (QCPData, Q_MOVABLE_TYPE)
- Q_DECLARE_TYPEINFO (QCPCurveData, Q_MOVABLE_TYPE)
- Q_DECLARE_TYPEINFO (QCPBarData, Q_MOVABLE_TYPE)
- Q_DECLARE_TYPEINFO (QCPFinancialData, Q_MOVABLE_TYPE)

8.57.1 Detailed Description

Definition in file qcustomplot.h.

8.57.2 Function Documentation

8.57.2.1 const QCPRange operator∗ (const QCPRange & *range,* double *value*) [inline]

Multiplies both boundaries of the range by *value*.

Definition at line 572 of file qcustomplot.h.

8.57.2.2 const QCPRange operator∗ (double *value*, const QCPRange & *range*) [inline]

Multiplies both boundaries of the range by *value*.

Definition at line 582 of file qcustomplot.h.

8.57.2.3 const QCPRange operator+ (const QCPRange & *range*, double *value*) [inline]

Adds *value* to both boundaries of the range.

Definition at line 542 of file qcustomplot.h.

8.57.2.4 const QCPRange operator+ (double *value*, const QCPRange & *range*) [inline]

Adds *value* to both boundaries of the range.

Definition at line 552 of file qcustomplot.h.

8.57.2.5 const QCPRange operator- (const QCPRange & *range*, double *value*) [inline]

Subtracts *value* from both boundaries of the range.

Definition at line 562 of file qcustomplot.h.

8.57.2.6 const QCPRange operator/ (const QCPRange & *range*, double *value*) [inline]

Divides both boundaries of the range by *value*.

Definition at line 592 of file qcustomplot.h.

8.57.2.7 Q_DECLARE_TYPEINFO (QCPScatterStyle , Q_MOVABLE_TYPE)

8.57.2.8 Q_DECLARE_TYPEINFO (QCPRange , Q_MOVABLE_TYPE)

8.57.2.9 Q_DECLARE_TYPEINFO (QCPLineEnding , Q_MOVABLE_TYPE)

8.57.2.10 Q_DECLARE_TYPEINFO (QCPData , Q_MOVABLE_TYPE)

8.57.2.11 Q_DECLARE_TYPEINFO (QCPCurveData , Q_MOVABLE_TYPE)

8.57.2.12 Q_DECLARE_TYPEINFO (QCPBarData , Q_MOVABLE_TYPE)

8.57.2.13 Q_DECLARE_TYPEINFO (QCPFinancialData , Q_MOVABLE_TYPE)

8.58 referee.cpp File Reference

```
#include "referee.h"
#include "hermescodes.h"
#include "hermes.h"
```

Include dependency graph for referee.cpp:

8.59 referee.h File Reference

```
#include <QObject>
```

```
#include <QMap>
```
Include dependency graph for referee.h: This graph shows which files directly or indirectly include this file:

Classes

- class Referee

  Die Schiedsrichterklasse.

Enumerations

- enum TeamColor { yellow, blue }

  Enum dient zur Angabe der Farbe des eigenen Teams.

8.59.1 Enumeration Type Documentation

8.59.1.1 enum TeamColor

Enum dient zur Angabe der Farbe des eigenen Teams.

Enumerator:

yellow

blue

Definition at line 36 of file referee.h.

8.60 robotThread.cpp File Reference

```
#include "robotThread.h"
#include <QDebug>
#include <QThread>
#include <QCoreApplication>
#include "GUI/mainwindow.h"
#include "Data/define.h"
#include "player.h"
#include "Actor/actorLowLevel.h"
#include "Actor/actorhighlevel.h"
#include "Sensor/sensorhighlevel.h"
#include "Sensor/sensorLowLevel.h"
#include "AI/pathplanning.h"
#include "AI/gameengine.h"
#include "Structs/pathplotdata.h"
#include "Sensor/cam.h"
#include "AI/game.h"
```
Include dependency graph for Main/robotThread.cpp:

8.61 robotThread.cpp File Reference

```
#include "robotThread.h"
#include "mainwindow.h"
#include <QDebug>
#include "Data/define.h"
#include "player.h"
#include "Actor/actorLowLevel.h"
#include "Actor/actorhighlevel.h"
#include "Sensor/sensorhighlevel.h"
#include "Sensor/sensorLowLevel.h"
#include "AI/pathplanning.h"
#include "AI/gameengine.h"
#include "Structs/pathplotdata.h"
#include "Sensor/cam.h"
```

Include dependency graph for robotThread.cpp:

Variables

- MainWindow * mainWindow

8.61.1 Variable Documentation

8.61.1.1 MainWindow* mainWindow

8.62 robotThread.h File Reference

```
#include <QThread>
```

Include dependency graph for Main/robotThread.h: This graph shows which files directly or indirectly include this file:

Classes

- class RobotThread

  The RobotThread class is responsible for the communication of all classes and is the software representation of the robot, all threads are forked there and will joinen in the end. The class will move diverent tasks to different threads.

8.63 robotThread.h File Reference

```
#include <QThread>
```

Include dependency graph for robotThread.h: This graph shows which files directly or indirectly include this file:

Classes

- class RobotThread

  The RobotThread class is responsible for the communication of all classes and is the software representation of the robot, all threads are forked there and will joinen in the end. The class will move diverent tasks to different threads.

8.64 sensor.h File Reference

```
#include <QObject>
#include <QPair>
#include <QVector>
#include <QVector3D>
#include <QMutex>
#include <QElapsedTimer>
#include <atomic>
#include "Data/obstacle.h"
#include "Plots/LaserPlotData.h"
```

Include dependency graph for sensor.h:

Classes

- struct FilterParams
- class SensorHighLevel

  The SensorHighLevel class is responsible for the processing of the raw laser data and adds all objects to the MapData.

Enumerations

- enum SensorStates {
  WAIT, ORIENTATION, RECOGNITION, ORIENTATION_VALIDATION,
  COLOR_DETECTION_START, COLOR_DETECTION_WAIT, WAIT, ORIENTATION,
  RECOGNITION, PRE_GAME_STATE, ORIENTATION_VALIDATION, COLOR_DETECTION_START,
  COLOR_DETECTION_WAIT }

8.64.1 Enumeration Type Documentation

8.64.1.1 enum SensorStates

Enumerator:

WAIT

ORIENTATION

RECOGNITION

ORIENTATION_VALIDATION

COLOR_DETECTION_START

COLOR_DETECTION_WAIT

WAIT

ORIENTATION

RECOGNITION

PRE_GAME_STATE

ORIENTATION_VALIDATION

COLOR_DETECTION_START

COLOR_DETECTION_WAIT

Definition at line 25 of file sensor.h.

8.65 sensorhighlevel.cpp File Reference

```
#include "sensorhighlevel.h"
#include <QtMath>
#include <QDebug>
#include <QtAlgorithms>
#include <QMutexLocker>
#include <QMutex>
#include <QElapsedTimer>
#include <QThread>
#include "Data/define.h"
#include "Data/mapdata.h"
#include "Sensor/orientierung.h"
#include "Sensor/medianfilter.h"
#include "Sensor/medianfilter_new.h"
#include "Sensor/cam.h"
#include "AI/pathplanning.h"
```

Include dependency graph for sensorhighlevel.cpp:

8.66 sensorhighlevel.h File Reference

```
#include <QObject>
#include <QPair>
#include <QVector>
#include <QList>
#include <atomic>
#include "Structs/position.h"
#include "Structs/laserplotdata.h"
#include "Structs/filterparams.h"
#include "Structs/constrainedlaserdata.h"
```

Include dependency graph for sensorhighlevel.h: This graph shows which files directly or indirectly include this file:

Classes

- class SensorHighLevel

  The SensorHighLevel class is responsible for the processing of the raw laser data and adds all objects to the MapData.

Enumerations

- enum SensorStates {
  WAIT, ORIENTATION, RECOGNITION, ORIENTATION_VALIDATION,
  COLOR_DETECTION_START, COLOR_DETECTION_WAIT, WAIT, ORIENTATION,
  RECOGNITION, PRE_GAME_STATE, ORIENTATION_VALIDATION, COLOR_DETECTION_START,
  COLOR_DETECTION_WAIT }

  The SensorStates enum containing all needed states oth the HighLevelSensor.

8.66.1 Enumeration Type Documentation

8.66.1.1 enum SensorStates

The SensorStates enum containing all needed states oth the HighLevelSensor.

Enumerator:

WAIT

ORIENTATION

RECOGNITION

ORIENTATION_VALIDATION

COLOR_DETECTION_START

COLOR_DETECTION_WAIT

WAIT

ORIENTATION

RECOGNITION

PRE_GAME_STATE

ORIENTATION_VALIDATION

COLOR_DETECTION_START

COLOR_DETECTION_WAIT

Definition at line 24 of file sensorhighlevel.h.

8.67 sensorLowLevel.cpp File Reference

```
#include "sensorLowLevel.h"
#include <QDebug>
#include <QThread>
#include <QCoreApplication>
#include <libplayerc++/playerc++.h>
#include "Structs/laserplotdata.h"
#include "Data/define.h"
#include "Data/mapdata.h"
#include "Sensor/sensorhighlevel.h"
#include "Main/player.h"
```

Include dependency graph for sensorLowLevel.cpp:

8.68 sensorLowLevel.h File Reference

```
#include <QVector>
#include <QObject>
#include <QElapsedTimer>
#include <atomic>
#include "Structs/laserplotdata.h"
#include "Structs/position.h"
```

Include dependency graph for sensorLowLevel.h: This graph shows which files directly or indirectly include this file:

Classes

- class SensorLowLevel

  The SensorLowLevel class is collecting odometry and laser data from the player client.

Namespaces

- namespace PlayerCc

8.69 spline.cpp File Reference

```
#include "spline.h"
#include <QtGlobal>
#include <QtAlgorithms>
#include <cstdio>
#include <cassert>
```

Include dependency graph for spline.cpp:

Namespaces

- namespace tkqt

8.70 spline.h File Reference

```
#include <QVector>
```

Include dependency graph for spline.h: This graph shows which files directly or indirectly include this file:

Classes

- class tkqt::band_matrix

  The band_matrix class, which is the basis for cubic hermite spline creation.
- class tkqt::spline

  The spline class => This class will create an cubic hermite spline from two given vectors.

Namespaces

- namespace tkqt

8.71 trilateration.cpp File Reference

```
#include "trilateration.h"
#include <cmath>
```

Include dependency graph for trilateration.cpp:

Namespaces

- namespace trilateration

Functions

- int trilateration::circle_circle_intersection (double x0, double y0, double r0, double x1, double y1, double r1, double *xi, double *yi, double *xi_prime, double *yi_prime)

  circle_circle_intersection: Determine the points where 2 circles in a common plane intersect.

8.72 trilateration.h File Reference

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

Namespaces

- namespace trilateration

Functions

- int trilateration::circle_circle_intersection (double x0, double y0, double r0, double x1, double y1, double r1, double *xi, double *yi, double *xi_prime, double *yi_prime)

  circle_circle_intersection: Determine the points where 2 circles in a common plane intersect.

Index

~ActorHighLevel
ActorHighLevel, 37
~ActorLowLevel
ActorLowLevel, 45
~Cam
Cam, 52
~ConstrainedLaserData
ConstrainedLaserData, 57
~Game
Game, 65
~GameEngine
GameEngine, 72
~Hermes
Hermes, 80
~MainWindow
MainWindow, 92
~MapData
MapData, 114
~Obstacle
Obstacle, 129
~Orientation
Orientation, 136
~PathPlanning
PathPlanning, 141
~PathRealizer
PathRealizer, 153
~Referee
Referee, 171
~RobotThread
RobotThread, 176
~SensorHighLevel
SensorHighLevel, 182
~SensorLowLevel
SensorLowLevel, 191
~band_matrix
tkqt::band_matrix, 49
_USE_MATH_DEFINES
define.h, 205
_medianfilter
Filter, 31
_qimage
CVImageWidget, 61
_tmp
CVImageWidget, 61

ANNOY_FOE
Game, 65
AVERAGE
LaserPlotData, 81
abValues
Referee, 172
active
PathPlanning::GridPoint, 78
actorDistanceOfTargetOnSpline
config, 16
actorGatherPuckDistance
config, 16
ActorHighLevel, 35
~ActorHighLevel, 37
ActorHighLevel, 37
ActorHighLevel, 37
additionalReleasePuckDistance, 41
constrainAngle, 38
elapsedTime, 41
enabled, 41
getState, 38
iDeltaA, 41
iDeltaL, 41
ignoreSignals, 38
internalWP, 41
lastDeltaA, 41
lastDeltaL, 41
lowPass, 38
mutexPidHist, 41
mutexState, 42
numWP, 42
PID_A_D, 42
PID_A_I, 42
PID_A_P, 42
PID_V_D, 42
PID_V_I, 42
PID_V_P, 42
pidHistDistIst, 42
pidHistDistSoll, 43
pidHistTime, 43
pidHistWinkelIst, 43
pidHistWinkelSoll, 43
positionWasReached, 43
quitting, 43
releasePuckOrigin, 43
resetPIDtempVars, 38
robotPosition, 43
setState, 38
signalPIDPlot, 39
signalPuckDone, 39
signalSendRobotControlParams, 39
signalSplinePlot, 39
slotChangePIDParams, 39
slotGatherPuck, 39

slotPushAndReleasePuck, 39
slotReleasePuck, 40
slotTimerSendPIDPlot, 40
slotUpdateWaypoints, 40
splineX, 43
splineY, 43
startPIDController, 40
state, 44
streamPIDEnabled, 44
takeDimension, 40
targetDistDiffLast, 44
targetDistLast, 44
timeOfStart, 44
timerPIDPlot, 44
actorHighLevel
RobotThread, 177
ActorLowLevel, 44
∼ActorLowLevel, 45
ActorLowLevel, 45
ActorLowLevel, 45
isRefereeEmergency, 46
moveForward, 47
moveRobot, 45
positionProxy, 47
previousTurnRate, 47
previousVelocity, 47
setOdometry, 46
setRobotControllParams, 46
setRobotRemoteControllParams, 46
slotEmergencyStopEnabled, 46
tempTurnRate, 47
tempVelocity, 47
actorLowLevel
RobotThread, 177
actorLowLevel.cpp, 198
actorLowLevel.h, 198
actorLowPass
config, 16
actorMaxAngleLimiter
config, 16
actorMaxI
config, 17
actorMinAngleLimiter
config, 17
actorPeriodMotionControl
config, 17
actorPushAndReleaseAdditionalReverseDist
config, 17
actorPushPuckDistance
config, 17
actorReleasePuckDistance
config, 17
actorWPLowPassAlpha
config, 17
actorWaypointMaxAngleDeviation
config, 17
actorWaypointReachedDiffChange
config, 17
actorWaypointReachedDistance
config, 17
actorhighlevel.h
GATHER_PUCK, 198
PUSH_AND_RELEASE_PUCK, 198
RELEASE_PUCK, 198
RELEASE_PUCK_FROM_PUSH, 198
RUNNING, 198
STOP, 198
actorhighlevel.cpp, 197
actorhighlevel.h, 197
StatePathProcessing, 198
addAngle
ConstrainedLaserData, 57
addFilteredDepth
ConstrainedLaserData, 58
addRawDepth
ConstrainedLaserData, 58
additionalReleasePuckDistance
ActorHighLevel, 41
angleBetweenAB
Orientation, 136
angleWeightedAverage
SensorLowLevel, 191
angles
ConstrainedLaserData, 58
LaserPlotData, 82
annoyFoeState
Game, 65
avoidRestOfField
PathPlanning, 144
avoideCollision
SensorHighLevel, 182

BLOCKED
obstacle.h, 217
BLUE
cam.h, 199
bInitialized
Obstacle, 134
BORDERSIDE_CM
config, 18
BORDERTOP_CM
config, 18
back
MainWindow, 92
band_matrix
tkqt::band_matrix, 48
beginOrientation
Orientation, 136
blue
referee.h, 225

COLOR_DETECTION_START
sensor.h, 227
sensorhighlevel.h, 229
COLOR_DETECTION_WAIT
sensor.h, 227
sensorhighlevel.h, 229

CRITICAL
LogParams, 84, 85
cLastUpdate
Obstacle, 134
cPolePositions
SensorHighLevel, 187
cPosition
Obstacle, 134
CVImageWidget, 59
_qimage, 61
_tmp, 61
CVImageWidget, 60
CVImageWidget, 60
minimumSizeHint, 60
paintEvent, 60
showImage, 60
sizeHint, 61
calculateIntrinsicCost
PathPlanning::GridPoint, 77
calculateObjCenter
SensorHighLevel, 182
calculatePathCosts
PathPlanning, 141
calculateWaypoints
PathPlanning, 141
Cam, 51
∼Cam, 52
Cam, 52
color, 54
cp, 54
enabled, 54
getLastColor, 53
getPixelColor, 53
grabFrameAndColor, 53
mutexVideoCapture, 54
signalColorDetected, 53
signalDisplayFrame, 53
slotSetCameraParams, 53
slotStartColorDetection, 53
streamCamEnabled, 54
timer, 54
timerSendFrame, 54
videoCapture, 54
cam
RobotThread, 177
cam.h
BLUE, 199
ERROR, 199
GREEN, 199
NONE, 199
YELLOW, 199
cam.cpp, 199
cam.h, 199
CamColor, 199
CamColor
cam.h, 199
CameraParams, 55
colorBlueMax, 55
colorBlueMin, 55
colorGreenMax, 55
colorGreenMin, 55
colorYellowMax, 56
colorYellowMin, 56
height, 56
source, 56
updatePeriod, 56
width, 56
cameraparams.h, 200
certainty
Position, 166
changeCamParams
MainWindow, 92
changeFilterParams
MainWindow, 92
changeLogParams
MainWindow, 92
changePIDParams
MainWindow, 93
checkForEnemyNearTraget
MapData, 114
checkObjectsOnLine
Orientation, 136
checkTargetPuckAvailable
Game, 66
circle_circle_intersection
trilateration, 32
cleanup
MapData, 114
clearData
ConstrainedLaserData, 58
clearTargets
MainWindow, 93
MapData, 114
collectorObj
SensorHighLevel, 186
color
Cam, 54
colorBlueMax
CameraParams, 55
colorBlueMin
CameraParams, 55
colorGreenMax
CameraParams, 55
colorGreenMin
CameraParams, 55
colorMap
MainWindow, 108
colorYellowMax
CameraParams, 56
colorYellowMin
CameraParams, 56
colorfail
Game, 69
colors
MainWindow, 108
compareObstacleTimestamps

MapData, 114
config, 11
actorDistanceOfTargetOnSpline, 16
actorGatherPuckDistance, 16
actorLowPass, 16
actorMaxAngleLimiter, 16
actorMaxI, 17
actorMinAngleLimiter, 17
actorPeriodMotionControl, 17
actorPushAndReleaseAdditionalReverseDist, 17
actorPushPuckDistance, 17
actorReleasePuckDistance, 17
actorWPLowPassAlpha, 17
actorWaypointMaxAngleDeviation, 17
actorWaypointReachedDiffChange, 17
actorWaypointReachedDistance, 17
BORDERSIDE_CM, 18
BORDERTOP_CM, 18
DUMP_SLOT_2_3, 19
DUMP_SLOT_3_4, 19
DUMP_SLOT_4_5, 19
DUMP_SLOT_5_6, 19
enableDebugActorHighLevel, 19
enableDebugActorLowLevel, 19
enableDebugCam, 19
enableDebugGame, 19
enableDebugMainwindow, 19
enableDebugMapData, 20
enableDebugOrientation, 20
enableDebugPathPlanning, 20
enableDebugSensorHighLevel, 20
enableDebugSensorLowLevel, 20
FIELDHEIGHT_CM, 20
FIELDWIDTH_CM, 20
GUIREMOTETURNRATE, 23
GUIREMOTEVELOCITY, 23
GUIopponent, 23
GUIpuckOpponent, 23
GUIpuckSelf, 23
GUIpuckUndef, 23
GUIself, 23
GUItarget, 24
gameAnnoyPositionL, 15
gameAnnoyPositionR, 15
gameBisZuWelchemAbstandWirdZielwackeln-Gefiltert, 20
gameGoalSlotL, 15
gameGoalSlotM, 15
gameGoalSlotOutsideLeft, 15
gameGoalSlotOutsideRight, 15
gameGoalSlotR, 15
gameMinimumDistanceEnemyToDumpSlot, 20
gameMinimumDistanceToEnemyRobot, 21
gameWieWeitMussDerGegnerVomZielEntfernt-SeinImAnnoyModus, 21
gameWievielBesserMussEinPuckSeinUmDasZiel-ZuWechseln, 21
gameZielwackelfilterTiefpassKoeffizient, 21
geoFieldHeight, 21
geoFieldWidth, 21
geoGoalHeight, 21
geoGoalMarginBottom, 21
geoGoalMarginSide, 21
geoGoalWidth, 21
geoPol_1_14, 22
geoPol_1_2, 22
geoPol_2_3, 22
geoPol_3_4, 22
geoPoleRadiusReal, 22
geoPoleRadiusSim, 22
geoPuckRadiusBottom, 22
geoPuckRadiusTopReal, 22
geoPuckRadiusTopSim, 22
geoRobotForkCenterDist, 23
guiBrushGoalBlue, 15
guiBrushGoalUndef, 15
guiBrushGoalYellow, 15
guiBrushMe, 15
guiBrushPole, 15
guiBrushPuckBlocked, 15
guiBrushPuckMoving, 15
guiPenDummy, 15
guiPenFieldPrimary, 15
guiPenFieldSecondary, 16
guiPenMe, 16
guiPenOpponent, 16
guiPenOrient, 16
guiPenPole, 16
guiPenPuckMe, 16
guiPenPuckMeOuter, 16
guiPenPuckOpponent, 16
guiPenPuckOpponentOuter, 16
guiPenPuckUndef, 16
guiPenPuckUndefOuter, 16
guiPenTarget, 16
guiPuckKlickTolerance, 23
mapAbstandRoboterZentrumZuGabel, 24
mapIgnorePuckInsideEnemyDistance, 24
mapPolePuckFusionDistance, 24
mapPuckPuckFusionDistance, 24
mapToleranzBisWohinEinPuckInDerGabelIstMAX, 24
mapToleranzBisWohinEinPuckInDerGabelIstMIN, 24
obstacleCoordinateTolerance, 24
obstacleNumberOfPoles, 24
obstacleNumberOfPucks, 25
PATH_SHOWRES, 25
POLESIZE_CM, 28
PUCKSIZE_INNER_CM, 28
PUCKSIZE_OUTER_CM, 28
pathAdjacencyMultiplier, 25
pathArenaFieldAvoidMaxY, 25
pathArenaMaxX, 25
pathArenaMaxY, 25
pathArenaMinX, 25

pathArenaMinY, 26
pathEnemyCloseCost, 26
pathEnemyCloseDist, 26
pathEnemyFarDist, 26
pathEnemyStartCost, 26
pathGridSpacingBase, 26
pathMaxWPIterations, 26
pathPlanningEnabledUpwardsOfThisDistance, 26
pathPoleCloseCost, 26
pathPoleCloseDist, 27
pathPoleFarDist, 27
pathPoleStartCost, 27
pathPuckCloseCost, 27
pathPuckCloseDist, 27
pathPuckFarDist, 27
pathPuckStartCost, 27
pathRobotRadius, 27
pathTargetApproachAngleFullCost, 27
pathTargetApproachAngleInfluenceDistance, 27
pathTargetApproachAngleMaxDeviationWithoutFullCost, 28
periodAlive, 28
periodEgoPos, 28
periodTillAnnoy, 28
puckIsCloseToPoleDistance, 28
ROBOSIZE_CM, 29
refIP, 28
refPort, 29
refVerbose, 29
SENSOR_COLLISION_AT, 29
SENSOR_DELTA_ANGLE, 29
SENSOR_RADIUS_ROBOT, 30
SENSOR_WAIT_COUNTER, 30
TARGET_POLE_VARIANCE, 30
TARGETSIZE_CM, 30
teamID, 30
connectFailed
Referee, 172
connectToServer
Referee, 172
connected
Referee, 172
connection
Referee, 174
constrainAngle
ActorHighLevel, 38
PathPlanning::GridPoint, 77
PathRealizer, 153
constrainData
SensorHighLevel, 183
constrainedData
SensorHighLevel, 186
ConstrainedLaserData, 56
~ConstrainedLaserData, 57
addAngle, 57
addFilteredDepth, 58
addRawDepth, 58
angles, 58
clearData, 58
ConstrainedLaserData, 57
ConstrainedLaserData, 57
filteredDepths, 58
m_angles, 59
m_filteredDepth, 59
m_rawDepths, 59
rawDepths, 58
constrainedlaserdata.cpp, 200
constrainedlaserdata.h, 200
convertPolToGlobalCoordinates
SensorHighLevel, 183
convertX
MainWindow, 93
convertY
MainWindow, 93, 94
counter
SensorHighLevel, 187
cp
Cam, 54
createTargetInfrontPuck
Game, 66
currentObjects
SensorHighLevel, 187
currentOdometryPosition
SensorLowLevel, 192
currentOdometryTime
SensorLowLevel, 193
currentPosition
SensorHighLevel, 187
currentPuck
Game, 69
currentPuckPrio
Game, 69
currentState
SensorHighLevel, 187
customLogger
Log, 83
cv, 30
cvimagewidget.h, 200, 201

DEBUG
LogParams, 84, 85
DETECTION
GameEngine, 72
DRIVE_2_SLOT
Game, 65
DRIVE_LEFT
Game, 65
DRIVE_RIGHT
Game, 65
DRIVE_TO_DUMP
Game, 65
DRIVE_TO_GOAL
Game, 64
DRIVE_TO_PUCK
Game, 64
DUMMY
obstacle.h, 217

DUMP_SLOT_2_3
config, 19
DUMP_SLOT_3_4
config, 19
DUMP_SLOT_4_5
config, 19
DUMP_SLOT_5_6
config, 19
data
LaserPlotData, 82
PathPlotData, 149
dataSizeX
PathPlotData, 149
dataSizeY
PathPlotData, 149
dataType
LaserPlotData, 82
PathPlotData, 149
DataTypeEnum
LaserPlotData, 81
PathPlotData, 148, 149
define.h, 201
_USE_MATH_DEFINES, 205
deleteFirstTarget
MapData, 115
deleteObstacle
MapData, 115
detectionStart
Referee, 172
didWeStartPlayerOurselves
PlayerX, 163
dim
tkqt::band_matrix, 49
disableEmergency
MapData, 121
disconnected
Referee, 172
distanceKartesisch
SensorHighLevel, 183
distancePolar
Orientation, 137
SensorHighLevel, 183
distanzIst
PIDPlotData, 161
distanzSoll
PIDPlotData, 161
drawMap
MainWindow, 94
driveToGoal
Game, 66
driveToPreposition
SensorHighLevel, 183
dummyAngleOffset
SensorHighLevel, 187
dummyPosition
SensorHighLevel, 187
dynsections.js, 205
toggleFolder, 205
toggleInherit, 205
toggleLevel, 205
toggleVisibility, 205
updateStripes, 205

ENEMY_AREA
obstacle.h, 216
ENEMY_BEHINDGOAL_AREA
obstacle.h, 216
ENEMY_GOAL_AREA
obstacle.h, 216
ENEMY_GOAL_INFLUENCE
obstacle.h, 216
ERROR
cam.h, 199
elapsedTime
ActorHighLevel, 41
PathRealizer, 155
elapsedTimer
SensorLowLevel, 193
element
Filter, 31
enableDebugActorHighLevel
config, 19
enableDebugActorLowLevel
config, 19
enableDebugCam
config, 19
enableDebugGame
config, 19
enableDebugMainwindow
config, 19
enableDebugMapData
config, 20
enableDebugOrientation
config, 20
enableDebugPathPlanning
config, 20
enableDebugSensorHighLevel
config, 20
enableDebugSensorLowLevel
config, 20
enabled
ActorHighLevel, 41
Cam, 54
PathPlanning, 144
enumColor
Obstacle, 134
enumStatus
Obstacle, 134
enumType
Obstacle, 134
extractObjects
SensorHighLevel, 183

FATAL
LogParams, 84, 85
FIND_SLOT
Game, 65

FIELDHEIGHT_CM
config, 20
FIELDWIDTH_CM
config, 20
FieldArea
obstacle.h, 216
Filter, 31
_medianfilter, 31
element, 31
medianfilter, 31
filter
MedianFilter, 124
filterParameter
SensorHighLevel, 187
FilterParams, 61
FilterParams, 62
FilterParams, 62
kernel, 62
ObsFilterAnzahl, 62
ObsFilterSchnitt, 62
PosFilterAnzahl, 62
PosFilterSchnitt, 62
filteredDepths
ConstrainedLaserData, 58
filterparams.h, 206
findAndSelectBestPuck
Game, 66
findBestGoalSlot
Game, 66
findBestPuck
Game, 66
findDumpSlot
Game, 67
forward
MainWindow, 94

GAME
GameEngine, 72
GATHER_PUCK
actorhighlevel.h, 198
GATHERING_PUCK
Game, 64
GOAL_ZONE_LEFT
obstacle.h, 216
GOAL_ZONE_MID
obstacle.h, 216
GOAL_ZONE_RIGHT
obstacle.h, 216
GREEN
cam.h, 199
GRID
PathPlanning::GridPoint, 76
GUI/mainwindow.cpp
timer, 212
GUIREMOTETURNRATE
config, 23
GUIREMOTEVELOCITY
config, 23
GUIopponent
config, 23
GUIpuckOpponent
config, 23
GUIpuckSelf
config, 23
GUIpuckUndef
config, 23
GUIself
config, 23
GUItarget
config, 24
Game, 62
∼Game, 65
ANNOY_FOE, 65
annoyFoeState, 65
checkTargetPuckAvailable, 66
colorfail, 69
createTargetInfrontPuck, 66
currentPuck, 69
currentPuckPrio, 69
DRIVE_2_SLOT, 65
DRIVE_LEFT, 65
DRIVE_RIGHT, 65
DRIVE_TO_DUMP, 65
DRIVE_TO_GOAL, 64
DRIVE_TO_PUCK, 64
driveToGoal, 66
FIND_SLOT, 65
findAndSelectBestPuck, 66
findBestGoalSlot, 66
findBestPuck, 66
findDumpSlot, 67
GATHERING_PUCK, 64
Game, 65
gameEngine, 69
GameState, 64
getParkPosition, 67
getPositionToTurnRoboToMiddlePoint, 67
getTargetForPuck, 67
getTargetForPuckBeforePole, 67
goalSlotCounterList, 69
handleEnemyPuck, 67
HandleEnemyPuckState, 65
handleEnemyPuckState, 69
isGameStarted, 69
isPuckAwayFromEnemy, 67
isPuckAwayFromPole, 67
lastTargetedPuck, 69
quit, 68
quitting, 70
RELEASE, 65
run, 68
SELECT_GOAL_SLOT, 64
SELECT_PUCK, 64
signalGatherPuck, 68
signalPuckRelease, 68
signalPushAndRelease, 68
signalReportGoal, 68

signalStartColorDetectAI, 68
slotActorHighLevelIsDoneWithPuck, 68
slotAnnoyFoe, 68
slotColorDetect, 68
slotStartGame, 69
state, 70
subAnnoyState, 70
SubGameStateAnnoy, 65
timerLostPuck, 70
timerUpdateGoalSlot, 70
WAIT_FOR_RELEASE, 65
game
RobotThread, 177
game.cpp, 206
game.h, 206
GameEngine
DETECTION, 72
GAME, 72
INIT, 72
STOP, 72
WAIT_FOR_DETECTION, 72
WAIT_FOR_GAME, 72
gameAnnoyPositionL
config, 15
gameAnnoyPositionR
config, 15
gameBisZuWelchemAbstandWirdZielwackelnGefiltert
config, 20
GameEngine, 70
∼GameEngine, 72
GameEngine, 72
GameEngine, 72
getState, 72
referee, 75
signalAnnoyFoe, 72
signalEmergencyStopEnabled, 72
signalStartDetection, 73
signalStartGame, 73
slotDetectionFinished, 73
slotRefConnectFailed, 73
slotRefConnected, 73
slotRefDetectionStart, 73
slotRefDisconnected, 73
slotRefGameOver, 74
slotRefGameStart, 74
slotRefStopMovement, 74
slotRefTrueColorOfTeam, 74
slotReportGoal, 74
slotTimerAlive, 74
slotTimerAnnoy, 74
slotTimerEgoPos, 74
startGameEngine, 74
state, 75
StateNameEnum, 72
timerAlive, 75
timerEgoPos, 75
timerTillAnnoyEnemy, 75
gameEngine
Game, 69
RobotThread, 177
gameGoalSlotL
config, 15
gameGoalSlotM
config, 15
gameGoalSlotOutsideLeft
config, 15
gameGoalSlotOutsideRight
config, 15
gameGoalSlotR
config, 15
gameMinimumDistanceEnemyToDumpSlot
config, 20
gameMinimumDistanceToEnemyRobot
config, 21
gameOver
Referee, 172
gameStart
Referee, 172
GameState
Game, 64
gameWieWeitMussDerGegnerVomZielEntferntSeinImAnnoyModus
config, 21
gameWievielBesserMussEinPuckSeinUmDasZielZuWechseln
config, 21
gameZielwackelfilterTiefpassKoeffizient
config, 21
gameengine.cpp, 207
gameengine.h, 207
generateGrid
PathPlanning, 141
geoFieldHeight
config, 21
geoFieldWidth
config, 21
geoGoalHeight
config, 21
geoGoalMarginBottom
config, 21
geoGoalMarginSide
config, 21
geoGoalWidth
config, 21
geoPol_1_14
config, 22
geoPol_1_2
config, 22
geoPol_2_3
config, 22
geoPol_3_4
config, 22
geoPoleRadiusReal
config, 22
geoPoleRadiusSim
config, 22

geoPuckRadiusBottom
config, 22
geoPuckRadiusTopReal
config, 22
geoPuckRadiusTopSim
config, 22
geoRobotForkCenterDist
config, 23
getAvoidRestOfField
PathPlanning, 141
getColor
Obstacle, 129
getCoords
Obstacle, 130
getDisableEmergency
MapData, 115
getDistance
PathPlanning::GridPoint, 77
PathRealizer, 153
getDistanceTo
Obstacle, 130
Position, 166
getEnabled
PathPlanning, 141
getFirstTarget
MapData, 115
getGlobalPolarPosition
Orientation, 137
getGlobalPosition
Orientation, 138
getGradientPoint
PathPlanning::GridPoint, 77
getGridRotation
PathPlanning, 142
getIgnorePucks
PathPlanning, 142
getInitialized
Obstacle, 130
getInstance
PlayerX, 162
getLaserData
SensorHighLevel, 184
getLastColor
Cam, 53
getLastUpdate
Obstacle, 130
getListByType
MapData, 115
getNeighbors
PathPlanning::GridPoint, 77
getObstacle
MapData, 116
getOrientation
Obstacle, 130
getParkPosition
Game, 67
getPathPlanning
RobotThread, 177
getPixelColor
Cam, 53
getPointerToPathPlanner
MapData, 117
getPosition
Obstacle, 131
getPositionToTurnRoboToMiddlePoint
Game, 67
getRobotPosition
MapData, 117
getSelfpath
PlayerX, 162
getSimulationDetected
MapData, 117
getSonarData
SensorHighLevel, 184
getState
ActorHighLevel, 38
GameEngine, 72
SensorHighLevel, 184
getStatus
Obstacle, 131
getTargetForPuck
Game, 67
getTargetForPuckBeforePole
Game, 67
getTargetNearEnemy
MapData, 117
getTeamColor
MapData, 118
SensorHighLevel, 184
getType
Obstacle, 131
getVelocityProfile
PathRealizer, 153
goalSlotCounterList
Game, 69
grabFrameAndColor
Cam, 53
graphLaserMedian
MainWindow, 109
graphLaserObjects
MainWindow, 109
graphLaserRaw
MainWindow, 109
graphLaserReduced
MainWindow, 109
graphPIDAIst
MainWindow, 109
graphPIDASoll
MainWindow, 109
graphPIDDIst
MainWindow, 109
graphPIDDSoll
MainWindow, 109
graphRobotScatter
MainWindow, 109
graphSplineCurve

MainWindow, 110
graphWPScatter
MainWindow, 110
graphicsScene_1
MainWindow, 108
graphicsScene_2
MainWindow, 108
graphicsScene_3
MainWindow, 108
graphicsScene_4
MainWindow, 108
graphicsScene_5
MainWindow, 108
graphicsScene_6
MainWindow, 109
grid
PathPlanning, 145
grid2AB
PathPlanning, 142
grid2XY
PathPlanning, 142, 143
gridA
PathPlanning::GridPoint, 78
gridB
PathPlanning::GridPoint, 78
gridIndex
PathPlanning, 143
GridPoint
PathPlanning::GridPoint, 76
gridRotation
PathPlanning, 145
gridSizeA
PathPlanning, 145
gridSizeB
PathPlanning, 145
gridSpacing
PathPlanning, 145
guiBrushGoalBlue
config, 15
guiBrushGoalUndef
config, 15
guiBrushGoalYellow
config, 15
guiBrushMe
config, 15
guiBrushPole
config, 15
guiBrushPuckBlocked
config, 15
guiBrushPuckMoving
config, 15
guiPenDummy
config, 15
guiPenFieldPrimary
config, 15
guiPenFieldSecondary
config, 16
guiPenMe
config, 16
guiPenOpponent
config, 16
guiPenOrient
config, 16
guiPenPole
config, 16
guiPenPuckMe
config, 16
guiPenPuckMeOuter
config, 16
guiPenPuckOpponent
config, 16
guiPenPuckOpponentOuter
config, 16
guiPenPuckUndef
config, 16
guiPenPuckUndefOuter
config, 16
guiPenTarget
config, 16
guiPuckKlickTolerance
config, 23

HERMES_A_B
hermescodes.h, 208
HERMES_ANGELINAFOUND
hermescodes.h, 208
HERMES_CONNECT
hermescodes.h, 208
HERMES_DATA_T1
hermescodes.h, 208
HERMES_DATA_T2
hermescodes.h, 208
HERMES_DATA_T3
hermescodes.h, 208
HERMES_DATA_T4
hermescodes.h, 208
HERMES_DONE
hermescodes.h, 209
HERMES_ERROR
hermescodes.h, 209
HERMES_GAME_OVER
hermescodes.h, 209
HERMES_GAME_START
hermescodes.h, 209
HERMES_KEEP_ALIVE
hermescodes.h, 209
HERMES_LOOKINGFOR
hermescodes.h, 209
HERMES_READY
hermescodes.h, 209
HERMES_SCORE
hermescodes.h, 209
HERMES_STATUS
hermescodes.h, 209
HERMES_TEAMCOLOR
hermescodes.h, 209
hadEmergency

SensorHighLevel, 187
handleEnemyPuck
Game, 67
HandleEnemyPuckState
Game, 65
handleEnemyPuckState
Game, 69
height
CameraParams, 56
Hermes, 79
∼Hermes, 80
Hermes, 80
messageSize, 80
myTeamID, 80
referee, 80
hermes.cpp, 207
hermes.h, 207
hermescodes.h, 207
HERMES_A_B, 208
HERMES_ANGELINAFOUND, 208
HERMES_CONNECT, 208
HERMES_DATA_T1, 208
HERMES_DATA_T2, 208
HERMES_DATA_T3, 208
HERMES_DATA_T4, 208
HERMES_DONE, 209
HERMES_ERROR, 209
HERMES_GAME_OVER, 209
HERMES_GAME_START, 209
HERMES_KEEP_ALIVE, 209
HERMES_LOOKINGFOR, 209
HERMES_READY, 209
HERMES_SCORE, 209
HERMES_STATUS, 209
HERMES_TEAMCOLOR, 209

INENEMYGOAL
obstacle.h, 217
INIT
GameEngine, 72
SensorLowLevel, 191
INMYGOAL
obstacle.h, 217
ISMOVING
obstacle.h, 217
iDeltaA
ActorHighLevel, 41
PathRealizer, 156
iDeltaL
ActorHighLevel, 41
PathRealizer, 156
ignorePucks
PathPlanning, 145
ignoreSignals
ActorHighLevel, 38
init
PathPlanning::GridPoint, 78
initGridPoint
PathPlanning, 143
integrationTime
PathRealizer, 156
internalWP
ActorHighLevel, 41
PathRealizer, 156
intrinsicCost
PathPlanning::GridPoint, 78
isConnected
Referee, 172
isConsimilarTo
Position, 166, 167
isGameStarted
Game, 69
isInSlowTurn
SensorHighLevel, 188
isInSpecifiedArea
Obstacle, 131
isOutsideArena
PathPlanning::GridPoint, 78
isPositionInStartField
Position, 167
isPuckAwayFromEnemy
Game, 67
isPuckAwayFromPole
Game, 67
isPuckInFork
MapData, 118
isRefereeEmergency
ActorLowLevel, 46
isVerbose
Referee, 172

kernel
FilterParams, 62

l_solve
tkqt::band_matrix, 49
LaserPlotData
AVERAGE, 81
MEDIAN, 81
OBJECTS, 81
RAW, 81
REDUCED, 81
laserArrayLength
SensorLowLevel, 193
laserData
SensorLowLevel, 193
LaserPlotData, 80
angles, 82
data, 82
dataType, 82
DataTypeEnum, 81
LaserPlotData, 81
LaserPlotData, 81
sizes, 82
laserProxy
SensorLowLevel, 193
laserTime
SensorLowLevel, 193

laserplotdata.h, 210
lastDeltaA
ActorHighLevel, 41
PathRealizer, 156
lastDeltaL
ActorHighLevel, 41
PathRealizer, 156
lastTargetedPuck
Game, 69
left
MainWindow, 94
Log, 82
customLogger, 83
Log, 83
logParams, 83
mainWindow, 83
setMainWindowReference, 83
streamLogEnabled, 83
log.cpp, 210
log.h, 210
LogParams
CRITICAL, 84, 85
DEBUG, 84, 85
FATAL, 84, 85
WARNING, 84, 85
logAI
LogParams, 85
logActor
LogParams, 85
logData
LogParams, 85
logLevel
LogParams, 85
logLevelEnum
LogParams, 84, 85
logMain
LogParams, 85
logOthers
LogParams, 85
LogParams, 84
logAI, 85
logActor, 85
logData, 85
logLevel, 85
logLevelEnum, 84, 85
logMain, 85
logOthers, 85
logPlots, 85
logSensor, 86
logParams
Log, 83
mainwindow.h, 214
LogParams.h, 211
logPlots
LogParams, 85
logSensor
LogParams, 86
logparams.h, 210
lowPass
ActorHighLevel, 38
PathRealizer, 154
lu_decompose
tkqt::band_matrix, 49
lu_solve
tkqt::band_matrix, 49

ME
obstacle.h, 217
MEDIAN
LaserPlotData, 81
MY_AREA
obstacle.h, 216
MY_BEHIND_GOAL_AREA
obstacle.h, 216
MY_GOAL_AREA
obstacle.h, 216
m_a
tkqt::spline, 195
m_angles
ConstrainedLaserData, 59
m_b
tkqt::spline, 195
m_c
tkqt::spline, 195
m_certainty
Position, 169
m_changeOrientation
MainWindow, 110
m_d
tkqt::spline, 196
m_filteredDepth
ConstrainedLaserData, 59
m_lower
tkqt::band_matrix, 51
m_rawDepths
ConstrainedLaserData, 59
m_rot
Position, 169
m_size
Position, 169
m_upper
tkqt::band_matrix, 51
m_x
Position, 169
tkqt::spline, 196
m_y
Position, 169
tkqt::spline, 196
main
main.cpp, 211
main.cpp, 211
main, 211
MainWindow, 86
∼MainWindow, 92
back, 92
changeCamParams, 92
changeFilterParams, 92

changeLogParams, 92
changePIDParams, 93
clearTargets, 93
colorMap, 108
colors, 108
convertX, 93
convertY, 93, 94
drawMap, 94
forward, 94
graphLaserMedian, 109
graphLaserObjects, 109
graphLaserRaw, 109
graphLaserReduced, 109
graphPIDAIst, 109
graphPIDASoll, 109
graphPIDDIst, 109
graphPIDDSoll, 109
graphRobotScatter, 109
graphSplineCurve, 110
graphWPScatter, 110
graphicsScene_1, 108
graphicsScene_2, 108
graphicsScene_3, 108
graphicsScene_4, 108
graphicsScene_5, 108
graphicsScene_6, 109
left, 94
m_changeOrientation, 110
MainWindow, 92
MainWindow, 92
map, 110
mousePressEvent, 94, 95
on_btn_ReleasePuck_clicked, 95
on_btn_StartGame_clicked, 95
on_btnDetectColor_clicked, 95
on_camSourceSpin_valueChanged, 95
on_cbStream_stateChanged, 95, 96
on_logCBAI_stateChanged, 96
on_logCBActor_stateChanged, 96
on_logCBData_stateChanged, 96, 97
on_logCBOther_stateChanged, 97
on_logCBPlot_stateChanged, 97
on_logCBSensor_stateChanged, 97, 98
on_logLevel_currentIndexChanged, 98
on_pushButton_StartOrientation_clicked, 98
on_refreshTime_valueChanged, 98
on_sizeX_textChanged, 98, 99
on_sizeY_textChanged, 99
on_spinBox_kernel_valueChanged, 99
on_spinPIDAD_valueChanged, 99
on_spinPIDAI_valueChanged, 100
on_spinPIDAP_valueChanged, 100
on_spinPIDVD_valueChanged, 100
on_spinPIDVI_valueChanged, 101
on_spinPIDVP_valueChanged, 101
on_streamLog_stateChanged, 101
on_streamPID_stateChanged, 102
on_streamPath_stateChanged, 102
on_streamSensor_stateChanged, 102
on_updateSpinner_valueChanged, 102
orientationSetup, 103
refresh, 103
refreshtimer, 110
right, 103
robotRemoteControllUpdate, 103
scatterLaserObjects, 110
scatterRobotStyle, 110
scatterWPStyle, 110
setrefreshrate, 103
setup, 104
signalChangeCamParams, 104
signalChangeFilterParams, 104
signalChangePIDParams, 104
signalStartOrientation, 105
signalTestColorDetect, 105
signalTestPuckRelease, 105
signalTestStartGame, 105
slotDisplayFrame, 105
slotLaserDisplay, 105, 106
slotLog, 106
slotPIDPlot, 106
slotRestartTimerDisplay, 106
slotSimulationDetect, 107
slotUpdateColorLabel, 107
stop, 107
strongleft, 107
strongright, 107
timer, 110
TrackingMe, 111
TrackingYou, 111
ui, 111
updatePathDisplay, 107, 108
updateRemoteOdometry, 108
mainWindow
Log, 83
RobotThread, 177
robotThread.cpp, 226
mainwindow.cpp, 212
mainwindow.h, 213
logParams, 214
map
MainWindow, 110
mapAbstandRoboterZentrumZuGabel
config, 24
MapData, 111
∼MapData, 114
checkForEnemyNearTraget, 114
cleanup, 114
clearTargets, 114
compareObstacleTimestamps, 114
deleteFirstTarget, 115
deleteObstacle, 115
disableEmergency, 121
getDisableEmergency, 115
getFirstTarget, 115
getListByType, 115

getObstacle, 116
getPointerToPathPlanner, 117
getRobotPosition, 117
getSimulationDetected, 117
getTargetNearEnemy, 117
getTeamColor, 118
isPuckInFork, 118
MapData, 114
MapData, 114
mutexDisableEmergency, 121
mutexPointerToPathPlanner, 121
mutexPoles, 121
mutexPuckInFork, 121
mutexPucks, 121
mutexRobotDummy, 122
mutexRobotME, 122
mutexRobotOpponent, 122
mutexSimulationDetected, 122
mutexTargetNearEnemy, 122
mutexTargets, 122
mutexTeamColor, 122
mutexUnidentified, 122
obstacleDummy, 122
obstacleMe, 123
obstacleOpponent, 123
obstaclesPoles, 123
obstaclesPucks, 123
obstaclesTargets, 123
obstaclesUnidentified, 123
operator=, 118
organizeObstacles, 118
pointerToPathPlanner, 123
puckInFork, 123
setActualColor, 119
setDisableEmergency, 119
setObstacle, 119, 120
setPointerToPathPlanner, 120
setProbableColor, 120
setPuckInFork, 120
setSimulationDetected, 121
setTargetNearEnemy, 121
simulationDetected, 123
targetNearEnemy, 124
teamColor, 124
mapIgnorePuckInsideEnemyDistance
config, 24
mapPolePuckFusionDistance
config, 24
mapPuckPuckFusionDistance
config, 24
mapToleranzBisWohinEinPuckInDerGabelIstMAX
config, 24
mapToleranzBisWohinEinPuckInDerGabelIstMIN
config, 24
mapdata.cpp, 214
mapdata.h, 214
maxA
PathPlanning, 145
maxAngle
SensorHighLevel, 188
maxB
PathPlanning, 145
maxWaviness
PathRealizer, 156
MedianFilter, 124
filter, 124
medianfilter
Filter, 31
medianfilter.cpp, 214
medianfilter.h, 214
medianfilter_new.cpp, 215
medianfilter_new.h, 215
mergeWith
Obstacle, 131
messageSize
Hermes, 80
Referee, 174
messengerOfTheGods
Referee, 174
minA
PathPlanning, 145
minAngle
SensorHighLevel, 188
minB
PathPlanning, 145
minimumSizeHint
CVImageWidget, 60
mousePressEvent
MainWindow, 94, 95
moveForward
ActorLowLevel, 47
moveRobot
ActorLowLevel, 45
mutexDisableEmergency
MapData, 121
mutexFilterParameter
SensorHighLevel, 188
mutexPidHist
ActorHighLevel, 41
mutexPointerToPathPlanner
MapData, 121
mutexPoles
MapData, 121
mutexPuckInFork
MapData, 121
mutexPucks
MapData, 121
mutexRobotDummy
MapData, 122
mutexRobotME
MapData, 122
mutexRobotOpponent
MapData, 122
mutexSimulationDetected
MapData, 122
mutexState

ActorHighLevel, 42
SensorHighLevel, 188
mutexTargetNearEnemy
MapData, 122
mutexTargets
MapData, 122
mutexTeamColor
MapData, 122
SensorHighLevel, 188
mutexUnidentified
MapData, 122
mutexVideoCapture
Cam, 54
myTeamID
Hermes, 80
Referee, 174

NEUTRAL_AREA
obstacle.h, 216
NOMODIFY
obstacle.h, 217
NONE
cam.h, 199
num_lower
tkqt::band_matrix, 50
num_upper
tkqt::band_matrix, 50
numWP
ActorHighLevel, 42
PathRealizer, 156

OBJECTS
LaserPlotData, 81
OPPONENT
obstacle.h, 217
ORIENTATION
sensor.h, 227
sensorhighlevel.h, 229
ORIENTATION_VALIDATION
sensor.h, 227
sensorhighlevel.h, 229
OUT_OF_AREA
obstacle.h, 216
ObsFilterAnzahl
FilterParams, 62
ObsFilterSchnitt
FilterParams, 62
Obstacle, 125
~Obstacle, 129
bInitialized, 134
cLastUpdate, 134
cPosition, 134
enumColor, 134
enumStatus, 134
enumType, 134
getColor, 129
getCoords, 130
getDistanceTo, 130
getInitialized, 130
getLastUpdate, 130
getOrientation, 130
getPosition, 131
getStatus, 131
getType, 131
isInSpecifiedArea, 131
mergeWith, 131
Obstacle, 127–129
operator<, 132
operator==, 132
setColor, 132
setCoords, 132
setInitialized, 133
setLastUpdate, 133
setOrientation, 133
setPosition, 133
setStatus, 133
setType, 133
obstacle.h
BLOCKED, 217
DUMMY, 217
ENEMY_AREA, 216
ENEMY_BEHINDGOAL_AREA, 216
ENEMY_GOAL_AREA, 216
ENEMY_GOAL_INFLUENCE, 216
GOAL_ZONE_LEFT, 216
GOAL_ZONE_MID, 216
GOAL_ZONE_RIGHT, 216
INENEMYGOAL, 217
INMYGOAL, 217
ISMOVING, 217
ME, 217
MY_AREA, 216
MY_BEHIND_GOAL_AREA, 216
MY_GOAL_AREA, 216
NEUTRAL_AREA, 216
NOMODIFY, 217
OPPONENT, 217
OUT_OF_AREA, 216
POLE, 217
POLE_AREA, 216
PUCK, 217
TARGET, 217
UNBLOCKED, 217
UNDEFINED, 217
UNIDENTIFIED, 217
obstacle.cpp, 215
obstacle.h, 215
FieldArea, 216
ObstacleColor, 216
ObstacleStatus, 217
ObstacleType, 217
ObstacleColor
obstacle.h, 216
obstacleCoordinateTolerance
config, 24
obstacleDummy
MapData, 122

obstacleMe
MapData, 123
obstacleNumberOfPoles
config, 24
obstacleNumberOfPucks
config, 25
obstacleOpponent
MapData, 123
ObstacleStatus
obstacle.h, 217
ObstacleType
obstacle.h, 217
obstaclesEnemy
PathPlanning, 146
obstaclesPole
PathPlanning, 146
obstaclesPoles
MapData, 123
obstaclesPuck
PathPlanning, 146
obstaclesPucks
MapData, 123
obstaclesTargets
MapData, 123
obstaclesUnidentified
MapData, 123
on_btn_ReleasePuck_clicked
MainWindow, 95
on_btn_StartGame_clicked
MainWindow, 95
on_btnDetectColor_clicked
MainWindow, 95
on_camSourceSpin_valueChanged
MainWindow, 95
on_cbStream_stateChanged
MainWindow, 95, 96
on_logCBAI_stateChanged
MainWindow, 96
on_logCBActor_stateChanged
MainWindow, 96
on_logCBData_stateChanged
MainWindow, 96, 97
on_logCBOther_stateChanged
MainWindow, 97
on_logCBPlot_stateChanged
MainWindow, 97
on_logCBSensor_stateChanged
MainWindow, 97, 98
on_logLevel_currentIndexChanged
MainWindow, 98
on_pushButton_StartOrientation_clicked
MainWindow, 98
on_refreshTime_valueChanged
MainWindow, 98
on_sizeX_textChanged
MainWindow, 98, 99
on_sizeY_textChanged
MainWindow, 99
on_spinBox_kernel_valueChanged
MainWindow, 99
on_spinPIDAD_valueChanged
MainWindow, 99
on_spinPIDAI_valueChanged
MainWindow, 100
on_spinPIDAP_valueChanged
MainWindow, 100
on_spinPIDVD_valueChanged
MainWindow, 100
on_spinPIDVI_valueChanged
MainWindow, 101
on_spinPIDVP_valueChanged
MainWindow, 101
on_streamLog_stateChanged
MainWindow, 101
on_streamPID_stateChanged
MainWindow, 102
on_streamPath_stateChanged
MainWindow, 102
on_streamSensor_stateChanged
MainWindow, 102
on_updateSpinner_valueChanged
MainWindow, 102
operator<
Obstacle, 132
operator∗
qcustomplot.h, 223
operator()
tkqt::band_matrix, 50
tkqt::spline, 195
operator+
qcustomplot.h, 224
operator-
qcustomplot.h, 224
operator/
qcustomplot.h, 224
operator=
MapData, 118
Orientation, 138
operator==
Obstacle, 132
Position, 167
organizeObstacles
MapData, 118
Orientation, 135
∼Orientation, 136
angleBetweenAB, 136
beginOrientation, 136
checkObjectsOnLine, 136
distancePolar, 137
getGlobalPolarPosition, 137
getGlobalPosition, 138
operator=, 138
Orientation, 136
orientationSetup
MainWindow, 103
orientierung.cpp, 218

orientierung.h, 218

POLE
obstacle.h, 217
position.h, 222
POLE_AREA
obstacle.h, 216
PRE_GAME_STATE
sensor.h, 227
sensorhighlevel.h, 229
PUCK
obstacle.h, 217
position.h, 222
PUSH_AND_RELEASE_PUCK
actorhighlevel.h, 198
PATH_SHOWRES
config, 25
PID_A_D
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PID_A_I
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PID_A_P
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PID_V_D
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PID_V_I
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PID_V_P
ActorHighLevel, 42
PathRealizer, 157
PIDParams, 160
PIDParams, 159
PID_A_D, 160
PID_A_I, 160
PID_A_P, 160
PID_V_D, 160
PID_V_I, 160
PID_V_P, 160
PIDPlotData, 160
distanzIst, 161
distanzSoll, 161
time, 161
winkelIst, 161
winkelSoll, 161
POLESIZE_CM
config, 28
PUCKSIZE_INNER_CM
config, 28
PUCKSIZE_OUTER_CM
config, 28
paintEvent
CVImageWidget, 60
PathPlanning::GridPoint
GRID, 76
ROBOT, 76
TARGET, 76
PathPlotData
SPLINE, 149
WAYPOINTS, 149
PathRealizer
RUNNING, 153
STOP, 153
pathAdjacencyMultiplier
config, 25
pathArenaFieldAvoidMaxY
config, 25
pathArenaMaxX
config, 25
pathArenaMaxY
config, 25
pathArenaMinX
config, 25
pathArenaMinY
config, 26
pathDisplay
PathPlanning, 143
pathEnemyCloseCost
config, 26
pathEnemyCloseDist
config, 26
pathEnemyFarDist
config, 26
pathEnemyStartCost
config, 26
pathGridSpacingBase
config, 26
pathMaxWPIterations
config, 26
pathPlanner
RobotThread, 178
PathPlanning, 138
~PathPlanning, 141
avoidRestOfField, 144
calculatePathCosts, 141
calculateWaypoints, 141
enabled, 144
generateGrid, 141
getAvoidRestOfField, 141
getEnabled, 141
getGridRotation, 142
getIgnorePucks, 142
grid, 145
grid2AB, 142
grid2XY, 142, 143
gridIndex, 143
gridRotation, 145
gridSizeA, 145

gridSizeB, 145
gridSpacing, 145
ignorePucks, 145
initGridPoint, 143
maxA, 145
maxB, 145
minA, 145
minB, 145
obstaclesEnemy, 146
obstaclesPole, 146
obstaclesPuck, 146
pathDisplay, 143
PathPlanning, 141
PathPlanning, 141
planPath, 144
robot, 146
robotA, 146
robotB, 146
robotRot, 146
robotX, 146
robotY, 146
sendUpdatedWaypoints, 144
setAvoidRestOfField, 144
setEnabled, 144
setIgnorePucks, 144
streamPathEnabled, 147
targetA, 147
targetB, 147
targetRot, 147
targetX, 147
targetY, 147
timer, 147
pathPlanning
PathPlanning::GridPoint, 78
PathPlanning::GridPoint, 75
active, 78
calculateIntrinsicCost, 77
constrainAngle, 77
getDistance, 77
getGradientPoint, 77
getNeighbors, 77
gridA, 78
gridB, 78
GridPoint, 76
init, 78
intrinsicCost, 78
isOutsideArena, 78
pathPlanning, 78
PointType, 76
positionX, 79
positionY, 79
type, 79
value, 79
pathPlanningEnabledUpwardsOfThisDistance
config, 26
PathPlotData, 148
data, 149
dataSizeX, 149
dataSizeY, 149
dataType, 149
DataTypeEnum, 148, 149
PathPlotData, 149
PathPlotData, 149
robot, 150
splineLength, 150
splineX, 150
splineY, 150
target, 150
waypoints, 150
PathPlotData::Point, 163
value, 163
x, 163
y, 163
pathPoleCloseCost
config, 26
pathPoleCloseDist
config, 27
pathPoleFarDist
config, 27
pathPoleStartCost
config, 27
pathPuckCloseCost
config, 27
pathPuckCloseDist
config, 27
pathPuckFarDist
config, 27
pathPuckStartCost
config, 27
PathRealizer, 150
∼PathRealizer, 153
constrainAngle, 153
elapsedTime, 155
getDistance, 153
getVelocityProfile, 153
iDeltaA, 156
iDeltaL, 156
integrationTime, 156
internalWP, 156
lastDeltaA, 156
lastDeltaL, 156
lowPass, 154
maxWaviness, 156
numWP, 156
PID_A_D, 157
PID_A_I, 157
PID_A_P, 157
PID_V_D, 157
PID_V_I, 157
PID_V_P, 157
PathRealizer, 153
PathRealizer, 153
periodMotionControl, 156
pidHistDistIst, 157
pidHistDistSoll, 157
pidHistMutex, 157

pidHistTime, 158
pidHistWinkelIst, 158
pidHistWinkelSoll, 158
robotObstacle, 158
signalPIDPlot, 154
signalSendRobotControlParams, 154
signalSplinePlot, 154
slotChangePIDParams, 154
slotMotionControl, 155
slotTimerSendPIDPlot, 155
slotUpdateWaypoints, 155
splineProgress, 158
splineToQVector, 155
splineX, 158
splineY, 158
state, 158
StatePathProcessing, 153
streamPIDEnabled, 158
takeDimension, 155
timeOfStart, 159
timerMotionControl, 159
timerPIDPlot, 159
velProfile, 159
pathRealizer
RobotThread, 178
pathRealizer.h, 220
pathRobotRadius
config, 27
pathTargetApproachAngleFullCost
config, 27
pathTargetApproachAngleInfluenceDistance
config, 27
pathTargetApproachAngleMaxDeviationWithoutFullCost
config, 28
pathplanning.cpp, 218
round, 218
pathplanning.h, 219
pathplotdata.h, 219
periodAlive
config, 28
periodEgoPos
config, 28
periodMotionControl
PathRealizer, 156
periodTillAnnoy
config, 28
pidHistDistIst
ActorHighLevel, 42
PathRealizer, 157
pidHistDistSoll
ActorHighLevel, 43
PathRealizer, 157
pidHistMutex
PathRealizer, 157
pidHistTime
ActorHighLevel, 43
PathRealizer, 158
pidHistWinkelIst
ActorHighLevel, 43
PathRealizer, 158
pidHistWinkelSoll
ActorHighLevel, 43
PathRealizer, 158
pidparams.h, 220
pidplotdata.h, 220, 221
planPath
PathPlanning, 144
player.cpp, 221
player.h, 221
PlayerCc, 31
PlayerX, 161
didWeStartPlayerOurselves, 163
getInstance, 162
getSelfpath, 162
startPlayer, 162
stopPlayerIfStarted, 162
PointType
PathPlanning::GridPoint, 76
pointerToPathPlanner
MapData, 123
PosFilterAnzahl
FilterParams, 62
PosFilterSchnitt
FilterParams, 62
Position, 164
certainty, 166
getDistanceTo, 166
isConsimilarTo, 166, 167
isPositionInStartField, 167
m_certainty, 169
m_rot, 169
m_size, 169
m_x, 169
m_y, 169
operator==, 167
Position, 165, 166
rot, 167, 168
setCertainty, 168
sizeType, 168
x, 168
y, 169
position.h
POLE, 222
PUCK, 222
ROBOT, 222
UNKNOWN, 222
position.cpp, 221
position.h, 222
SizeType, 222
positionProxy
ActorLowLevel, 47
SensorLowLevel, 193
positionWasReached
ActorHighLevel, 43
positionX
PathPlanning::GridPoint, 79

positionY
PathPlanning::GridPoint, 79
prepositionInitialized
SensorHighLevel, 188
previousObjects
SensorHighLevel, 189
previousOdometryPosition
SensorLowLevel, 193
previousOdometryTime
SensorLowLevel, 193
previousPosition
SensorHighLevel, 189
previousTurnRate
ActorLowLevel, 47
previousVelocity
ActorLowLevel, 47
puckGrabbed
SensorHighLevel, 184
puckInFork
MapData, 123
puckIsCloseToPoleDistance
config, 28

Q_DECLARE_TYPEINFO
qcustomplot.h, 224
qcustomplot.cpp, 222
qcustomplot.h, 223
operator∗, 223
operator+, 224
operator-, 224
operator/, 224
Q_DECLARE_TYPEINFO, 224
quit
Game, 68
SensorLowLevel, 192
quitting
ActorHighLevel, 43
Game, 70
SensorLowLevel, 194

RAW
LaserPlotData, 81
RECOGNITION
sensor.h, 227
sensorhighlevel.h, 229
REDUCED
LaserPlotData, 81
RELEASE
Game, 65
RELEASE_PUCK
actorhighlevel.h, 198
RELEASE_PUCK_FROM_PUSH
actorhighlevel.h, 198
ROBOT
PathPlanning::GridPoint, 76
position.h, 222
RUN
SensorLowLevel, 191
RUNNING
actorhighlevel.h, 198
PathRealizer, 153
r_solve
tkqt::band_matrix, 50
ROBOSIZE_CM
config, 29
rawDepths
ConstrainedLaserData, 58
readSensorData
SensorLowLevel, 192
ready
Referee, 175
recognition
SensorHighLevel, 184, 185
refIP
config, 28
refPort
config, 29
refVerbose
config, 29
Referee, 170
∼Referee, 171
abValues, 172
connectFailed, 172
connectToServer, 172
connected, 172
connection, 174
detectionStart, 172
disconnected, 172
gameOver, 172
gameStart, 172
isConnected, 172
isVerbose, 172
messageSize, 174
messengerOfTheGods, 174
myTeamID, 174
ready, 175
Referee, 171
reportDone, 173
reportGoal, 173
reportReady, 173
sendAlive, 173
setVerbose, 173
slotConnected, 173
slotDisconnected, 173
slotRead, 173
stopMovement, 173
tellAbRatio, 174
tellEgoPos, 174
tellTeamColor, 174
testMode, 175
trueColorOfTeam, 174
verbose, 175
wLimit, 175
referee
GameEngine, 75
Hermes, 80
referee.h

blue, 225
yellow, 225
referee.cpp, 224
referee.h, 224
TeamColor, 225
refresh
MainWindow, 103
refreshtimer
MainWindow, 110
releasePuckOrigin
ActorHighLevel, 43
reportDone
Referee, 173
reportGoal
Referee, 173
reportReady
Referee, 173
resetPIDtempVars
ActorHighLevel, 38
resize
tkqt::band_matrix, 50
right
MainWindow, 103
robot
PathPlanning, 146
PathPlotData, 150
SensorLowLevel, 194
robotA
PathPlanning, 146
robotB
PathPlanning, 146
robotObstacle
PathRealizer, 158
robotPosition
ActorHighLevel, 43
robotRemoteControllUpdate
MainWindow, 103
robotRot
PathPlanning, 146
RobotThread, 175
∼RobotThread, 176
actorHighLevel, 177
actorLowLevel, 177
cam, 177
game, 177
gameEngine, 177
getPathPlanning, 177
mainWindow, 177
pathPlanner, 178
pathRealizer, 178
RobotThread, 176
RobotThread, 176
sensorHighLevel, 178
sensorLowLevel, 178
threadActorHighLevel, 178
threadCam, 178
threadGame, 178
threadGameEngine, 178
threadPathPlanner, 178
threadPathRealizer, 179
threadRobotLowLevel, 179
threadSensorHighLevel, 179
robotThread.cpp, 225, 226
mainWindow, 226
robotThread.h, 226
robotX
PathPlanning, 146
robotY
PathPlanning, 146
rot
Position, 167, 168
round
pathplanning.cpp, 218
run
Game, 68
SensorLowLevel, 192

SELECT_GOAL_SLOT
Game, 64
SELECT_PUCK
Game, 64
SPLINE
PathPlotData, 149
STOP
actorhighlevel.h, 198
GameEngine, 72
PathRealizer, 153
SENSOR_COLLISION_AT
config, 29
SENSOR_DELTA_ANGLE
config, 29
SENSOR_RADIUS_ROBOT
config, 30
SENSOR_WAIT_COUNTER
config, 30
saved_diag
tkqt::band_matrix, 51
scatterLaserObjects
MainWindow, 110
scatterRobotStyle
MainWindow, 110
scatterWPStyle
MainWindow, 110
sendAlive
Referee, 173
sendOdometryData
SensorHighLevel, 185
sendUpdatedWaypoints
PathPlanning, 144
sensor.h
COLOR_DETECTION_START, 227
COLOR_DETECTION_WAIT, 227
ORIENTATION, 227
ORIENTATION_VALIDATION, 227
PRE_GAME_STATE, 227
RECOGNITION, 227
WAIT, 227

sensor.h, 227
SensorStates, 227
SensorLowLevel
INIT, 191
RUN, 191
SensorHighLevel, 179
∼SensorHighLevel, 182
avoideCollision, 182
cPolePositions, 187
calculateObjCenter, 182
collectorObj, 186
constrainData, 183
constrainedData, 186
convertPolToGlobalCoordinates, 183
counter, 187
currentObjects, 187
currentPosition, 187
currentState, 187
distanceKartesisch, 183
distancePolar, 183
driveToPreposition, 183
dummyAngleOffset, 187
dummyPosition, 187
extractObjects, 183
filterParameter, 187
getLaserData, 184
getSonarData, 184
getState, 184
getTeamColor, 184
hadEmergency, 187
isInSlowTurn, 188
maxAngle, 188
minAngle, 188
mutexFilterParameter, 188
mutexState, 188
mutexTeamColor, 188
prepositionInitialized, 188
previousObjects, 189
previousPosition, 189
puckGrabbed, 184
recognition, 184, 185
sendOdometryData, 185
SensorHighLevel, 182
SensorHighLevel, 182
setState, 185
setTeamColor, 185
signalEmergencyStopEnabled, 185
signalPlanNewPath, 185
signalSendLaserData, 185, 186
signalSendOdometryData, 186
signalSendRobotControlParams, 186
signalSendTeamColor, 186
signalStartColorDetection, 186
slotColorDetected, 186
slotGetLaserData, 186
slotSetFilterParams, 186
slotStartDetection, 186
streamSensorEnabled, 189
targetsSet, 189
teamColor, 189
timeSinceStart, 189
timerToAbondonColorDetection, 189
timerWaitForValidColorFrame, 189
transmissionPosition, 189
sensorHighLevel
RobotThread, 178
SensorLowLevel, 190
∼SensorLowLevel, 191
angleWeightedAverage, 191
currentOdometryPosition, 192
currentOdometryTime, 193
elapsedTimer, 193
laserArrayLength, 193
laserData, 193
laserProxy, 193
laserTime, 193
positionProxy, 193
previousOdometryPosition, 193
previousOdometryTime, 193
quit, 192
quitting, 194
readSensorData, 192
robot, 194
run, 192
SensorLowLevel, 191
SensorState, 191
SensorLowLevel, 191
signalLaserDataReady, 192
signalLaserPlotRaw, 192
signalSimulationDetect, 192
state, 194
sensorLowLevel
RobotThread, 178
sensorLowLevel.cpp, 229
sensorLowLevel.h, 229
SensorState
SensorLowLevel, 191
SensorStates
sensor.h, 227
sensorhighlevel.h, 228
sensorhighlevel.h
COLOR_DETECTION_START, 229
COLOR_DETECTION_WAIT, 229
ORIENTATION, 229
ORIENTATION_VALIDATION, 229
PRE_GAME_STATE, 229
RECOGNITION, 229
WAIT, 228, 229
sensorhighlevel.cpp, 228
sensorhighlevel.h, 228
SensorStates, 228
set_points
tkqt::spline, 195
setActualColor
MapData, 119
setAvoidRestOfField

PathPlanning, 144
setCertainty
Position, 168
setColor
Obstacle, 132
setCoords
Obstacle, 132
setDisableEmergency
MapData, 119
setEnabled
PathPlanning, 144
setIgnorePucks
PathPlanning, 144
setInitialized
Obstacle, 133
setLastUpdate
Obstacle, 133
setMainWindowReference
Log, 83
setObstacle
MapData, 119, 120
setOdometry
ActorLowLevel, 46
setOrientation
Obstacle, 133
setPointerToPathPlanner
MapData, 120
setPosition
Obstacle, 133
setProbableColor
MapData, 120
setPuckInFork
MapData, 120
setRobotControllParams
ActorLowLevel, 46
setRobotRemoteControllParams
ActorLowLevel, 46
setSimulationDetected
MapData, 121
setState
ActorHighLevel, 38
SensorHighLevel, 185
setStatus
Obstacle, 133
setTargetNearEnemy
MapData, 121
setTeamColor
SensorHighLevel, 185
setType
Obstacle, 133
setVerbose
Referee, 173
setrefreshrate
MainWindow, 103
setup
MainWindow, 104
showImage
CVImageWidget, 60
signalAnnoyFoe
GameEngine, 72
signalChangeCamParams
MainWindow, 104
signalChangeFilterParams
MainWindow, 104
signalChangePIDParams
MainWindow, 104
signalColorDetected
Cam, 53
signalDisplayFrame
Cam, 53
signalEmergencyStopEnabled
GameEngine, 72
SensorHighLevel, 185
signalGatherPuck
Game, 68
signalLaserDataReady
SensorLowLevel, 192
signalLaserPlotRaw
SensorLowLevel, 192
signalPIDPlot
ActorHighLevel, 39
PathRealizer, 154
signalPlanNewPath
SensorHighLevel, 185
signalPuckDone
ActorHighLevel, 39
signalPuckRelease
Game, 68
signalPushAndRelease
Game, 68
signalReportGoal
Game, 68
signalSendLaserData
SensorHighLevel, 185, 186
signalSendOdometryData
SensorHighLevel, 186
signalSendRobotControlParams
ActorHighLevel, 39
PathRealizer, 154
SensorHighLevel, 186
signalSendTeamColor
SensorHighLevel, 186
signalSimulationDetect
SensorLowLevel, 192
signalSplinePlot
ActorHighLevel, 39
PathRealizer, 154
signalStartColorDetectAI
Game, 68
signalStartColorDetection
SensorHighLevel, 186
signalStartDetection
GameEngine, 73
signalStartGame
GameEngine, 73
signalStartOrientation

MainWindow, 105
signalTestColorDetect
MainWindow, 105
signalTestPuckRelease
MainWindow, 105
signalTestStartGame
MainWindow, 105
simulationDetected
MapData, 123
sizeHint
CVImageWidget, 61
SizeType
position.h, 222
sizeType
Position, 168
sizes
LaserPlotData, 82
slotActorHighLevelIsDoneWithPuck
Game, 68
slotAnnoyFoe
Game, 68
slotChangePIDParams
ActorHighLevel, 39
PathRealizer, 154
slotColorDetect
Game, 68
slotColorDetected
SensorHighLevel, 186
slotConnected
Referee, 173
slotDetectionFinished
GameEngine, 73
slotDisconnected
Referee, 173
slotDisplayFrame
MainWindow, 105
slotEmergencyStopEnabled
ActorLowLevel, 46
slotGatherPuck
ActorHighLevel, 39
slotGetLaserData
SensorHighLevel, 186
slotLaserDisplay
MainWindow, 105, 106
slotLog
MainWindow, 106
slotMotionControl
PathRealizer, 155
slotPIDPlot
MainWindow, 106
slotPushAndReleasePuck
ActorHighLevel, 39
slotRead
Referee, 173
slotRefConnectFailed
GameEngine, 73
slotRefConnected
GameEngine, 73
slotRefDetectionStart
GameEngine, 73
slotRefDisconnected
GameEngine, 73
slotRefGameOver
GameEngine, 74
slotRefGameStart
GameEngine, 74
slotRefStopMovement
GameEngine, 74
slotRefTrueColorOfTeam
GameEngine, 74
slotReleasePuck
ActorHighLevel, 40
slotReportGoal
GameEngine, 74
slotRestartTimerDisplay
MainWindow, 106
slotSetCameraParams
Cam, 53
slotSetFilterParams
SensorHighLevel, 186
slotSimulationDetect
MainWindow, 107
slotStartColorDetection
Cam, 53
slotStartDetection
SensorHighLevel, 186
slotStartGame
Game, 69
slotTimerAlive
GameEngine, 74
slotTimerAnnoy
GameEngine, 74
slotTimerEgoPos
GameEngine, 74
slotTimerSendPIDPlot
ActorHighLevel, 40
PathRealizer, 155
slotUpdateColorLabel
MainWindow, 107
slotUpdateWaypoints
ActorHighLevel, 40
PathRealizer, 155
source
CameraParams, 56
spline.cpp, 230
spline.h, 230
splineLength
PathPlotData, 150
splineProgress
PathRealizer, 158
splineToQVector
PathRealizer, 155
splineX
ActorHighLevel, 43
PathPlotData, 150
PathRealizer, 158

splineY
ActorHighLevel, 43
PathPlotData, 150
PathRealizer, 158
startGameEngine
GameEngine, 74
startPIDController
ActorHighLevel, 40
startPlayer
PlayerX, 162
state
ActorHighLevel, 44
Game, 70
GameEngine, 75
PathRealizer, 158
SensorLowLevel, 194
StateNameEnum
GameEngine, 72
StatePathProcessing
actorhighlevel.h, 198
PathRealizer, 153
stop
MainWindow, 107
stopMovement
Referee, 173
stopPlayerIfStarted
PlayerX, 162
streamCamEnabled
Cam, 54
streamLogEnabled
Log, 83
streamPIDEnabled
ActorHighLevel, 44
PathRealizer, 158
streamPathEnabled
PathPlanning, 147
streamSensorEnabled
SensorHighLevel, 189
strongleft
MainWindow, 107
strongright
MainWindow, 107
subAnnoyState
Game, 70
SubGameStateAnnoy
Game, 65

TARGET
obstacle.h, 217
PathPlanning::GridPoint, 76
TARGET_POLE_VARIANCE
config, 30
TARGETSIZE_CM
config, 30
takeDimension
ActorHighLevel, 40
PathRealizer, 155
target
PathPlotData, 150
targetA
PathPlanning, 147
targetB
PathPlanning, 147
targetDistDiffLast
ActorHighLevel, 44
targetDistLast
ActorHighLevel, 44
targetNearEnemy
MapData, 124
targetRot
PathPlanning, 147
targetX
PathPlanning, 147
targetY
PathPlanning, 147
targetsSet
SensorHighLevel, 189
TeamColor
referee.h, 225
teamColor
MapData, 124
SensorHighLevel, 189
teamID
config, 30
tellAbRatio
Referee, 174
tellEgoPos
Referee, 174
tellTeamColor
Referee, 174
tempTurnRate
ActorLowLevel, 47
tempVelocity
ActorLowLevel, 47
testMode
Referee, 175
threadActorHighLevel
RobotThread, 178
threadCam
RobotThread, 178
threadGame
RobotThread, 178
threadGameEngine
RobotThread, 178
threadPathPlanner
RobotThread, 178
threadPathRealizer
RobotThread, 179
threadRobotLowLevel
RobotThread, 179
threadSensorHighLevel
RobotThread, 179
time
PIDPlotData, 161
timeOfStart
ActorHighLevel, 44
PathRealizer, 159

timeSinceStart
SensorHighLevel, 189
timer
Cam, 54
GUI/mainwindow.cpp, 212
MainWindow, 110
PathPlanning, 147
timerAlive
GameEngine, 75
timerEgoPos
GameEngine, 75
timerLostPuck
Game, 70
timerMotionControl
PathRealizer, 159
timerPIDPlot
ActorHighLevel, 44
PathRealizer, 159
timerSendFrame
Cam, 54
timerTillAnnoyEnemy
GameEngine, 75
timerToAbondonColorDetection
SensorHighLevel, 189
timerUpdateGoalSlot
Game, 70
timerWaitForValidColorFrame
SensorHighLevel, 189
tkqt, 31
tkqt::band_matrix, 47
∼band_matrix, 49
band_matrix, 48
dim, 49
l_solve, 49
lu_decompose, 49
lu_solve, 49
m_lower, 51
m_upper, 51
num_lower, 50
num_upper, 50
operator(), 50
r_solve, 50
resize, 50
saved_diag, 51
tkqt::spline, 194
m_a, 195
m_b, 195
m_c, 195
m_d, 196
m_x, 196
m_y, 196
operator(), 195
set_points, 195
toggleFolder
dynsections.js, 205
toggleInherit
dynsections.js, 205
toggleLevel
dynsections.js, 205
toggleVisibility
dynsections.js, 205
TrackingMe
MainWindow, 111
TrackingYou
MainWindow, 111
transmissionPosition
SensorHighLevel, 189
trilateration, 32
circle_circle_intersection, 32
trilateration.cpp, 230
trilateration.h, 230
trueColorOfTeam
Referee, 174
type
PathPlanning::GridPoint, 79

UNBLOCKED
obstacle.h, 217
UNDEFINED
obstacle.h, 217
UNIDENTIFIED
obstacle.h, 217
UNKNOWN
position.h, 222
Ui, 33
ui
MainWindow, 111
updatePathDisplay
MainWindow, 107, 108
updatePeriod
CameraParams, 56
updateRemoteOdometry
MainWindow, 108
updateStripes
dynsections.js, 205

value
PathPlanning::GridPoint, 79
PathPlotData::Point, 163
velProfile
PathRealizer, 159
verbose
Referee, 175
videoCapture
Cam, 54

WAIT
sensor.h, 227
sensorhighlevel.h, 228, 229
WAIT_FOR_DETECTION
GameEngine, 72
WAIT_FOR_GAME
GameEngine, 72
WAIT_FOR_RELEASE
Game, 65
WARNING
LogParams, 84, 85

WAYPOINTS
PathPlotData, 149
wLimit
Referee, 175
waypoints
PathPlotData, 150
width
CameraParams, 56
winkelIst
PIDPlotData, 161
winkelSoll
PIDPlotData, 161

x
PathPlotData::Point, 163
Position, 168

y
PathPlotData::Point, 163
Position, 169
YELLOW
cam.h, 199
yellow
referee.h, 225