

LabRoboticsProject

Generated by Doxygen 1.8.15

1 Namespace Index	1
1.1 Namespace List	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Namespace Documentation	9
5.1 CHRONO Namespace Reference	9
5.1.1 Enumeration Type Documentation	9
5.1.1.1 TIME_TYPE	9
5.1.2 Function Documentation	9
5.1.2.1 getElapsed() [1/2]	10
5.1.2.2 getElapsed() [2/2]	10
5.1.2.3 getType()	10
5.2 ClipperLib Namespace Reference	10
5.2.1 Typedef Documentation	13
5.2.1.1 cInt	13
5.2.1.2 EdgeList	13
5.2.1.3 IntersectList	13
5.2.1.4 JoinList	13
5.2.1.5 long64	14
5.2.1.6 Path	14
5.2.1.7 Paths	14
5.2.1.8 PolyNodes	14
5.2.1.9 PolyOutList	14
5.2.1.10 ulong64	14
5.2.2 Enumeration Type Documentation	14
5.2.2.1 ClipType	14
5.2.2.2 Direction	15
5.2.2.3 EdgeSide	15
5.2.2.4 EndType	15
5.2.2.5 InitOptions	16
5.2.2.6 JoinType	16
5.2.2.7 NodeType	16
5.2.2.8 PolyFillType	16
5.2.2.9 PolyType	17
5.2.3 Function Documentation	17
5.2.3.1 Abs()	17

5.2.3.2 AddPolyNodeToPaths()	17
5.2.3.3 Area() [1/3]	17
5.2.3.4 Area() [2/3]	17
5.2.3.5 Area() [3/3]	18
5.2.3.6 CleanPolygon() [1/2]	18
5.2.3.7 CleanPolygon() [2/2]	18
5.2.3.8 CleanPolygons() [1/2]	18
5.2.3.9 CleanPolygons() [2/2]	18
5.2.3.10 ClosedPathsFromPolyTree()	18
5.2.3.11 DisposeOutPts()	19
5.2.3.12 DistanceFromLineSqrd()	19
5.2.3.13 DistanceSqrd()	19
5.2.3.14 DupOutPt()	19
5.2.3.15 E2InsertsBeforeE1()	19
5.2.3.16 EdgesAdjacent()	19
5.2.3.17 ExcludeOp()	20
5.2.3.18 FindNextLocMin()	20
5.2.3.19 FirstIsBottomPt()	20
5.2.3.20 GetBottomPt()	20
5.2.3.21 GetDx()	20
5.2.3.22 GetHorzDirection()	20
5.2.3.23 GetLowermostRec()	21
5.2.3.24 GetMaximaPair()	21
5.2.3.25 GetMaximaPairEx()	21
5.2.3.26 GetNextInAEL()	21
5.2.3.27 GetOverlap()	21
5.2.3.28 GetOverlapSegment()	21
5.2.3.29 GetUnitNormal()	22
5.2.3.30 HorzSegmentsOverlap()	22
5.2.3.31 InitEdge()	22
5.2.3.32 InitEdge2()	22
5.2.3.33 Int128Mul()	22
5.2.3.34 IntersectListSort()	22
5.2.3.35 IntersectPoint()	23
5.2.3.36 IsHorizontal()	23
5.2.3.37 IsIntermediate()	23
5.2.3.38 IsMaxima()	23
5.2.3.39 IsMinima()	23
5.2.3.40 JoinHorz()	23
5.2.3.41 Minkowski()	24
5.2.3.42 MinkowskiDiff()	24
5.2.3.43 MinkowskiSum() [1/2]	24

5.2.3.44 MinkowskiSum()	[2/2]	24
5.2.3.45 OpenPathsFromPolyTree()		24
5.2.3.46 operator<<()	[1/5]	25
5.2.3.47 operator<<()	[2/5]	25
5.2.3.48 operator<<()	[3/5]	25
5.2.3.49 operator<<()	[4/5]	25
5.2.3.50 operator<<()	[5/5]	25
5.2.3.51 Orientation()		25
5.2.3.52 OutRec1RightOfOutRec2()		26
5.2.3.53 ParseFirstLeft()		26
5.2.3.54 PointCount()		26
5.2.3.55 PointInPolygon()	[1/2]	26
5.2.3.56 PointInPolygon()	[2/2]	26
5.2.3.57 PointIsVertex()		26
5.2.3.58 PointsAreClose()		27
5.2.3.59 Poly2ContainsPoly1()		27
5.2.3.60 PolyTreeToPaths()		27
5.2.3.61 Pt2IsBetweenPt1AndPt3()		27
5.2.3.62 RangeTest()		27
5.2.3.63 RemoveEdge()		27
5.2.3.64 ReverseHorizontal()		28
5.2.3.65 ReversePath()		28
5.2.3.66 ReversePaths()		28
5.2.3.67 ReversePolyPtLinks()		28
5.2.3.68 Round()		28
5.2.3.69 SetDx()		28
5.2.3.70 SimplifyPolygon()		28
5.2.3.71 SimplifyPolygons()	[1/2]	29
5.2.3.72 SimplifyPolygons()	[2/2]	29
5.2.3.73 SlopesEqual()	[1/3]	29
5.2.3.74 SlopesEqual()	[2/3]	29
5.2.3.75 SlopesEqual()	[3/3]	29
5.2.3.76 SlopesNearCollinear()		30
5.2.3.77 SwapIntersectNodes()		30
5.2.3.78 SwapPoints()		30
5.2.3.79 SwapPolyIndexes()		30
5.2.3.80 SwapSides()		30
5.2.3.81 TopX()		30
5.2.3.82 TranslatePath()		31
5.2.3.83 UpdateOutPtIdxs()		31
5.2.4 Variable Documentation		31
5.2.4.1 def_arc_tolerance		31

5.2.4.2 hiRange	31
5.2.4.3 loRange	31
5.2.4.4 pi	31
5.2.4.5 Skip	31
5.2.4.6 two_pi	32
5.2.4.7 Unassigned	32
5.3 DW Namespace Reference	32
5.3.1 Function Documentation	32
5.3.1.1 changeBuffer()	32
5.3.1.2 init()	32
5.3.2 Variable Documentation	32
5.3.2.1 map_buffer	33
5.3.2.2 window	33
5.4 timeutils Namespace Reference	33
5.4.1 Function Documentation	33
5.4.1.1 getTimeS()	33
5.4.1.2 timespecDiff()	33
6 Class Documentation	35
6.1 Angle Class Reference	35
6.1.1 Detailed Description	37
6.1.2 Member Enumeration Documentation	37
6.1.2.1 ANGLE_TYPE	37
6.1.3 Constructor & Destructor Documentation	37
6.1.3.1 Angle() [1/2]	37
6.1.3.2 Angle() [2/2]	37
6.1.4 Member Function Documentation	38
6.1.4.1 add()	38
6.1.4.2 checkValue()	38
6.1.4.3 copy()	38
6.1.4.4 cos()	39
6.1.4.5 degToRad()	39
6.1.4.6 div()	39
6.1.4.7 equal()	40
6.1.4.8 get()	40
6.1.4.9 getType()	40
6.1.4.10 getTypeName()	40
6.1.4.11 greater()	40
6.1.4.12 less()	41
6.1.4.13 mul()	41
6.1.4.14 normalize()	42
6.1.4.15 operator *()	42

6.1.4.16 operator *=()	42
6.1.4.17 operator double()	43
6.1.4.18 operator float()	43
6.1.4.19 operator int()	43
6.1.4.20 operator long()	43
6.1.4.21 operator !=()	43
6.1.4.22 operator +()	44
6.1.4.23 operator +=()	44
6.1.4.24 operator -()	45
6.1.4.25 operator ==()	45
6.1.4.26 operator /()	45
6.1.4.27 operator /=()	46
6.1.4.28 operator <()	46
6.1.4.29 operator <=()	46
6.1.4.30 operator =() [1/2]	46
6.1.4.31 operator =() [2/2]	47
6.1.4.32 operator ==()	47
6.1.4.33 operator >()	47
6.1.4.34 operator >=()	47
6.1.4.35 radToDeg()	48
6.1.4.36 set()	48
6.1.4.37 setType()	48
6.1.4.38 sin()	49
6.1.4.39 sub()	49
6.1.4.40 tan()	49
6.1.4.41 to_string()	49
6.1.4.42 toDeg()	50
6.1.4.43 toRad()	50
6.1.5 Friends And Related Function Documentation	50
6.1.5.1 operator <<	50
6.2 CalSettings Class Reference	51
6.2.1 Member Enumeration Documentation	53
6.2.1.1 InputType	53
6.2.1.2 Pattern	53
6.2.2 Constructor & Destructor Documentation	53
6.2.2.1 CalSettings()	53
6.2.3 Member Function Documentation	53
6.2.3.1 isListOfImages()	53
6.2.3.2 nextImage()	54
6.2.3.3 read()	54
6.2.3.4 readStringList()	54
6.2.3.5 validate()	55

6.2.3.6 write()	55
6.2.4 Member Data Documentation	56
6.2.4.1 aspectRatio	56
6.2.4.2 atImageList	56
6.2.4.3 boardSize	56
6.2.4.4 calibFixPrincipalPoint	56
6.2.4.5 calibrationPattern	56
6.2.4.6 calibZeroTangentDist	56
6.2.4.7 cameraID	57
6.2.4.8 delay	57
6.2.4.9 fixK1	57
6.2.4.10 fixK2	57
6.2.4.11 fixK3	57
6.2.4.12 fixK4	57
6.2.4.13 fixK5	58
6.2.4.14 flag	58
6.2.4.15 flipVertical	58
6.2.4.16 goodInput	58
6.2.4.17 imageList	58
6.2.4.18 input	58
6.2.4.19 inputCapture	58
6.2.4.20 inputType	59
6.2.4.21 nrFrames	59
6.2.4.22 outputFileName	59
6.2.4.23 showUndistorted	59
6.2.4.24 squareSize	59
6.2.4.25 useFisheye	59
6.2.4.26 writeExtrinsics	60
6.2.4.27 writePoints	60
6.3 CameraCapture Class Reference	60
6.3.1 Constructor & Destructor Documentation	60
6.3.1.1 CameraCapture()	60
6.3.1.2 ~CameraCapture()	61
6.3.2 Member Function Documentation	61
6.3.2.1 grab()	61
6.3.2.2 isAlive()	61
6.3.2.3 isOpened()	62
6.3.2.4 loadCoefficients()	62
6.3.2.5 startCamera()	62
6.4 ClipperLib::Clipper Class Reference	62
6.4.1 Constructor & Destructor Documentation	63
6.4.1.1 Clipper()	64

6.4.2 Member Function Documentation	64
6.4.2.1 Execute() [1/4]	64
6.4.2.2 Execute() [2/4]	64
6.4.2.3 Execute() [3/4]	64
6.4.2.4 Execute() [4/4]	64
6.4.2.5 ExecuteInternal()	65
6.4.2.6 ReverseSolution() [1/2]	65
6.4.2.7 ReverseSolution() [2/2]	65
6.4.2.8 StrictlySimple() [1/2]	65
6.4.2.9 StrictlySimple() [2/2]	65
6.5 ClipperLib::ClipperBase Class Reference	66
6.5.1 Member Typedef Documentation	67
6.5.1.1 MinimaList	67
6.5.1.2 ScanbeamList	68
6.5.2 Constructor & Destructor Documentation	68
6.5.2.1 ClipperBase()	68
6.5.2.2 ~ClipperBase()	68
6.5.3 Member Function Documentation	68
6.5.3.1 AddBoundsToLML()	68
6.5.3.2 AddPath()	68
6.5.3.3 AddPaths()	68
6.5.3.4 Clear()	69
6.5.3.5 CreateOutRec()	69
6.5.3.6 DeleteFromAEL()	69
6.5.3.7 DisposeAllOutRecs()	69
6.5.3.8 DisposeLocalMinimaList()	69
6.5.3.9 DisposeOutRec()	69
6.5.3.10 GetBounds()	69
6.5.3.11 InsertScanbeam()	70
6.5.3.12 LocalMinimaPending()	70
6.5.3.13 PopLocalMinima()	70
6.5.3.14 PopScanbeam()	70
6.5.3.15 PreserveCollinear() [1/2]	70
6.5.3.16 PreserveCollinear() [2/2]	70
6.5.3.17 ProcessBound()	70
6.5.3.18 Reset()	71
6.5.3.19 SwapPositionsInAEL()	71
6.5.3.20 UpdateEdgeIntoAEL()	71
6.5.4 Member Data Documentation	71
6.5.4.1 m_ActiveEdges	71
6.5.4.2 m_CurrentLM	71
6.5.4.3 m_edges	71

6.5.4.4 m_HasOpenPaths	71
6.5.4.5 m_MinimalList	72
6.5.4.6 m_PolyOuts	72
6.5.4.7 m_PreserveCollinear	72
6.5.4.8 m_Scanbeam	72
6.5.4.9 m_UseFullRange	72
6.6 ClipperLib::clipperException Class Reference	72
6.6.1 Constructor & Destructor Documentation	73
6.6.1.1 clipperException()	73
6.6.1.2 ~clipperException()	73
6.6.2 Member Function Documentation	73
6.6.2.1 what()	73
6.7 ClipperLib::ClipperOffset Class Reference	73
6.7.1 Constructor & Destructor Documentation	74
6.7.1.1 ClipperOffset()	74
6.7.1.2 ~ClipperOffset()	74
6.7.2 Member Function Documentation	74
6.7.2.1 AddPath()	74
6.7.2.2 AddPaths()	74
6.7.2.3 Clear()	74
6.7.2.4 Execute() [1/2]	75
6.7.2.5 Execute() [2/2]	75
6.7.3 Member Data Documentation	75
6.7.3.1 ArcTolerance	75
6.7.3.2 MiterLimit	75
6.8 Configuration2< T1 > Class Template Reference	75
6.8.1 Detailed Description	77
6.8.2 Constructor & Destructor Documentation	77
6.8.2.1 Configuration2() [1/3]	77
6.8.2.2 Configuration2() [2/3]	77
6.8.2.3 Configuration2() [3/3]	78
6.8.3 Member Function Documentation	78
6.8.3.1 angle() [1/2]	78
6.8.3.2 angle() [2/2]	78
6.8.3.3 copy()	79
6.8.3.4 distance()	79
6.8.3.5 equal()	80
6.8.3.6 EuDistance()	80
6.8.3.7 MaDistance()	80
6.8.3.8 offset() [1/3]	81
6.8.3.9 offset() [2/3]	81
6.8.3.10 offset() [3/3]	81

6.8.3.11 offset_angle()	82
6.8.3.12 offset_x()	82
6.8.3.13 offset_y()	83
6.8.3.14 operator Configuration2< T2 >()	83
6.8.3.15 operator Point2< T2 >()	83
6.8.3.16 operator=()	84
6.8.3.17 operator==()	84
6.8.3.18 point()	84
6.8.3.19 to_string()	85
6.8.3.20 x() [1/2]	85
6.8.3.21 x() [2/2]	85
6.8.3.22 y() [1/2]	85
6.8.3.23 y() [2/2]	86
6.8.4 Friends And Related Function Documentation	86
6.8.4.1 operator<<	86
6.9 Curve< T > Class Template Reference	87
6.9.1 Detailed Description	88
6.9.2 Constructor & Destructor Documentation	88
6.9.2.1 Curve() [1/4]	88
6.9.2.2 Curve() [2/4]	88
6.9.2.3 Curve() [3/4]	89
6.9.2.4 Curve() [4/4]	89
6.9.3 Member Function Documentation	90
6.9.3.1 begin() [1/2]	90
6.9.3.2 begin() [2/2]	90
6.9.3.3 end() [1/2]	90
6.9.3.4 end() [2/2]	90
6.9.3.5 to_string()	91
6.9.4 Friends And Related Function Documentation	91
6.9.4.1 operator<<	91
6.9.5 Member Data Documentation	91
6.9.5.1 P0	91
6.9.5.2 P1	92
6.10 ClipperLib::DoublePoint Struct Reference	92
6.10.1 Constructor & Destructor Documentation	92
6.10.1.1 DoublePoint() [1/2]	92
6.10.1.2 DoublePoint() [2/2]	92
6.10.2 Member Data Documentation	92
6.10.2.1 X	93
6.10.2.2 Y	93
6.11 Dubins< T > Class Template Reference	93
6.11.1 Detailed Description	94

6.11.2 Constructor & Destructor Documentation	95
6.11.2.1 Dubins() [1/4]	95
6.11.2.2 Dubins() [2/4]	95
6.11.2.3 Dubins() [3/4]	95
6.11.2.4 Dubins() [4/4]	96
6.11.3 Member Function Documentation	96
6.11.3.1 check()	96
6.11.3.2 draw()	97
6.11.3.3 getA1()	97
6.11.3.4 getA2()	98
6.11.3.5 getA3()	98
6.11.3.6 getId()	98
6.11.3.7 getKMax()	98
6.11.3.8 length()	98
6.11.3.9 LRL()	98
6.11.3.10 LSL()	99
6.11.3.11 LSR()	99
6.11.3.12 rangeSymm()	100
6.11.3.13 RLR()	100
6.11.3.14 RSL()	100
6.11.3.15 RSR()	101
6.11.3.16 scaleFromStandard()	101
6.11.3.17 scaleToStandard()	102
6.11.3.18 shortest_path()	102
6.11.3.19 splitIt()	102
6.11.3.20 to_string()	103
6.11.4 Friends And Related Function Documentation	103
6.11.4.1 operator<<	103
6.12 DubinsArc< T1, T2 > Class Template Reference	103
6.12.1 Detailed Description	105
6.12.2 Constructor & Destructor Documentation	105
6.12.2.1 DubinsArc() [1/2]	105
6.12.2.2 DubinsArc() [2/2]	105
6.12.3 Member Function Documentation	106
6.12.3.1 draw()	106
6.12.3.2 getK()	106
6.12.3.3 length()	106
6.12.3.4 splitIt()	106
6.12.3.5 to_string()	107
6.12.4 Friends And Related Function Documentation	107
6.12.4.1 operator<<	107
6.13 DubinsSet< T > Class Template Reference	108

6.13.1 Detailed Description	108
6.13.2 Constructor & Destructor Documentation	109
6.13.2.1 DubinsSet() [1/4]	109
6.13.2.2 DubinsSet() [2/4]	109
6.13.2.3 DubinsSet() [3/4]	109
6.13.2.4 DubinsSet() [4/4]	110
6.13.3 Member Function Documentation	110
6.13.3.1 find_best()	110
6.13.3.2 getDubins()	111
6.13.3.3 getDubinses()	111
6.13.3.4 getKmax()	111
6.13.3.5 getLength()	111
6.13.3.6 getSize()	112
6.13.3.7 to_string()	112
6.13.4 Friends And Related Function Documentation	112
6.13.4.1 operator<<	112
6.14 Filter Class Reference	113
6.14.1 Detailed Description	113
6.14.2 Constructor & Destructor Documentation	114
6.14.2.1 Filter() [1/3]	114
6.14.2.2 Filter() [2/3]	114
6.14.2.3 Filter() [3/3]	114
6.14.3 Member Function Documentation	115
6.14.3.1 copy()	115
6.14.3.2 High()	115
6.14.3.3 Low()	115
6.14.3.4 operator vector< int >()	115
6.14.3.5 operator=()	116
6.14.3.6 to_string()	116
6.14.4 Friends And Related Function Documentation	116
6.14.4.1 operator<<	116
6.14.5 Member Data Documentation	117
6.14.5.1 high_h	117
6.14.5.2 high_s	117
6.14.5.3 high_v	117
6.14.5.4 low_h	117
6.14.5.5 low_s	117
6.14.5.6 low_v	118
6.15 Gate Class Reference	118
6.15.1 Constructor & Destructor Documentation	119
6.15.1.1 Gate()	119
6.15.2 Member Function Documentation	119

6.15.2.1 print()	119
6.15.2.2 toString()	120
6.16 CameraCapture::input_options_t Struct Reference	120
6.16.1 Detailed Description	120
6.16.2 Constructor & Destructor Documentation	121
6.16.2.1 input_options_t() [1/3]	121
6.16.2.2 input_options_t() [2/3]	121
6.16.2.3 input_options_t() [3/3]	121
6.16.3 Member Data Documentation	121
6.16.3.1 cameraFPS	121
6.16.3.2 frameHeight_px	121
6.16.3.3 frameWidth_px	121
6.16.3.4 nameCamera	122
6.17 ClipperLib::Int128 Class Reference	122
6.17.1 Constructor & Destructor Documentation	122
6.17.1.1 Int128() [1/3]	122
6.17.1.2 Int128() [2/3]	123
6.17.1.3 Int128() [3/3]	123
6.17.2 Member Function Documentation	123
6.17.2.1 operator "!="()	123
6.17.2.2 operator -()	123
6.17.2.3 operator -=()	123
6.17.2.4 operator >()	123
6.17.2.5 operator >=()	124
6.17.2.6 operator double()	124
6.17.2.7 operator+()	124
6.17.2.8 operator+=()	124
6.17.2.9 operator-()	124
6.17.2.10 operator<()	124
6.17.2.11 operator<=()	124
6.17.2.12 operator=()	125
6.17.2.13 operator==()	125
6.17.3 Member Data Documentation	125
6.17.3.1 hi	125
6.17.3.2 lo	125
6.18 ClipperLib::IntersectNode Struct Reference	126
6.18.1 Member Data Documentation	126
6.18.1.1 Edge1	126
6.18.1.2 Edge2	126
6.18.1.3 Pt	127
6.19 ClipperLib::IntPoint Struct Reference	127
6.19.1 Constructor & Destructor Documentation	127

6.19.1.1 IntPoint()	127
6.19.2 Friends And Related Function Documentation	127
6.19.2.1 operator"!="	128
6.19.2.2 operator=="	128
6.19.3 Member Data Documentation	128
6.19.3.1 X	128
6.19.3.2 Y	128
6.20 ClipperLib::IntRect Struct Reference	128
6.20.1 Member Data Documentation	129
6.20.1.1 bottom	129
6.20.1.2 left	129
6.20.1.3 right	129
6.20.1.4 top	129
6.21 ClipperLib::Join Struct Reference	129
6.21.1 Member Data Documentation	130
6.21.1.1 OffPt	130
6.21.1.2 OutPt1	130
6.21.1.3 OutPt2	130
6.22 ClipperLib::LocalMinimum Struct Reference	131
6.22.1 Member Data Documentation	131
6.22.1.1 LeftBound	131
6.22.1.2 RightBound	131
6.22.1.3 Y	132
6.23 ClipperLib::LocMinSorter Struct Reference	132
6.23.1 Member Function Documentation	132
6.23.1.1 operator>()	132
6.24 Mapp Class Reference	132
6.24.1 Constructor & Destructor Documentation	134
6.24.1.1 Mapp()	135
6.24.2 Member Function Documentation	136
6.24.2.1 addObject()	136
6.24.2.2 addObjects() [1 / 4]	136
6.24.2.3 addObjects() [2 / 4]	137
6.24.2.4 addObjects() [3 / 4]	137
6.24.2.5 addObjects() [4 / 4]	137
6.24.2.6 cellsFromSegment()	138
6.24.2.7 checkSegment()	138
6.24.2.8 checkSegmentCollisionWithType()	138
6.24.2.9 createMapRepresentation()	139
6.24.2.10 getGateCenter()	139
6.24.2.11 getPointType()	139
6.24.2.12 getVictimCenters()	140

6.24.2.13 imageAddPoint()	140
6.24.2.14 imageAddPoints()	140
6.24.2.15 imageAddSegment()	141
6.24.2.16 imageAddSegments()	141
6.24.2.17 matrixToString()	141
6.24.2.18 minPathNPoints()	142
6.24.2.19 minPathTwoPoints()	142
6.24.2.20 minPathTwoPointsInternal()	142
6.24.2.21 printDimensions()	143
6.24.2.22 printMap()	143
6.24.2.23 resetDistanceMap()	143
6.24.2.24 sampleNPoints() [1/2]	144
6.24.2.25 sampleNPoints() [2/2]	144
6.24.2.26 samplePointsEachNCells()	144
6.24.3 Member Data Documentation	145
6.24.3.1 baseDistance	145
6.24.3.2 borderSize	145
6.24.3.3 borderSizeDefault	145
6.24.3.4 cellSize	145
6.24.3.5 dimX	145
6.24.3.6 dimY	146
6.24.3.7 foundLimit	146
6.24.3.8 lengthX	146
6.24.3.9 lengthY	146
6.24.3.10 map	146
6.24.3.11 nPoints	146
6.24.3.12 offsetValue	146
6.24.3.13 pixX	146
6.24.3.14 pixY	147
6.24.3.15 range	147
6.24.3.16 vGates	147
6.24.3.17 vObstacles	147
6.24.3.18 vVictims	147
6.25 MyException< T > Class Template Reference	147
6.25.1 Constructor & Destructor Documentation	148
6.25.1.1 MyException()	148
6.25.2 Member Function Documentation	148
6.25.2.1 what()	148
6.25.3 Member Data Documentation	148
6.25.3.1 a	148
6.25.3.2 b	148
6.25.3.3 type	149

6.26 Object Class Reference	149
6.26.1 Member Function Documentation	150
6.26.1.1 computeCenter()	150
6.26.1.2 computeRadius()	150
6.26.1.3 getCenter()	151
6.26.1.4 getPoints()	151
6.26.1.5 getRadius()	151
6.26.1.6 insidePoly()	151
6.26.1.7 insidePolyApprox()	152
6.26.1.8 nPoints()	152
6.26.1.9 offsetting()	152
6.26.1.10 size()	153
6.26.1.11 toString()	153
6.26.2 Member Data Documentation	153
6.26.2.1 center	153
6.26.2.2 points	153
6.26.2.3 radius	153
6.27 Obstacle Class Reference	154
6.27.1 Constructor & Destructor Documentation	155
6.27.1.1 Obstacle()	155
6.27.2 Member Function Documentation	155
6.27.2.1 print()	155
6.27.2.2 toString()	155
6.28 ClipperLib::OutPt Struct Reference	156
6.28.1 Member Data Documentation	156
6.28.1.1 Idx	156
6.28.1.2 Next	156
6.28.1.3 Prev	156
6.28.1.4 Pt	157
6.29 ClipperLib::OutRec Struct Reference	157
6.29.1 Member Data Documentation	157
6.29.1.1 BottomPt	157
6.29.1.2 FirstLeft	158
6.29.1.3 Idx	158
6.29.1.4 IsHole	158
6.29.1.5 IsOpen	158
6.29.1.6 PolyNd	158
6.29.1.7 Pts	158
6.30 Point2< T > Class Template Reference	158
6.30.1 Detailed Description	160
6.30.2 Constructor & Destructor Documentation	160
6.30.2.1 Point2() [1/3]	160

6.30.2.2 Point2() [2/3]	160
6.30.2.3 Point2() [3/3]	161
6.30.3 Member Function Documentation	161
6.30.3.1 copy()	161
6.30.3.2 distance()	161
6.30.3.3 equal()	162
6.30.3.4 EuDistance()	162
6.30.3.5 MaDistance()	163
6.30.3.6 offset() [1/3]	163
6.30.3.7 offset() [2/3]	163
6.30.3.8 offset() [3/3]	164
6.30.3.9 offset_x()	164
6.30.3.10 offset_y()	164
6.30.3.11 operator cv::Point()	165
6.30.3.12 operator"!="()	165
6.30.3.13 operator<()	165
6.30.3.14 operator=()	166
6.30.3.15 operator==(())	166
6.30.3.16 th()	166
6.30.3.17 to_string()	166
6.30.3.18 x() [1/2]	167
6.30.3.19 x() [2/2]	167
6.30.3.20 y() [1/2]	167
6.30.3.21 y() [2/2]	167
6.30.4 Friends And Related Function Documentation	168
6.30.4.1 operator<<	168
6.31 ClipperLib::PolyNode Class Reference	168
6.31.1 Constructor & Destructor Documentation	169
6.31.1.1 PolyNode()	170
6.31.1.2 ~PolyNode()	170
6.31.2 Member Function Documentation	170
6.31.2.1 ChildCount()	170
6.31.2.2 GetNext()	170
6.31.2.3 IsHole()	170
6.31.2.4 IsOpen()	170
6.31.3 Friends And Related Function Documentation	170
6.31.3.1 Clipper	170
6.31.3.2 ClipperOffset	171
6.31.4 Member Data Documentation	171
6.31.4.1 Childs	171
6.31.4.2 Contour	171
6.31.4.3 Parent	171

6.32 ClipperLib::PolyTree Class Reference	171
6.32.1 Constructor & Destructor Documentation	172
6.32.1.1 ~PolyTree()	172
6.32.2 Member Function Documentation	172
6.32.2.1 Clear()	172
6.32.2.2 GetFirst()	173
6.32.2.3 Total()	173
6.32.3 Friends And Related Function Documentation	173
6.32.3.1 Clipper	173
6.33 Settings Class Reference	173
6.33.1 Detailed Description	176
6.33.2 Member Enumeration Documentation	176
6.33.2.1 COLOR	176
6.33.3 Constructor & Destructor Documentation	176
6.33.3.1 Settings()	176
6.33.3.2 ~Settings()	177
6.33.4 Member Function Documentation	177
6.33.4.1 addUnMap()	177
6.33.4.2 changeMask() [1/2]	177
6.33.4.3 changeMask() [2/2]	178
6.33.4.4 clean()	178
6.33.4.5 cleanAndRead()	178
6.33.4.6 getTemplates() [1/3]	178
6.33.4.7 getTemplates() [2/3]	179
6.33.4.8 getTemplates() [3/3]	179
6.33.4.9 maps() [1/4]	179
6.33.4.10 maps() [2/4]	181
6.33.4.11 maps() [3/4]	181
6.33.4.12 maps() [4/4]	182
6.33.4.13 readFromFile()	182
6.33.4.14 save()	182
6.33.4.15 to_string()	183
6.33.4.16 unMaps() [1/4]	183
6.33.4.17 unMaps() [2/4]	184
6.33.4.18 unMaps() [3/4]	184
6.33.4.19 unMaps() [4/4]	185
6.33.4.20 writeToFile()	185
6.33.5 Friends And Related Function Documentation	185
6.33.5.1 operator<<	185
6.33.6 Member Data Documentation	186
6.33.6.1 blackMask	186
6.33.6.2 blueMask	186

6.33.6.3 calibrationFile	186
6.33.6.4 convexHullFile	186
6.33.6.5 greenMask	187
6.33.6.6 intrinsicCalibrationFile	187
6.33.6.7 kernelSide	187
6.33.6.8 mapsFolder	187
6.33.6.9 mapsNames	187
6.33.6.10 mapsUnNames	187
6.33.6.11 redMask	188
6.33.6.12 robotMask	188
6.33.6.13 templates	188
6.33.6.14 templatesFolder	188
6.33.6.15 victimMask	188
6.33.6.16 whiteMask	188
6.34 ClipperLib::TEdge Struct Reference	189
6.34.1 Member Data Documentation	189
6.34.1.1 Bot	190
6.34.1.2 Curr	190
6.34.1.3 Dx	190
6.34.1.4 Next	190
6.34.1.5 NextInAEL	190
6.34.1.6 NextInLML	190
6.34.1.7 NextInSEL	190
6.34.1.8 OutIdx	190
6.34.1.9 PolyTyp	191
6.34.1.10 Prev	191
6.34.1.11 PrevInAEL	191
6.34.1.12 PrevInSEL	191
6.34.1.13 Side	191
6.34.1.14 Top	191
6.34.1.15 WindCnt	191
6.34.1.16 WindCnt2	191
6.34.1.17 WindDelta	192
6.35 Tuple< T > Class Template Reference	192
6.35.1 Detailed Description	193
6.35.2 Constructor & Destructor Documentation	193
6.35.2.1 Tuple() [1/2]	193
6.35.2.2 Tuple() [2/2]	194
6.35.3 Member Function Documentation	194
6.35.3.1 add()	194
6.35.3.2 addIfNot()	194
6.35.3.3 ahead()	195

6.35.3.4 begin() [1/2]	195
6.35.3.5 begin() [2/2]	195
6.35.3.6 distance()	195
6.35.3.7 end() [1/2]	196
6.35.3.8 end() [2/2]	196
6.35.3.9 equal()	196
6.35.3.10 eraseAll()	196
6.35.3.11 EuDistance()	196
6.35.3.12 find()	197
6.35.3.13 get()	197
6.35.3.14 MaDistance()	197
6.35.3.15 mul() [1/2]	198
6.35.3.16 mul() [2/2]	198
6.35.3.17 operator *()	198
6.35.3.18 operator *=()	198
6.35.3.19 operator std::string()	198
6.35.3.20 operator vector< T1 >()	199
6.35.3.21 operator+()	199
6.35.3.22 operator+=()	199
6.35.3.23 operator==()	199
6.35.3.24 operator[]()	199
6.35.3.25 remove()	200
6.35.3.26 set()	200
6.35.3.27 size()	200
6.35.3.28 sum() [1/2]	201
6.35.3.29 sum() [2/2]	201
6.35.3.30 to_std_string()	201
6.35.3.31 to_string()	201
6.35.4 Friends And Related Function Documentation	201
6.35.4.1 operator<<	201
6.36 Victim Class Reference	202
6.36.1 Constructor & Destructor Documentation	203
6.36.1.1 Victim()	203
6.36.2 Member Function Documentation	203
6.36.2.1 getValue()	203
6.36.2.2 print()	204
6.36.2.3 setValue()	204
6.36.2.4 toString()	204
6.36.3 Member Data Documentation	204
6.36.3.1 value	204

7.1 src/calibration.cc File Reference	205
7.1.1 Function Documentation	206
7.1.1.1 calcBoardCornerPositions()	206
7.1.1.2 calibration()	206
7.1.1.3 computeReprojectionErrors()	206
7.1.1.4 read()	207
7.1.1.5 runCalibration()	207
7.1.1.6 runCalibrationAndSave()	208
7.1.1.7 saveCameraParams()	209
7.2 src/camera_capture.cc File Reference	209
7.2.1 Macro Definition Documentation	210
7.2.1.1 SDEBUG	210
7.3 src/clipper.cc File Reference	210
7.3.1 Macro Definition Documentation	213
7.3.1.1 HORIZONTAL	213
7.3.1.2 NEAR_ZERO	213
7.3.1.3 TOLERANCE	213
7.4 src/configure.cc File Reference	213
7.4.1 Function Documentation	214
7.4.1.1 acquireImage()	214
7.4.1.2 configure()	214
7.4.1.3 on_high_h_thresh_trackbar()	214
7.4.1.4 on_high_s_thresh_trackbar()	215
7.4.1.5 on_high_v_thresh_trackbar()	215
7.4.1.6 on_low_h_thresh_trackbar()	215
7.4.1.7 on_low_s_thresh_trackbar()	215
7.4.1.8 on_low_v_thresh_trackbar()	215
7.4.1.9 show_all_conditions()	215
7.4.1.10 update_trackers()	216
7.4.2 Variable Documentation	216
7.4.2.1 filter	216
7.5 src/detection.cc File Reference	216
7.5.1 Macro Definition Documentation	217
7.5.1.1 EPS_CURVE	217
7.5.1.2 MIN_AREA_SIZE	218
7.5.2 Function Documentation	218
7.5.2.1 _compare()	218
7.5.2.2 computeConversionParameters()	218
7.5.2.3 crop_number_section()	218
7.5.2.4 detection()	218
7.5.2.5 erode_dilation()	219
7.5.2.6 find_contours()	219

7.5.2.7 load_number_template()	219
7.5.2.8 localize() [1/2]	219
7.5.2.9 localize() [2/2]	220
7.5.2.10 number_recognition()	220
7.5.2.11 save_convex_hull()	220
7.5.2.12 shape_detection()	220
7.5.3 Variable Documentation	221
7.5.3.1 robotShape	221
7.5.3.2 templates	221
7.6 src/dubins.cc File Reference	221
7.7 src/include/calibration.hh File Reference	221
7.7.1 Detailed Description	223
7.7.2 Enumeration Type Documentation	223
7.7.2.1 anonymous enum	223
7.7.3 Function Documentation	223
7.7.3.1 calibration()	223
7.7.3.2 runCalibrationAndSave()	223
7.7.4 Variable Documentation	224
7.7.4.1 sett	224
7.8 src/include/camera_capture.hh File Reference	224
7.9 src/include/clipper.hh File Reference	225
7.9.1 Macro Definition Documentation	227
7.9.1.1 CLIPPER_VERSION	228
7.9.1.2 use_lines	228
7.10 src/include/configure.hh File Reference	228
7.10.1 Function Documentation	229
7.10.1.1 acquireImage()	229
7.10.1.2 configure()	229
7.10.1.3 show_all_conditions()	229
7.11 src/include/detection.hh File Reference	230
7.11.1 Enumeration Type Documentation	231
7.11.1.1 COLOR_TYPE	231
7.11.2 Function Documentation	231
7.11.2.1 computeConversionParameters()	232
7.11.2.2 crop_number_section()	232
7.11.2.3 detection()	232
7.11.2.4 erode_dilation()	232
7.11.2.5 find_contours()	233
7.11.2.6 load_number_template()	233
7.11.2.7 localize() [1/2]	233
7.11.2.8 localize() [2/2]	233
7.11.2.9 number_recognition()	233

7.11.2.10 save_convex_hull()	234
7.11.2.11 shape_detection()	234
7.12 src/include/draw.hh File Reference	234
7.12.1 Typedef Documentation	235
7.12.1.1 int	235
7.13 src/include/dubins.hh File Reference	236
7.13.1 Macro Definition Documentation	237
7.13.1.1 DELTA	237
7.13.1.2 KMAX	237
7.13.1.3 PIECE_LENGTH	238
7.13.1.4 PREC	238
7.13.2 Function Documentation	238
7.13.2.1 circline()	238
7.13.2.2 disp()	238
7.13.2.3 reduce_points()	239
7.13.2.4 sinc()	239
7.13.2.5 toBase()	239
7.13.3 Variable Documentation	240
7.13.3.1 countTries	240
7.13.3.2 elapsedArcs	240
7.13.3.3 elapsedBest	240
7.13.3.4 elapsedCheck	240
7.13.3.5 elapsedCirc	241
7.13.3.6 elapsedLRL	241
7.13.3.7 elapsedLSL	241
7.13.3.8 elapsedLSR	241
7.13.3.9 elapsedPrimitives	241
7.13.3.10 elapsedRLR	241
7.13.3.11 elapsedRSL	241
7.13.3.12 elapsedRSR	241
7.13.3.13 elapsedScale	242
7.13.3.14 elapsedSet	242
7.13.3.15 elapsedVar	242
7.14 src/include/filter.hh File Reference	242
7.15 src/include/map.hh File Reference	243
7.15.1 Enumeration Type Documentation	244
7.15.1.1 OBJ_TYPE	244
7.16 src/include/math.h File Reference	244
7.16.1 Macro Definition Documentation	246
7.16.1.1 A_180	246
7.16.1.2 A_2PI	246
7.16.1.3 A_360	246

7.16.1.4 A_90	246
7.16.1.5 A_DEG_NULL	247
7.16.1.6 A_PI	247
7.16.1.7 A_PI2	247
7.16.1.8 A_RAD_NULL	247
7.16.1.9 DInf	247
7.16.1.10 Epsi	247
7.16.1.11 tupleConstIter	247
7.16.1.12 tupleIter	248
7.16.2 Enumeration Type Documentation	248
7.16.2.1 DISTANCE_TYPE	248
7.16.3 Function Documentation	249
7.16.3.1 equal()	249
7.16.3.2 pow2()	249
7.16.4 Variable Documentation	249
7.16.4.1 DEGTORAD	249
7.16.4.2 RADTO DEG	250
7.17 src/include/objects.hh File Reference	250
7.18 src/include/planning.hh File Reference	251
7.18.1 Function Documentation	251
7.18.1.1 createMapp()	252
7.18.1.2 loadVP()	252
7.18.1.3 loadVVP()	252
7.18.1.4 planning()	252
7.19 src/include/settings.hh File Reference	253
7.19.1 Variable Documentation	253
7.19.1.1 sett	254
7.20 src/include/unwrapping.hh File Reference	254
7.20.1 Function Documentation	255
7.20.1.1 find_rect()	255
7.20.1.2 loadCoefficients()	255
7.20.1.3 unwrapping()	255
7.21 src/include/utils.hh File Reference	256
7.21.1 Macro Definition Documentation	257
7.21.1.1 COUT	257
7.21.1.2 INFO	258
7.21.1.3 NAME	258
7.21.2 Typedef Documentation	258
7.21.2.1 Clock	258
7.21.3 Enumeration Type Documentation	258
7.21.3.1 EXCEPTION_TYPE	258
7.21.4 Function Documentation	258

7.21.4.1 my_imshow()	258
7.21.4.2 mywaitkey() [1/2]	259
7.21.4.3 mywaitkey() [2/2]	259
7.22 src/map.cc File Reference	259
7.23 src/math.cc File Reference	260
7.24 src/objects.cc File Reference	260
7.25 src/planning.cc File Reference	260
7.25.1 Function Documentation	261
7.25.1.1 createMapp()	261
7.25.1.2 loadVP()	261
7.25.1.3 loadVVP()	261
7.25.1.4 planning()	262
7.26 src/run/calibration_run.cc File Reference	262
7.26.1 Function Documentation	262
7.26.1.1 main()	262
7.27 src/run/detection_run.cc File Reference	262
7.27.1 Function Documentation	263
7.27.1.1 main()	263
7.28 src/run/main.cc File Reference	263
7.28.1 Function Documentation	263
7.28.1.1 main()	263
7.28.2 Variable Documentation	264
7.28.2.1 sett	264
7.29 src/run/planning_run.cc File Reference	264
7.29.1 Function Documentation	264
7.29.1.1 main()	264
7.30 src/run/unwrapping_run.cc File Reference	264
7.30.1 Function Documentation	265
7.30.1.1 main()	265
7.31 src/settings.cc File Reference	265
7.31.1 Macro Definition Documentation	265
7.31.1.1 NPOS	265
7.31.2 Function Documentation	266
7.31.2.1 getFiles()	266
7.31.2.2 vecToFile()	266
7.32 src/unwrapping.cc File Reference	266
7.32.1 Macro Definition Documentation	267
7.32.1.1 AREA_MIN	267
7.32.1.2 AREA_RATIO	267
7.32.2 Function Documentation	267
7.32.2.1 distance()	267
7.32.2.2 find_rect()	268

7.32.2.3 loadCoefficients()	268
7.32.2.4 unwrapping()	269
7.33 src/utls.cc File Reference	269
7.33.1 Function Documentation	269
7.33.1.1 my_imshow()	269
7.33.1.2 mywaitkey() [1/2]	270
7.33.1.3 mywaitkey() [2/2]	270
Index	271

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

CHRONO	9
ClipperLib	10
DW	32
timeutils	33

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Angle	35
CalSettings	51
ClipperLib::ClipperBase	66
ClipperLib::Clipper	62
ClipperLib::ClipperOffset	73
Configuration2< T1 >	75
Configuration2< double >	75
Configuration2< T >	75
Configuration2< T2 >	75
Curve< T >	87
Dubins< T >	93
Curve< double >	87
DubinsArc< T >	103
Curve< T2 >	87
DubinsArc< T1, T2 >	103
ClipperLib::DoublePoint	92
DubinsSet< T >	108
exception	
MyException< T >	147
exception	
ClipperLib::clipperException	72
Filter	113
CameraCapture::input_options_t	120
ClipperLib::Int128	122
ClipperLib::IntersectNode	126
ClipperLib::IntPoint	127
ClipperLib::IntRect	128
ClipperLib::Join	129
ClipperLib::LocalMinimum	131
ClipperLib::LocMinSorter	132
Mapp	132
Object	149
Gate	118
Obstacle	154

Victim	202
ClipperLib::OutPt	156
ClipperLib::OutRec	157
Point2< T >	158
Point2< double >	158
Point2< int >	158
Point2< T1 >	158
Point2< T2 >	158
ClipperLib::PolyNode	168
ClipperLib::PolyTree	171
Settings	173
ClipperLib::TEdge	189
Tuple< T >	192
Tuple< Dubins< T > >	192
Tuple< string >	192
VideoCapture	
CameraCapture	60

Chapter 3

Class Index

3.1 Class List

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

Angle	This class allows to save and handle angles. It supports DEG and RAD, operations such as addition and subtraction with operators overloading, conversion from RAD to DEG and viceversa and normalization of the angle	35
CalSettings		51
CameraCapture		60
ClipperLib::Clipper		62
ClipperLib::ClipperBase		66
ClipperLib::clipperException		72
ClipperLib::ClipperOffset		73
Configuration2< T1 >	This class stores a configuration, that is a point and an angle	75
Curve< T >		87
ClipperLib::DoublePoint		92
Dubins< T >	Class to store a Dubins curve. This class inherits from Curve and is composed of three DubinsArc	93
DubinsArc< T1, T2 >	Class to store a maneuver of Dubins . It inherits from Curve . Since each Dubins is formed of atmost 3 maneuvers, this class is meant to store one of this maneuver, which can be L, R or S respectively Left, Right, Straight	103
DubinsSet< T >	Given a set of point, compute the shortest set of Dubins that allows to go from start to end through all points	108
Filter		113
Gate		118
CameraCapture::input_options_t	Structure for store the input option for the class CameraCapture	120
ClipperLib::Int128		122
ClipperLib::IntersectNode		126
ClipperLib::IntPoint		127
ClipperLib::IntRect		128
ClipperLib::Join		129
ClipperLib::LocalMinimum		131
ClipperLib::LocMinSorter		132

Mapp	132
MyException< T >	147
Object	149
Obstacle	154
ClipperLib::OutPt	156
ClipperLib::OutRec	157
Point2< T >	158
Class that stores two value to construct a point in 2D. The value is saved in a Tuple	
ClipperLib::PolyNode	168
ClipperLib::PolyTree	171
Settings	173
ClipperLib::TEdge	189
Tuple< T >	192
Victim	202

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/calibration.cc	205
src/camera_capture.cc	209
src/clipper.cc	210
src/configure.cc	213
src/detection.cc	216
src/dubins.cc	221
src/map.cc	259
src/maths.cc	260
src/objects.cc	260
src/planning.cc	260
src/settings.cc	265
src/unwrapping.cc	266
src/utils.cc	269
src/include/calibration.hh	
Library for calibration	221
src/include/camera_capture.hh	224
src/include/clipper.hh	225
src/include/configure.hh	228
src/include/detection.hh	230
src/include/draw.hh	234
src/include/dubins.hh	236
src/include/filter.hh	242
src/include/map.hh	243
src/include/maths.hh	244
src/include/objects.hh	250
src/include/planning.hh	251
src/include/settings.hh	253
src/include/unwrapping.hh	254
src/include/utils.hh	256
src/run/calibration_run.cc	262
src/run/detection_run.cc	262
src/run/main.cc	263
src/run/planning_run.cc	264
src/run/unwrapping_run.cc	264

Chapter 5

Namespace Documentation

5.1 CHRONO Namespace Reference

Enumerations

- enum `TIME_TYPE` { `SEC`, `MSEC`, `MUSEC`, `NSEC` }

Functions

- string `getType` (`TIME_TYPE` type, string ret="")
- double `getElapsed` (Clock::time_point start, Clock::time_point stop, `TIME_TYPE` type=`MUSEC`)
- string `getElapsed` (Clock::time_point start, Clock::time_point stop, string ret, `TIME_TYPE` type=`MUSEC`)

5.1.1 Enumeration Type Documentation

5.1.1.1 TIME_TYPE

enum `CHRONO::TIME_TYPE`

Enumerator

SEC	
MSEC	
MUSEC	
NSEC	

5.1.2 Function Documentation

5.1.2.1 getElapsed() [1/2]

```
double CHRONO::getElapsed (
    Clock::time_point start,
    Clock::time_point stop,
    TIME_TYPE type = MUSEC ) [inline]
```

5.1.2.2 getElapsed() [2/2]

```
string CHRONO::getElapsed (
    Clock::time_point start,
    Clock::time_point stop,
    string ret,
    TIME_TYPE type = MUSEC ) [inline]
```

5.1.2.3 getType()

```
string CHRONO::getType (
    TIME_TYPE type,
    string ret = "" ) [inline]
```

5.2 ClipperLib Namespace Reference**Classes**

- class [Clipper](#)
- class [ClipperBase](#)
- class [clipperException](#)
- class [ClipperOffset](#)
- struct [DoublePoint](#)
- class [Int128](#)
- struct [IntersectNode](#)
- struct [IntPoint](#)
- struct [IntRect](#)
- struct [Join](#)
- struct [LocalMinimum](#)
- struct [LocMinSorter](#)
- struct [OutPt](#)
- struct [OutRec](#)
- class [PolyNode](#)
- class [PolyTree](#)
- struct [TEdge](#)

Typedefs

- typedef signed long long [clnt](#)
- typedef signed long long [long64](#)
- typedef unsigned long long [ulong64](#)
- typedef std::vector< [IntPoint](#) > [Path](#)
- typedef std::vector< [Path](#) > [Paths](#)
- typedef std::vector< [PolyNode](#) * > [PolyNodes](#)
- typedef std::vector< [OutRec](#) * > [PolyOutList](#)
- typedef std::vector< [TEdge](#) * > [EdgeList](#)
- typedef std::vector< [Join](#) * > [JoinList](#)
- typedef std::vector< [IntersectNode](#) * > [IntersectList](#)

Enumerations

- enum [Direction](#) { [dRightToLeft](#), [dLeftToRight](#) }
- enum [NodeType](#) { [ntAny](#), [ntOpen](#), [ntClosed](#) }
- enum [ClipType](#) { [ctIntersection](#), [ctUnion](#), [ctDifference](#), [ctXor](#) }
- enum [PolyType](#) { [ptSubject](#), [ptClip](#) }
- enum [PolyFillType](#) { [pftEvenOdd](#), [pftNonZero](#), [pftPositive](#), [pftNegative](#) }
- enum [InitOptions](#) { [ioReverseSolution](#) = 1, [ioStrictlySimple](#) = 2, [ioPreserveCollinear](#) = 4 }
- enum [JoinType](#) { [jtSquare](#), [jtRound](#), [jtMiter](#) }
- enum [EndType](#) { [etClosedPolygon](#), [etClosedLine](#), [etOpenButt](#), [etOpenSquare](#), [etOpenRound](#) }
- enum [EdgeSide](#) { [esLeft](#) = 1, [esRight](#) = 2 }

Functions

- [clnt Round](#) (double val)
- [clnt Abs](#) (clnt val)
- [Int128 Int128Mul](#) (long64 lhs, long64 rhs)
- bool [Orientation](#) (const [Path](#) &poly)
- double [Area](#) (const [Path](#) &poly)
- double [Area](#) (const [OutPt](#) *op)
- double [Area](#) (const [OutRec](#) &outRec)
- bool [PointIsVertex](#) (const [IntPoint](#) &Pt, [OutPt](#) *pp)
- int [PointInPolygon](#) (const [IntPoint](#) &pt, const [Path](#) &path)
- int [PointInPolygon](#) (const [IntPoint](#) &pt, [OutPt](#) *op)
- bool [Poly2ContainsPoly1](#) ([OutPt](#) *OutPt1, [OutPt](#) *OutPt2)
- bool [SlopesEqual](#) (const [TEdge](#) &e1, const [TEdge](#) &e2, bool UseFullInt64Range)
- bool [SlopesEqual](#) (const [IntPoint](#) pt1, const [IntPoint](#) pt2, const [IntPoint](#) pt3, bool UseFullInt64Range)
- bool [SlopesEqual](#) (const [IntPoint](#) pt1, const [IntPoint](#) pt2, const [IntPoint](#) pt3, const [IntPoint](#) pt4, bool UseFullInt64Range)
- bool [IsHorizontal](#) ([TEdge](#) &e)
- double [GetDx](#) (const [IntPoint](#) pt1, const [IntPoint](#) pt2)
- void [SetDx](#) ([TEdge](#) &e)
- void [SwapSides](#) ([TEdge](#) &Edge1, [TEdge](#) &Edge2)
- void [SwapPolyIndexes](#) ([TEdge](#) &Edge1, [TEdge](#) &Edge2)
- clnt [TopX](#) ([TEdge](#) &edge, const clnt currentY)
- void [IntersectPoint](#) ([TEdge](#) &Edge1, [TEdge](#) &Edge2, [IntPoint](#) &ip)
- void [ReversePolyPtLinks](#) ([OutPt](#) *pp)
- void [DisposeOutPts](#) ([OutPt](#) *&pp)

- void [InitEdge](#) ([TEdge](#) *e, [TEdge](#) *eNext, [TEdge](#) *ePrev, const [IntPoint](#) &Pt)
- void [InitEdge2](#) ([TEdge](#) &e, [PolyType](#) Pt)
- [TEdge](#) * [RemoveEdge](#) ([TEdge](#) *e)
- void [ReverseHorizontal](#) ([TEdge](#) &e)
- void [SwapPoints](#) ([IntPoint](#) &pt1, [IntPoint](#) &pt2)
- bool [GetOverlapSegment](#) ([IntPoint](#) pt1a, [IntPoint](#) pt1b, [IntPoint](#) pt2a, [IntPoint](#) pt2b, [IntPoint](#) &pt1, [IntPoint](#) &pt2)
- bool [FirstIsBottomPt](#) (const [OutPt](#) *btmPt1, const [OutPt](#) *btmPt2)
- [OutPt](#) * [GetBottomPt](#) ([OutPt](#) *pp)
- bool [Pt2IsBetweenPt1AndPt3](#) (const [IntPoint](#) pt1, const [IntPoint](#) pt2, const [IntPoint](#) pt3)
- bool [HorzSegmentsOverlap](#) ([clnt](#) seg1a, [clnt](#) seg1b, [clnt](#) seg2a, [clnt](#) seg2b)
- void [RangeTest](#) (const [IntPoint](#) &Pt, bool &useFullRange)
- [TEdge](#) * [FindNextLocMin](#) ([TEdge](#) *E)
- [OutRec](#) * [GetLowermostRec](#) ([OutRec](#) *outRec1, [OutRec](#) *outRec2)
- bool [OutRec1RightOfOutRec2](#) ([OutRec](#) *outRec1, [OutRec](#) *outRec2)
- bool [IsMinima](#) ([TEdge](#) *e)
- bool [IsMaxima](#) ([TEdge](#) *e, const [clnt](#) Y)
- bool [IsIntermediate](#) ([TEdge](#) *e, const [clnt](#) Y)
- [TEdge](#) * [GetMaximaPair](#) ([TEdge](#) *e)
- [TEdge](#) * [GetMaximaPairEx](#) ([TEdge](#) *e)
- [TEdge](#) * [GetNextInAEL](#) ([TEdge](#) *e, [Direction](#) dir)
- void [GetHorzDirection](#) ([TEdge](#) &HorzEdge, [Direction](#) &Dir, [clnt](#) &Left, [clnt](#) &Right)
- bool [IntersectListSort](#) ([IntersectNode](#) *node1, [IntersectNode](#) *node2)
- bool [EdgesAdjacent](#) (const [IntersectNode](#) &inode)
- int [PointCount](#) ([OutPt](#) *Pts)
- void [SwapIntersectNodes](#) ([IntersectNode](#) &int1, [IntersectNode](#) &int2)
- bool [E2InsertsBeforeE1](#) ([TEdge](#) &e1, [TEdge](#) &e2)
- bool [GetOverlap](#) (const [clnt](#) a1, const [clnt](#) a2, const [clnt](#) b1, const [clnt](#) b2, [clnt](#) &Left, [clnt](#) &Right)
- void [UpdateOutPtIdxs](#) ([OutRec](#) &outrec)
- [OutPt](#) * [DupOutPt](#) ([OutPt](#) *outPt, bool InsertAfter)
- bool [JoinHorz](#) ([OutPt](#) *op1, [OutPt](#) *op1b, [OutPt](#) *op2, [OutPt](#) *op2b, const [IntPoint](#) Pt, bool DiscardLeft)
- static [OutRec](#) * [ParseFirstLeft](#) ([OutRec](#) *FirstLeft)
- [DoublePoint](#) [GetUnitNormal](#) (const [IntPoint](#) &pt1, const [IntPoint](#) &pt2)
- void [ReversePath](#) ([Path](#) &p)
- void [ReversePaths](#) ([Paths](#) &p)
- void [SimplifyPolygon](#) (const [Path](#) &in_poly, [Paths](#) &out_polys, [PolyFillType](#) fillType)
- void [SimplifyPolygons](#) (const [Paths](#) &in_polys, [Paths](#) &out_polys, [PolyFillType](#) fillType)
- void [SimplifyPolygons](#) ([Paths](#) &polys, [PolyFillType](#) fillType)
- double [DistanceSqr](#) (const [IntPoint](#) &pt1, const [IntPoint](#) &pt2)
- double [DistanceFromLineSqr](#) (const [IntPoint](#) &pt, const [IntPoint](#) &ln1, const [IntPoint](#) &ln2)
- bool [SlopesNearCollinear](#) (const [IntPoint](#) &pt1, const [IntPoint](#) &pt2, const [IntPoint](#) &pt3, double distSqr)
- bool [PointsAreClose](#) ([IntPoint](#) pt1, [IntPoint](#) pt2, double distSqr)
- [OutPt](#) * [ExcludeOp](#) ([OutPt](#) *op)
- void [CleanPolygon](#) (const [Path](#) &in_poly, [Path](#) &out_poly, double distance)
- void [CleanPolygon](#) ([Path](#) &poly, double distance)
- void [CleanPolygons](#) (const [Paths](#) &in_polys, [Paths](#) &out_polys, double distance)
- void [CleanPolygons](#) ([Paths](#) &polys, double distance)
- void [Minkowski](#) (const [Path](#) &poly, const [Path](#) &path, [Paths](#) &solution, bool isSum, bool isClosed)
- void [MinkowskiSum](#) (const [Path](#) &pattern, const [Path](#) &path, [Paths](#) &solution, bool pathsClosed)
- void [TranslatePath](#) (const [Path](#) &input, [Path](#) &output, const [IntPoint](#) delta)
- void [MinkowskiSum](#) (const [Path](#) &pattern, const [Paths](#) &paths, [Paths](#) &solution, bool pathsClosed)
- void [MinkowskiDiff](#) (const [Path](#) &poly1, const [Path](#) &poly2, [Paths](#) &solution)
- void [AddPolyNodeToPaths](#) (const [PolyNode](#) &polynode, [NodeType](#) nodetype, [Paths](#) &paths)
- void [PolyTreeToPaths](#) (const [PolyTree](#) &polytree, [Paths](#) &paths)
- void [ClosedPathsFromPolyTree](#) (const [PolyTree](#) &polytree, [Paths](#) &paths)

- void [OpenPathsFromPolyTree](#) ([PolyTree](#) &polytree, [Paths](#) &paths)
- std::ostream & [operator<<](#) (std::ostream &s, const [IntPoint](#) &p)
- std::ostream & [operator<<](#) (std::ostream &s, const [Path](#) &p)
- std::ostream & [operator<<](#) (std::ostream &s, const [Paths](#) &p)
- [Path](#) & [operator<<](#) ([Path](#) &poly, const [IntPoint](#) &p)
- [Paths](#) & [operator<<](#) ([Paths](#) &polys, const [Path](#) &p)

Variables

- static double const [pi](#) = 3.141592653589793238
- static double const [two_pi](#) = [pi](#) *2
- static double const [def_arc_tolerance](#) = 0.25
- static [int](#) const [Unassigned](#) = -1
- static [int](#) const [Skip](#) = -2
- static [cInt](#) const [loRange](#) = 0x3FFFFFFF
- static [cInt](#) const [hiRange](#) = 0x3FFFFFFFFFFFFFFFLL

5.2.1 Typedef Documentation

5.2.1.1 cInt

```
typedef signed long long ClipperLib::cInt
```

5.2.1.2 EdgeList

```
typedef std::vector< TEdge\* > ClipperLib::EdgeList
```

5.2.1.3 IntersectList

```
typedef std::vector< IntersectNode\* > ClipperLib::IntersectList
```

5.2.1.4 JoinList

```
typedef std::vector< Join\* > ClipperLib::JoinList
```

5.2.1.5 long64

```
typedef signed long long ClipperLib::long64
```

5.2.1.6 Path

```
typedef std::vector< IntPoint > ClipperLib::Path
```

5.2.1.7 Paths

```
typedef std::vector< Path > ClipperLib::Paths
```

5.2.1.8 PolyNodes

```
typedef std::vector< PolyNode* > ClipperLib::PolyNodes
```

5.2.1.9 PolyOutList

```
typedef std::vector< OutRec* > ClipperLib::PolyOutList
```

5.2.1.10 ulong64

```
typedef unsigned long long ClipperLib::ulong64
```

5.2.2 Enumeration Type Documentation

5.2.2.1 ClipType

```
enum ClipperLib::ClipType
```

Enumerator

ctIntersection	
ctUnion	
ctDifference	
ctXor	

5.2.2.2 Direction

enum `ClipperLib::Direction`

Enumerator

dRightToLeft	
dLeftToRight	

5.2.2.3 EdgeSide

enum `ClipperLib::EdgeSide`

Enumerator

esLeft	
esRight	

5.2.2.4 EndType

enum `ClipperLib::EndType`

Enumerator

etClosedPolygon	
etClosedLine	
etOpenButt	
etOpenSquare	
etOpenRound	

5.2.2.5 InitOptions

enum `ClipperLib::InitOptions`

Enumerator

ioReverseSolution	
ioStrictlySimple	
ioPreserveCollinear	

5.2.2.6 JoinType

enum `ClipperLib::JoinType`

Enumerator

jtSquare	
jtRound	
jtMiter	

5.2.2.7 NodeType

enum `ClipperLib::NodeType`

Enumerator

ntAny	
ntOpen	
ntClosed	

5.2.2.8 PolyFillType

enum `ClipperLib::PolyFillType`

Enumerator

pftEvenOdd	
pftNonZero	
pftPositive	
pftNegative	

5.2.2.9 PolyType

```
enum ClipperLib::PolyType
```

Enumerator

ptSubject	
ptClip	

5.2.3 Function Documentation

5.2.3.1 Abs()

```
cInt ClipperLib::Abs (  
    cInt val ) [inline]
```

5.2.3.2 AddPolyNodeToPaths()

```
void ClipperLib::AddPolyNodeToPaths (  
    const PolyNode & polynode,  
    NodeType nodetype,  
    Paths & paths )
```

5.2.3.3 Area() [1/3]

```
double ClipperLib::Area (  
    const Path & poly )
```

5.2.3.4 Area() [2/3]

```
double ClipperLib::Area (  
    const OutPt * op )
```

5.2.3.5 Area() [3/3]

```
double ClipperLib::Area (
    const OutRec & outRec )
```

5.2.3.6 CleanPolygon() [1/2]

```
void ClipperLib::CleanPolygon (
    const Path & in_poly,
    Path & out_poly,
    double distance )
```

5.2.3.7 CleanPolygon() [2/2]

```
void ClipperLib::CleanPolygon (
    Path & poly,
    double distance )
```

5.2.3.8 CleanPolygons() [1/2]

```
void ClipperLib::CleanPolygons (
    const Paths & in_polys,
    Paths & out_polys,
    double distance )
```

5.2.3.9 CleanPolygons() [2/2]

```
void ClipperLib::CleanPolygons (
    Paths & polys,
    double distance )
```

5.2.3.10 ClosedPathsFromPolyTree()

```
void ClipperLib::ClosedPathsFromPolyTree (
    const PolyTree & polytree,
    Paths & paths )
```

5.2.3.11 DisposeOutPts()

```
void ClipperLib::DisposeOutPts (
    OutPt *& pp )
```

5.2.3.12 DistanceFromLineSqrd()

```
double ClipperLib::DistanceFromLineSqrd (
    const IntPoint & pt,
    const IntPoint & ln1,
    const IntPoint & ln2 )
```

5.2.3.13 DistanceSqrd()

```
double ClipperLib::DistanceSqrd (
    const IntPoint & pt1,
    const IntPoint & pt2 ) [inline]
```

5.2.3.14 DupOutPt()

```
OutPt* ClipperLib::DupOutPt (
    OutPt * outPt,
    bool InsertAfter )
```

5.2.3.15 E2InsertsBeforeE1()

```
bool ClipperLib::E2InsertsBeforeE1 (
    TEdge & e1,
    TEdge & e2 ) [inline]
```

5.2.3.16 EdgesAdjacent()

```
bool ClipperLib::EdgesAdjacent (
    const IntersectNode & inode ) [inline]
```

5.2.3.17 ExcludeOp()

```
OutPt* ClipperLib::ExcludeOp (
    OutPt * op )
```

5.2.3.18 FindNextLocMin()

```
TEdge* ClipperLib::FindNextLocMin (
    TEdge * E )
```

5.2.3.19 FirstIsBottomPt()

```
bool ClipperLib::FirstIsBottomPt (
    const OutPt * btmPt1,
    const OutPt * btmPt2 )
```

5.2.3.20 GetBottomPt()

```
OutPt* ClipperLib::GetBottomPt (
    OutPt * pp )
```

5.2.3.21 GetDx()

```
double ClipperLib::GetDx (
    const IntPoint pt1,
    const IntPoint pt2 ) [inline]
```

5.2.3.22 GetHorzDirection()

```
void ClipperLib::GetHorzDirection (
    TEdge & HorzEdge,
    Direction & Dir,
    cInt & Left,
    cInt & Right )
```


5.2.3.23 GetLowermostRec()

```
OutRec* ClipperLib::GetLowermostRec (
    OutRec * outRec1,
    OutRec * outRec2 )
```

5.2.3.24 GetMaximaPair()

```
TEdge* ClipperLib::GetMaximaPair (
    TEdge * e )
```

5.2.3.25 GetMaximaPairEx()

```
TEdge* ClipperLib::GetMaximaPairEx (
    TEdge * e )
```

5.2.3.26 GetNextInAEL()

```
TEdge* ClipperLib::GetNextInAEL (
    TEdge * e,
    Direction dir )
```

5.2.3.27 GetOverlap()

```
bool ClipperLib::GetOverlap (
    const cInt a1,
    const cInt a2,
    const cInt b1,
    const cInt b2,
    cInt & Left,
    cInt & Right )
```

5.2.3.28 GetOverlapSegment()

```
bool ClipperLib::GetOverlapSegment (
    IntPoint pt1a,
    IntPoint pt1b,
    IntPoint pt2a,
    IntPoint pt2b,
    IntPoint & pt1,
    IntPoint & pt2 )
```

5.2.3.29 GetUnitNormal()

```
DoublePoint ClipperLib::GetUnitNormal (
    const IntPoint & pt1,
    const IntPoint & pt2 )
```

5.2.3.30 HorzSegmentsOverlap()

```
bool ClipperLib::HorzSegmentsOverlap (
    cInt seg1a,
    cInt seg1b,
    cInt seg2a,
    cInt seg2b )
```

5.2.3.31 InitEdge()

```
void ClipperLib::InitEdge (
    TEdge * e,
    TEdge * eNext,
    TEdge * ePrev,
    const IntPoint & Pt ) [inline]
```

5.2.3.32 InitEdge2()

```
void ClipperLib::InitEdge2 (
    TEdge & e,
    PolyType Pt )
```

5.2.3.33 Int128Mul()

```
Int128 ClipperLib::Int128Mul (
    long64 lhs,
    long64 rhs )
```

5.2.3.34 IntersectListSort()

```
bool ClipperLib::IntersectListSort (
    IntersectNode * node1,
    IntersectNode * node2 )
```

5.2.3.35 IntersectPoint()

```
void ClipperLib::IntersectPoint (
    TEdge & Edge1,
    TEdge & Edge2,
    IntPoint & ip )
```

5.2.3.36 IsHorizontal()

```
bool ClipperLib::IsHorizontal (
    TEdge & e ) [inline]
```

5.2.3.37 IsIntermediate()

```
bool ClipperLib::IsIntermediate (
    TEdge * e,
    const cInt Y ) [inline]
```

5.2.3.38 IsMaxima()

```
bool ClipperLib::IsMaxima (
    TEdge * e,
    const cInt Y ) [inline]
```

5.2.3.39 IsMinima()

```
bool ClipperLib::IsMinima (
    TEdge * e ) [inline]
```

5.2.3.40 JoinHorz()

```
bool ClipperLib::JoinHorz (
    OutPt * op1,
    OutPt * op1b,
    OutPt * op2,
    OutPt * op2b,
    const IntPoint Pt,
    bool DiscardLeft )
```

5.2.3.41 Minkowski()

```
void ClipperLib::Minkowski (
    const Path & poly,
    const Path & path,
    Paths & solution,
    bool isSum,
    bool isClosed )
```

5.2.3.42 MinkowskiDiff()

```
void ClipperLib::MinkowskiDiff (
    const Path & poly1,
    const Path & poly2,
    Paths & solution )
```

5.2.3.43 MinkowskiSum() [1/2]

```
void ClipperLib::MinkowskiSum (
    const Path & pattern,
    const Path & path,
    Paths & solution,
    bool pathIsClosed )
```

5.2.3.44 MinkowskiSum() [2/2]

```
void ClipperLib::MinkowskiSum (
    const Path & pattern,
    const Paths & paths,
    Paths & solution,
    bool pathIsClosed )
```

5.2.3.45 OpenPathsFromPolyTree()

```
void ClipperLib::OpenPathsFromPolyTree (
    PolyTree & polytree,
    Paths & paths )
```

5.2.3.46 operator<<() [1/5]

```
Path& ClipperLib::operator<< (
    Path & poly,
    const IntPoint & p ) [inline]
```

5.2.3.47 operator<<() [2/5]

```
Paths& ClipperLib::operator<< (
    Paths & polys,
    const Path & p ) [inline]
```

5.2.3.48 operator<<() [3/5]

```
std::ostream & ClipperLib::operator<< (
    std::ostream & s,
    const IntPoint & p )
```

5.2.3.49 operator<<() [4/5]

```
std::ostream & ClipperLib::operator<< (
    std::ostream & s,
    const Path & p )
```

5.2.3.50 operator<<() [5/5]

```
std::ostream & ClipperLib::operator<< (
    std::ostream & s,
    const Paths & p )
```

5.2.3.51 Orientation()

```
bool ClipperLib::Orientation (
    const Path & poly )
```

5.2.3.52 OutRec1RightOfOutRec2()

```
bool ClipperLib::OutRec1RightOfOutRec2 (
    OutRec * outRec1,
    OutRec * outRec2 )
```

5.2.3.53 ParseFirstLeft()

```
static OutRec* ClipperLib::ParseFirstLeft (
    OutRec * FirstLeft ) [static]
```

5.2.3.54 PointCount()

```
int ClipperLib::PointCount (
    OutPt * Pts )
```

5.2.3.55 PointInPolygon() [1/2]

```
int ClipperLib::PointInPolygon (
    const IntPoint & pt,
    const Path & path )
```

5.2.3.56 PointInPolygon() [2/2]

```
int ClipperLib::PointInPolygon (
    const IntPoint & pt,
    OutPt * op )
```

5.2.3.57 PointIsVertex()

```
bool ClipperLib::PointIsVertex (
    const IntPoint & Pt,
    OutPt * pp )
```

5.2.3.58 PointsAreClose()

```
bool ClipperLib::PointsAreClose (
    IntPoint pt1,
    IntPoint pt2,
    double distSqr )
```

5.2.3.59 Poly2ContainsPoly1()

```
bool ClipperLib::Poly2ContainsPoly1 (
    OutPt * OutPt1,
    OutPt * OutPt2 )
```

5.2.3.60 PolyTreeToPaths()

```
void ClipperLib::PolyTreeToPaths (
    const PolyTree & polytree,
    Paths & paths )
```

5.2.3.61 Pt2IsBetweenPt1AndPt3()

```
bool ClipperLib::Pt2IsBetweenPt1AndPt3 (
    const IntPoint pt1,
    const IntPoint pt2,
    const IntPoint pt3 )
```

5.2.3.62 RangeTest()

```
void ClipperLib::RangeTest (
    const IntPoint & Pt,
    bool & useFullRange )
```

5.2.3.63 RemoveEdge()

```
TEdge* ClipperLib::RemoveEdge (
    TEdge * e )
```

5.2.3.64 ReverseHorizontal()

```
void ClipperLib::ReverseHorizontal (
    TEdge & e ) [inline]
```

5.2.3.65 ReversePath()

```
void ClipperLib::ReversePath (
    Path & p )
```

5.2.3.66 ReversePaths()

```
void ClipperLib::ReversePaths (
    Paths & p )
```

5.2.3.67 ReversePolyPtLinks()

```
void ClipperLib::ReversePolyPtLinks (
    OutPt * pp )
```

5.2.3.68 Round()

```
cInt ClipperLib::Round (
    double val ) [inline]
```

5.2.3.69 SetDx()

```
void ClipperLib::SetDx (
    TEdge & e ) [inline]
```

5.2.3.70 SimplifyPolygon()

```
void ClipperLib::SimplifyPolygon (
    const Path & in_poly,
    Paths & out_polys,
    PolyFillType fillType )
```


5.2.3.71 SimplifyPolygons() [1/2]

```
void ClipperLib::SimplifyPolygons (
    const Paths & in_polys,
    Paths & out_polys,
    PolyFillType fillType )
```

5.2.3.72 SimplifyPolygons() [2/2]

```
void ClipperLib::SimplifyPolygons (
    Paths & polys,
    PolyFillType fillType )
```

5.2.3.73 SlopesEqual() [1/3]

```
bool ClipperLib::SlopesEqual (
    const TEdge & e1,
    const TEdge & e2,
    bool UseFullInt64Range )
```

5.2.3.74 SlopesEqual() [2/3]

```
bool ClipperLib::SlopesEqual (
    const IntPoint pt1,
    const IntPoint pt2,
    const IntPoint pt3,
    bool UseFullInt64Range )
```

5.2.3.75 SlopesEqual() [3/3]

```
bool ClipperLib::SlopesEqual (
    const IntPoint pt1,
    const IntPoint pt2,
    const IntPoint pt3,
    const IntPoint pt4,
    bool UseFullInt64Range )
```

5.2.3.76 SlopesNearCollinear()

```
bool ClipperLib::SlopesNearCollinear (
    const IntPoint & pt1,
    const IntPoint & pt2,
    const IntPoint & pt3,
    double distSqr )
```

5.2.3.77 SwapIntersectNodes()

```
void ClipperLib::SwapIntersectNodes (
    IntersectNode & int1,
    IntersectNode & int2 )
```

5.2.3.78 SwapPoints()

```
void ClipperLib::SwapPoints (
    IntPoint & pt1,
    IntPoint & pt2 )
```

5.2.3.79 SwapPolyIndexes()

```
void ClipperLib::SwapPolyIndexes (
    TEdge & Edge1,
    TEdge & Edge2 ) [inline]
```

5.2.3.80 SwapSides()

```
void ClipperLib::SwapSides (
    TEdge & Edge1,
    TEdge & Edge2 ) [inline]
```

5.2.3.81 TopX()

```
cInt ClipperLib::TopX (
    TEdge & edge,
    const cInt currentY ) [inline]
```

5.2.3.82 TranslatePath()

```
void ClipperLib::TranslatePath (
    const Path & input,
    Path & output,
    const IntPoint delta )
```

5.2.3.83 UpdateOutPtIdxs()

```
void ClipperLib::UpdateOutPtIdxs (
    OutRec & outrec ) [inline]
```

5.2.4 Variable Documentation

5.2.4.1 def_arc_tolerance

```
double const ClipperLib::def_arc_tolerance = 0.25 [static]
```

5.2.4.2 hiRange

```
cInt const ClipperLib::hiRange = 0x3FFFFFFFFFFFFFFFLL [static]
```

5.2.4.3 loRange

```
cInt const ClipperLib::loRange = 0x3FFFFFFF [static]
```

5.2.4.4 pi

```
double const ClipperLib::pi = 3.141592653589793238 [static]
```

5.2.4.5 Skip

```
int const ClipperLib::Skip = -2 [static]
```

5.2.4.6 two_pi

```
double const ClipperLib::two_pi = pi *2 [static]
```

5.2.4.7 Unassigned

```
int const ClipperLib::Unassigned = -1 [static]
```

5.3 DW Namespace Reference

Functions

- void [init](#) (x, y, GLfloat *vertices_buffer={0.0f})
- void [changeBuffer](#) (GLfloat *vertices_buffer, uint dim)

Variables

- GLFWwindow * [window](#)
- GLuint [map_buffer](#)

5.3.1 Function Documentation

5.3.1.1 changeBuffer()

```
void DW::changeBuffer (
    GLfloat * vertices_buffer,
    uint dim )
```

5.3.1.2 init()

```
void DW::init (
    x ,
    y ,
    GLfloat * vertices_buffer = {0.0f} )
```

5.3.2 Variable Documentation

5.3.2.1 map_buffer

```
GLuint DW::map_buffer
```

5.3.2.2 window

```
GLFWwindow* DW::window
```

5.4 timeutils Namespace Reference

Functions

- `int64_t timespecDiff` (struct timespec *timeA_p, struct timespec *timeB_p)
- `double getTimeS` ()

5.4.1 Function Documentation

5.4.1.1 getTimeS()

```
double timeutils::getTimeS ( )
```

5.4.1.2 timespecDiff()

```
int64_t timeutils::timespecDiff (
    struct timespec * timeA_p,
    struct timespec * timeB_p )
```


Chapter 6

Class Documentation

6.1 Angle Class Reference

This class allows to save and handle angles. It supports DEG and RAD, operations such as addition and subtraction with operators overloading, conversion from RAD to DEG and viceversa and normalization of the angle.

```
#include <maths.hh>
```

Public Types

- enum `ANGLE_TYPE` { `DEG`, `RAD`, `INVALID` }

Public Member Functions

- `Angle ()`
A void constructor to create an angle.
- `Angle (double _th, ANGLE_TYPE _type=RAD)`
This constructor takes the angle value and the type of angle and stores them. It also normalize the angle in case is above 2pi (360°) or below 0.
- `double get () const`
Returns the dimension of the angle.
- `ANGLE_TYPE getType () const`
Returns the type of the angle.
- `string getTypeName () const`
- `template<class T >`
`void set (const T _th)`
Set the value of the angle.
- `void setType (ANGLE_TYPE _type)`
Set the type of the angle.
- `double degToRad ()`
Convert and store the angle from DEG to RAD.
- `double radToDeg ()`
Converts and stores the angle from RAD to DEG.
- `double toRad () const`
Converts but does not store the value of the angle from DEG to RAD.

- double `toDeg ()` const
Converts but does not store the value of the angle from RAD to DEG.
- void `normalize ()`
Normalize the angle, that is to set it in $[0, 2\pi)$ or $[0, 360)$. Moreover it check if the value is infinite or NaN. In this case the type is set to `INVALID`.
- `Angle add (const Angle phi)`
Sums and angle to this one. In the process a new angle is created so `normalize ()` is also called.
- `Angle sub (const Angle phi)`
Subtracts and angle to this one. In the process a new angle is created so `normalize ()` is also called.
- `template<class T1 >`
`Angle mul (const T1 A)`
Multiply and angle by a costant. In the process a new angle is created so `normalize ()` is also called.
- `template<class T1 >`
`Angle div (const T1 A)`
Divide and angle by a costant. In the process a new angle is created so `normalize ()` is also called.
- `Angle copy (const Angle phi)`
Copies an angle to this one. In the process a new angle is created so `normalize ()` is also called.
- `Angle operator+ (const Angle phi)`
- `Angle operator- (const Angle phi)`
- `template<class T1 >`
`Angle operator * (const T1 A)`
- `template<class T1 >`
`Angle operator/ (const T1 A)`
- `Angle operator= (const Angle phi)`
- `Angle operator= (const double phi)`
- `Angle & operator+= (const Angle phi)`
- `Angle & operator-= (const Angle phi)`
- `template<class T >`
`Angle & operator *= (const T A)`
- `template<class T >`
`Angle & operator/= (const T A)`
- bool `equal (const Angle &phi)`
- bool `less (const Angle &phi)`
- bool `greater (const Angle &phi)`
- bool `operator== (const Angle &phi)`
- bool `operator!= (const Angle &phi)`
- bool `operator< (const Angle &phi)`
- bool `operator> (const Angle &phi)`
- bool `operator<= (const Angle &phi)`
- bool `operator>= (const Angle &phi)`
- double `cos ()` const
Compute the cosine of the angle. \retunrs A double that is the cosine of the angle.
- double `sin ()` const
Compute the sine of the angle. \retunrs A double that is the sine of the angle.
- double `tan ()` const
Compute the tangent of the angle. \retunrs A double that is the tangent of the angle.
- `operator int ()` const
Cast to int.
- `operator double ()` const
Cast to double.
- `operator float ()` const
Cast to float.
- `operator long ()` const
Cast to long.
- stringstream `to_string (ANGLE_TYPE _type=INVALID)` const

Static Public Member Functions

- static bool `checkValue` (const double th)

Friends

- ostream & `operator<<` (ostream &out, const `Angle` &data)

6.1.1 Detailed Description

This class allows to save and handle angles. It supports DEG and RAD, operations such as addition and subtraction with operators overloading, conversion from RAD to DEG and viceversa and normalization of the angle.

6.1.2 Member Enumeration Documentation

6.1.2.1 ANGLE_TYPE

```
enum Angle::ANGLE_TYPE
```

Enumerator

DEG	
RAD	
INVALID	

6.1.3 Constructor & Destructor Documentation

6.1.3.1 `Angle()` [1/2]

```
Angle::Angle ( ) [inline]
```

A void constructor to create an angle.

6.1.3.2 `Angle()` [2/2]

```
Angle::Angle (
    double _th,
    ANGLE_TYPE _type = RAD ) [inline]
```

This constructor takes the angle value and the type of angle and stores them. It also normalize the angle in case is above 2pi (360°) or below 0.

Parameters

in	<i>_th</i>	The dimension of the angle.
in	<i>_type</i>	The type of the angle.

6.1.4 Member Function Documentation**6.1.4.1 add()**

```
Angle Angle::add (  
    const Angle phi ) [inline]
```

Sums and angle to this one. In the process a new angle is created so `normalize()` is also called.

Parameters

in	<i>phi</i>	The angle to be summed.
----	------------	-------------------------

Returns

The angle summed.

6.1.4.2 checkValue()

```
static bool Angle::checkValue (  
    const double th ) [inline], [static]
```

6.1.4.3 copy()

```
Angle Angle::copy (  
    const Angle phi ) [inline]
```

Copies an angle to this one. In the process a new angle is created so `normalize()` is also called.

Parameters

in	<i>A</i>	The angle to be copied.
----	----------	-------------------------

Returns

The new angle.

6.1.4.4 cos()

```
double Angle::cos ( ) const [inline]
```

Compute the cosine of the angle. \return A double that is the cosine of the angle.

6.1.4.5 degToRad()

```
double Angle::degToRad ( ) [inline]
```

Convert and store the angle from DEG to RAD.

Returns

The value of the angle.

6.1.4.6 div()

```
template<class T1 >
Angle Angle::div (
    const T1 A ) [inline]
```

Divide and angle by a costant. In the process a new angle is created so `normalize()` is also called.

Template Parameters

<i>The</i>	type of the dividend.
------------	-----------------------

Parameters

<i>in</i>	<i>A</i>	The costant to use to divide.
-----------	----------	-------------------------------

Returns

The angle divided.

6.1.4.7 equal()

```
bool Angle::equal (
    const Angle & phi ) [inline]
```

This function takes an angle to compare, and using the `equal` function for `doubles` calculates if it is equal or not to this.

Parameters

in	<i>phi</i>	The angle to compare.
----	------------	-----------------------

Returns

`true` if the two angles are equal, `false` otherwise.

6.1.4.8 get()

```
double Angle::get ( ) const [inline]
```

Returns the dimension of the angle.

6.1.4.9 getType()

```
ANGLE_TYPE Angle::getType ( ) const [inline]
```

Returns the type of the angle.

6.1.4.10 getTypeName()

```
string Angle::getTypeName ( ) const [inline]
```

<Returns a string that tells the type of angle.

6.1.4.11 greater()

```
bool Angle::greater (
    const Angle & phi ) [inline]
```

This function takes the value in radians of an angle and compares it with this.

Parameters

in	<i>phi</i>	The angle to compare.
----	------------	-----------------------

Returns

true if this is more than phi, false otherwise.

6.1.4.12 less()

```
bool Angle::less (
    const Angle & phi ) [inline]
```

This function takes the value in radians of an angle and compares it with this.

Parameters

in	<i>phi</i>	The angle to compare.
----	------------	-----------------------

Returns

true if this is less than phi, false otherwise.

6.1.4.13 mul()

```
template<class T1 >
Angle Angle::mul (
    const T1 A ) [inline]
```

Multiply and angle by a constant. In the process a new angle is created so `normalize()` is also called.

Template Parameters

<i>The</i>	type of the coefficient.
------------	--------------------------

Parameters

in	<i>phi</i>	The constant to use to multiply.
----	------------	----------------------------------

Returns

The angle multiplied.

6.1.4.14 normalize()

```
void Angle::normalize ( ) [inline]
```

Normalize the angle, that is to set it in $[0, 2\pi)$ or $[0, 360)$. Moreover it check if the value is infinite or NaN. In this case the `type` is set to `INVALID`.

6.1.4.15 operator*()

```
template<class T1 >
Angle Angle::operator * (
    const T1 A ) [inline]
```

This function overload the operator `*`. It simply calls the `mul()` function.

Template Parameters

<i>The</i>	type of the coefficient.
------------	--------------------------

Parameters

in	<i>A</i>	The coefficient.
----	----------	------------------

Returns

The angle multiplied.

6.1.4.16 operator*=()

```
template<class T >
Angle& Angle::operator *= (
    const T A ) [inline]
```

This function overload the operator `*=`. It simply calls the `mul()` function and then assign the result to this.

Parameters

in	<i>A</i>	The coefficient.
----	----------	------------------

Returns

`this`.

6.1.4.17 operator double()

```
Angle::operator double ( ) const [inline]
```

Cast to double.

Returns

The value in RAD of the angle casted to double

6.1.4.18 operator float()

```
Angle::operator float ( ) const [inline]
```

Cast to float.

Returns

The value in RAD of the angle casted to float

6.1.4.19 operator int()

```
Angle::operator int ( ) const [inline]
```

Cast to int.

Returns

The value in RAD of the angle casted to int

6.1.4.20 operator long()

```
Angle::operator long ( ) const [inline]
```

Cast to long.

Returns

The value in RAD of the angle casted to long

6.1.4.21 operator!=(())

```
bool Angle::operator!= (
    const Angle & phi ) [inline]
```

This function overload the operator ==. It simply calls the `equal()` function.

Parameters

in	<i>phi</i>	The second angle.
----	------------	-------------------

Returns

`true` if the two angle are equal, `false` otherwise.

This function overload the operator `==`. It simply calls the `equal()` function and negates it.

Parameters

in	<i>phi</i>	The second angle.
----	------------	-------------------

Returns

`false` if the two angle are equal, `true` otherwise.

6.1.4.22 operator+()

```
Angle Angle::operator+ (
    const Angle phi ) [inline]
```

This function overload the operator `+`. It simply calls the `add()` function.

Parameters

in	<i>phi</i>	The angle to be summed.
----	------------	-------------------------

Returns

The angle summed.

6.1.4.23 operator+=()

```
Angle& Angle::operator+= (
    const Angle phi ) [inline]
```

This function overload the operator `+=`. It simply calls the `add()` function and then assign the result to this.

Parameters

in	<i>phi</i>	The angle to be summed.
----	------------	-------------------------

Returns

`this.`

6.1.4.24 operator-()

```
Angle Angle::operator- (
    const Angle phi ) [inline]
```

This function overload the operator -. It simply calls the `sub()` function.

Parameters

in	<i>phi</i>	The angle to be subtracted.
----	------------	-----------------------------

Returns

The angle subtracted.

6.1.4.25 operator-=()

```
Angle& Angle::operator-= (
    const Angle phi ) [inline]
```

This function overload the operator -=. It simply calls the `sub()` function and then assign the result to this.

Parameters

in	<i>phi</i>	The angle to be subtracted.
----	------------	-----------------------------

Returns

`this.`

6.1.4.26 operator/()

```
template<class T1 >
Angle Angle::operator/ (
    const T1 A ) [inline]
```

This function overload the operator /. It simply calls the `div()` function.

Template Parameters

<i>The</i>	type of the dividend.
------------	-----------------------

Parameters

in	<i>A</i>	The dividend.
----	----------	---------------

Returns

The angle divided.

6.1.4.27 operator/=()

```
template<class T >
Angle& Angle::operator/= (
    const T A ) [inline]
```

This function overload the operator /=. It simply calls the [div \(\)](#) function and then assign the result to this.

Parameters

in	<i>A</i>	The dividend.
----	----------	---------------

Returns

this.

6.1.4.28 operator<()

```
bool Angle::operator< (
    const Angle & phi ) [inline]
```

6.1.4.29 operator<=()

```
bool Angle::operator<= (
    const Angle & phi ) [inline]
```

6.1.4.30 operator=() [1/2]

```
Angle Angle::operator= (
    const Angle phi ) [inline]
```

This function overload the operator =. It simply calls the [copy \(\)](#) function.

Parameters

in	<i>phi</i>	The angle to be copied.
----	------------	-------------------------

Returns

The new angle.

6.1.4.31 operator=() [2/2]

```
Angle Angle::operator= (  
    const double phi ) [inline]
```

6.1.4.32 operator==()

```
bool Angle::operator== (  
    const Angle & phi ) [inline]
```

This function overload the operator ==. It simply calls the `equal()` function.

Parameters

in	<i>phi</i>	The second angle.
----	------------	-------------------

Returns

true if the two angle are equal, false otherwise.

6.1.4.33 operator>()

```
bool Angle::operator> (  
    const Angle & phi ) [inline]
```

6.1.4.34 operator>=()

```
bool Angle::operator>= (  
    const Angle & phi ) [inline]
```

6.1.4.35 radToDeg()

```
double Angle::radToDeg ( ) [inline]
```

Converts and stores the angle from RAD to DEG.

Returns

The value of the angle.

6.1.4.36 set()

```
template<class T >
void Angle::set (
    const T _th ) [inline]
```

Set the value of the angle.

Template Parameters

<i>T</i>	The programming type for the value to be stored. It's then cast to <code>double</code> .
----------	--

Parameters

in	↔	The dimension of the angle to be stored.
	↔ <i>th</i>	

6.1.4.37 setType()

```
void Angle::setType (
    ANGLE_TYPE _type ) [inline]
```

Set the type of the angle.

Parameters

in	↔	The type of the angle to be stored.
	↔ <i>th</i>	

6.1.4.38 sin()

```
double Angle::sin ( ) const [inline]
```

Compute the sine of the angle. \returns A `double` that is the sine of the angle.

6.1.4.39 sub()

```
Angle Angle::sub (
    const Angle phi ) [inline]
```

Subtracts and angle to this one. In the process a new angle is created so `normalize()` is also called.

Parameters

in	<i>phi</i>	The angle to be subtracted.
----	------------	-----------------------------

Returns

The angle subtracted.

6.1.4.40 tan()

```
double Angle::tan ( ) const [inline]
```

Compute the tangent of the angle. \returns A `double` that is the tangent of the angle.

6.1.4.41 to_string()

```
stringstream Angle::to_string (
    ANGLE_TYPE _type = INVALID ) const [inline]
```

This function create a strinstream object containing the most essential info, that is the dimension and the type of angle.

Parameters

in	<i>The</i>	type of values to be printed. Default is set to INVALID and it'll print the data of the <code>Angle</code> as it was saved.
----	------------	---

Returns

A string stream.

6.1.4.42 toDeg()

```
double Angle::toDeg ( ) const [inline]
```

Converts but does not store the value of the angle from RAD to DEG.

Returns

The value of the angle

6.1.4.43 toRad()

```
double Angle::toRad ( ) const [inline]
```

Converts but does not store the value of the angle from DEG to RAD.

Returns

The value of the angle

6.1.5 Friends And Related Function Documentation**6.1.5.1 operator<<**

```
ostream& operator<< (
    ostream & out,
    const Angle & data ) [friend]
```

This function overload the << operator so to print with `std::cout` the most essential info, that is the dimension and the type of angle.

Parameters

in	out	The out stream.
in	data	The angle to print.

Returns

An output stream to be printed.

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

- `src/include/maths.hh`

6.2 CalSettings Class Reference

```
#include <calibration.hh>
```

Public Types

- enum `Pattern` { `NOT_EXISTING` =0, `CHESSBOARD` =1 }
- enum `InputType` { `INVALID` =0, `IMAGE_LIST` =3 }

Public Member Functions

- `CalSettings ()`
Constructor that sets `goodInput` to false.
- void `write` (FileStorage &fs) const
Write serialization.
- void `read` (const FileNode &node)
Read serialization.
- void `validate` ()
This function validate the content of the file.
- Mat `nextImage` ()
Get next image from list.

Static Public Member Functions

- static bool `readStringList` (const string &filename, vector< string > &l)
Read from file a list of images.
- static bool `isListOfImages` (const string &filename)
Check if the file from which is trying to retrieve a list is a valid format (xml or yaml).

Public Attributes

- Size `boardSize`
The size of the board -> Number of items by width and height.
- Pattern calibrationPattern = `CHESSBOARD`
One of the Chessboard, circles, or asymmetric circle pattern.
- float `squareSize`
The size of a square in your defined unit (point, millimeter, etc).
- int `nrFrames`
The number of frames to use from the input for calibration.
- float `aspectRatio`
The aspect ratio.
- int `delay`
In case of a video input.
- bool `writePoints`
Write detected feature points.
- bool `writeExtrinsics`
Write extrinsic parameters.
- bool `calibZeroTangentDist`
Assume zero tangential distortion.
- bool `calibFixPrincipalPoint`
Fix the principal point at the center.
- bool `flipVertical`
Flip the captured images around the horizontal axis.
- string `outputFileName`
The name of the file where to write.
- bool `showUndistorted`
Show undistorted images after calibration.
- string `input`
The input.
- bool `useFisheye` = false
use fisheye camera model for calibration
- bool `fixK1`
fix K1 distortion coefficient
- bool `fixK2`
fix K2 distortion coefficient
- bool `fixK3`
fix K3 distortion coefficient
- bool `fixK4`
fix K4 distortion coefficient
- bool `fixK5`
fix K5 distortion coefficient
- int `cameraID`
- vector< string > `imageList`
- size_t `atImageList`
- VideoCapture `inputCapture`
- InputType `inputType` = `IMAGE_LIST`
- bool `goodInput`
- int `flag`

6.2.1 Member Enumeration Documentation

6.2.1.1 InputType

```
enum CalSettings::InputType
```

Enumerator

INVALID	
IMAGE_LIST	

6.2.1.2 Pattern

```
enum CalSettings::Pattern
```

Enumerator

NOT_EXISTING	
CHESSBOARD	

6.2.2 Constructor & Destructor Documentation

6.2.2.1 CalSettings()

```
CalSettings::CalSettings ( ) [inline]
```

Constructor that sets `goodInput` to `false`.

6.2.3 Member Function Documentation

6.2.3.1 isListOfImages()

```
bool CalSettings::isListOfImages (
    const string & filename ) [static]
```

Check if the file from which is trying to retrieve a list is a valid format (xml or yaml).

Parameters

in	<i>filename</i>	The name of the file to check for validity.
----	-----------------	---

Returns

`false` is the file is not xml or yaml
`true` otherwise.

6.2.3.2 nextImage()

```
Mat CalSettings::nextImage ( )
```

Get next image from list.

Returns

A matrix containing the next image to consider.

6.2.3.3 read()

```
void CalSettings::read (
    const FileNode & node )
```

Read serialization.

This function read data from a file and stores each node in their corresponding variables.

Parameters

in	<i>node</i>	The node of the file to consider.
----	-------------	-----------------------------------

6.2.3.4 readStringList()

```
bool CalSettings::readStringList (
    const string & filename,
    vector< string > & l ) [static]
```

Read from file a list of images.

Parameters

in	<i>filename</i>	The name of the file from which to read.
out	<i>l</i>	A vector which will contain the names of the file from the list.

Returns

`false` if the file could not be opened or if the file doesn't contain a list
`true` otherwise.

6.2.3.5 validate()

```
void CalSettings::validate ( )
```

This function validate the content of the file.

Even though this function doesn't return anything nor has any parameters for output, it sets a variable of the `CalSettings` class, that is `googInput`, to `false` if some infos were wrong. `true` otherwise. The options it takes in consideration are the following:

- Size must be positive.
- Cells must be greater than 10^{-6} .
- The number of frames considered, that is images, must be greater than 0.
- Check for valid input, that is a valid list of images.
- Else a list of image is being used.
- Check the field pattern: if it doesn't correspond to a known one than it's invalid.

6.2.3.6 write()

```
void CalSettings::write (
    FileStorage & fs ) const
```

Write serialization.

This function write data to a file.

Parameters

in	<i>fs</i>	The filename where to write.
----	-----------	------------------------------

6.2.4 Member Data Documentation

6.2.4.1 aspectRatio

```
float CalSettings::aspectRatio
```

The aspect ratio.

6.2.4.2 atImageList

```
size_t CalSettings::atImageList
```

6.2.4.3 boardSize

```
Size CalSettings::boardSize
```

The size of the board -> Number of items by width and height.

6.2.4.4 calibFixPrincipalPoint

```
bool CalSettings::calibFixPrincipalPoint
```

Fix the principal point at the center.

6.2.4.5 calibrationPattern

```
Pattern CalSettings::calibrationPattern = CHESSBOARD
```

One of the Chessboard, circles, or asymmetric circle pattern.

6.2.4.6 calibZeroTangentDist

```
bool CalSettings::calibZeroTangentDist
```

Assume zero tangential distortion.

6.2.4.7 cameraID

```
int CalSettings::cameraID
```

6.2.4.8 delay

```
int CalSettings::delay
```

In case of a video input.

6.2.4.9 fixK1

```
bool CalSettings::fixK1
```

fix K1 distortion coefficient

6.2.4.10 fixK2

```
bool CalSettings::fixK2
```

fix K2 distortion coefficient

6.2.4.11 fixK3

```
bool CalSettings::fixK3
```

fix K3 distortion coefficient

6.2.4.12 fixK4

```
bool CalSettings::fixK4
```

fix K4 distortion coefficient

6.2.4.13 fixK5

```
bool CalSettings::fixK5
```

fix K5 distortion coefficient

6.2.4.14 flag

```
int CalSettings::flag
```

6.2.4.15 flipVertical

```
bool CalSettings::flipVertical
```

Flip the captured images around the horizontal axis.

6.2.4.16 goodInput

```
bool CalSettings::goodInput
```

6.2.4.17 imageList

```
vector<string> CalSettings::imageList
```

6.2.4.18 input

```
string CalSettings::input
```

The input.

6.2.4.19 inputCapture

```
VideoCapture CalSettings::inputCapture
```

6.2.4.20 inputType

```
InputType CalSettings::inputType = IMAGE_LIST
```

6.2.4.21 nrFrames

```
int CalSettings::nrFrames
```

The number of frames to use from the input for calibration.

6.2.4.22 outputFileName

```
string CalSettings::outputFileName
```

The name of the file where to write.

6.2.4.23 showUndistorted

```
bool CalSettings::showUndistorted
```

Show undistorted images after calibration.

6.2.4.24 squareSize

```
float CalSettings::squareSize
```

The size of a square in your defined unit (point, millimeter,etc).

6.2.4.25 useFisheye

```
bool CalSettings::useFisheye = false
```

use fisheye camera model for calibration

6.2.4.26 writeExtrinsics

```
bool CalSettings::writeExtrinsics
```

Write extrinsic parameters.

6.2.4.27 writePoints

```
bool CalSettings::writePoints
```

Write detected feature points.

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

- [src/include/calibration.hh](#)
- [src/calibration.cc](#)

6.3 CameraCapture Class Reference

```
#include <camera_capture.hh>
```

Inherits VideoCapture.

Classes

- struct [input_options_t](#)
Structure for store the input option for the class [CameraCapture](#).

Public Member Functions

- [CameraCapture](#) ([input_options_t](#) options)
- bool [grab](#) (cv::Mat &img, double ×tamp)
- bool [isOpen](#) ()
- bool [isAlive](#) ()
- [~CameraCapture](#) ()
- bool [startCamera](#) ()
- bool [loadCoefficients](#) (std::string const &filename)

6.3.1 Constructor & Destructor Documentation

6.3.1.1 CameraCapture()

```
CameraCapture::CameraCapture (
    input\_options\_t options )
```

Initializer of the camera capture class

Parameters

<i>options</i>	for the class
----------------	---------------

Returns**6.3.1.2 ~CameraCapture()**

```
CameraCapture::~~CameraCapture ( )
```

release the resource

6.3.2 Member Function Documentation**6.3.2.1 grab()**

```
bool CameraCapture::grab (
    cv::Mat & img,
    double & timestamp )
```

Grab the first frame available and store it in frame variable

Returns

success if a frame is grabbed, false if not

6.3.2.2 isAlive()

```
bool CameraCapture::isAlive ( )
```

Check if the videostream is alive

Returns

true if open, false if not

6.3.2.3 isOpened()

```
bool CameraCapture::isOpened ( )
```

Check if the videostream is opened

Returns

true if open, false if not

6.3.2.4 loadCoefficients()

```
bool CameraCapture::loadCoefficients (
    std::string const & filename )
```

6.3.2.5 startCamera()

```
bool CameraCapture::startCamera ( )
```

get time in ns

Returns

time in ns

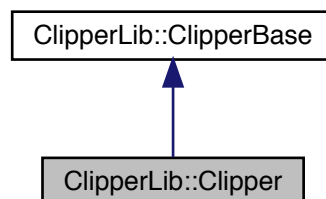
The documentation for this class was generated from the following files:

- [src/include/camera_capture.hh](#)
- [src/camera_capture.cc](#)

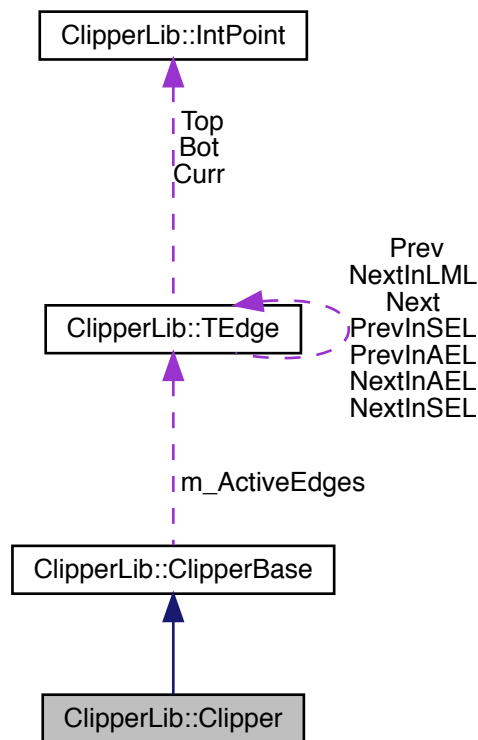
6.4 ClipperLib::Clipper Class Reference

```
#include <clipper.hh>
```

Inheritance diagram for ClipperLib::Clipper:



Collaboration diagram for ClipperLib::Clipper:



Public Member Functions

- **Clipper** (`int` initOptions=0)
- `bool Execute` (`ClipType` clipType, `Paths` &solution, `PolyFillType` fillType=`pftEvenOdd`)
- `bool Execute` (`ClipType` clipType, `Paths` &solution, `PolyFillType` subjFillType, `PolyFillType` clipFillType)
- `bool Execute` (`ClipType` clipType, `PolyTree` &polytree, `PolyFillType` fillType=`pftEvenOdd`)
- `bool Execute` (`ClipType` clipType, `PolyTree` &polytree, `PolyFillType` subjFillType, `PolyFillType` clipFillType)
- `bool ReverseSolution` ()
- `void ReverseSolution` (bool value)
- `bool StrictlySimple` ()
- `void StrictlySimple` (bool value)

Protected Member Functions

- `virtual bool ExecuteInternal` ()

Additional Inherited Members

6.4.1 Constructor & Destructor Documentation

6.4.1.1 Clipper()

```
ClipperLib::Clipper::Clipper (
    int initOptions = 0 )
```

6.4.2 Member Function Documentation

6.4.2.1 Execute() [1/4]

```
bool ClipperLib::Clipper::Execute (
    ClipType clipType,
    Paths & solution,
    PolyFillType fillType = pftEvenOdd )
```

6.4.2.2 Execute() [2/4]

```
bool ClipperLib::Clipper::Execute (
    ClipType clipType,
    Paths & solution,
    PolyFillType subjFillType,
    PolyFillType clipFillType )
```

6.4.2.3 Execute() [3/4]

```
bool ClipperLib::Clipper::Execute (
    ClipType clipType,
    PolyTree & polytree,
    PolyFillType fillType = pftEvenOdd )
```

6.4.2.4 Execute() [4/4]

```
bool ClipperLib::Clipper::Execute (
    ClipType clipType,
    PolyTree & polytree,
    PolyFillType subjFillType,
    PolyFillType clipFillType )
```

6.4.2.5 ExecuteInternal()

```
bool ClipperLib::Clipper::ExecuteInternal ( ) [protected], [virtual]
```

6.4.2.6 ReverseSolution() [1/2]

```
bool ClipperLib::Clipper::ReverseSolution ( ) [inline]
```

6.4.2.7 ReverseSolution() [2/2]

```
void ClipperLib::Clipper::ReverseSolution (
    bool value ) [inline]
```

6.4.2.8 StrictlySimple() [1/2]

```
bool ClipperLib::Clipper::StrictlySimple ( ) [inline]
```

6.4.2.9 StrictlySimple() [2/2]

```
void ClipperLib::Clipper::StrictlySimple (
    bool value ) [inline]
```

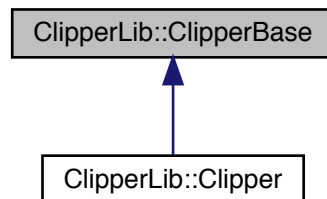
The documentation for this class was generated from the following files:

- [src/include/clipper.hh](#)
- [src/clipper.cc](#)

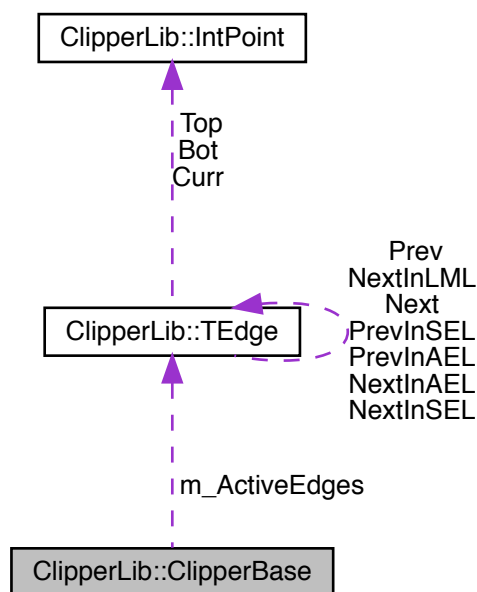
6.5 ClipperLib::ClipperBase Class Reference

```
#include <clipper.h>
```

Inheritance diagram for ClipperLib::ClipperBase:



Collaboration diagram for ClipperLib::ClipperBase:



Public Member Functions

- [ClipperBase](#) ()
- virtual [~ClipperBase](#) ()
- virtual bool [AddPath](#) (const [Path](#) &pg, [PolyType](#) PolyTyp, bool Closed)

- bool [AddPaths](#) (const [Paths](#) &ppg, [PolyType](#) PolyTyp, bool Closed)
- virtual void [Clear](#) ()
- [IntRect](#) [GetBounds](#) ()
- bool [PreserveCollinear](#) ()
- void [PreserveCollinear](#) (bool value)

Protected Types

- typedef std::vector< [LocalMinimum](#) > [MinimaList](#)
- typedef std::priority_queue< [cInt](#) > [ScanbeamList](#)

Protected Member Functions

- void [DisposeLocalMinimaList](#) ()
- [TEdge](#) * [AddBoundsToLML](#) ([TEdge](#) *e, bool IsClosed)
- virtual void [Reset](#) ()
- [TEdge](#) * [ProcessBound](#) ([TEdge](#) *E, bool IsClockwise)
- void [InsertScanbeam](#) (const [cInt](#) Y)
- bool [PopScanbeam](#) ([cInt](#) &Y)
- bool [LocalMinimaPending](#) ()
- bool [PopLocalMinima](#) ([cInt](#) Y, const [LocalMinimum](#) *&locMin)
- [OutRec](#) * [CreateOutRec](#) ()
- void [DisposeAllOutRecs](#) ()
- void [DisposeOutRec](#) ([PolyOutList::size_type](#) index)
- void [SwapPositionsInAEL](#) ([TEdge](#) *edge1, [TEdge](#) *edge2)
- void [DeleteFromAEL](#) ([TEdge](#) *e)
- void [UpdateEdgeIntoAEL](#) ([TEdge](#) *&e)

Protected Attributes

- [MinimaList::iterator](#) [m_CurrentLM](#)
- [MinimaList](#) [m_MinimaList](#)
- bool [m_UseFullRange](#)
- [EdgeList](#) [m_edges](#)
- bool [m_PreserveCollinear](#)
- bool [m_HasOpenPaths](#)
- [PolyOutList](#) [m_PolyOuts](#)
- [TEdge](#) * [m_ActiveEdges](#)
- [ScanbeamList](#) [m_Scanbeam](#)

6.5.1 Member Typedef Documentation

6.5.1.1 MinimaList

```
typedef std::vector<LocalMinimum> ClipperLib::ClipperBase::MinimaList [protected]
```

6.5.1.2 ScanbeamList

```
typedef std::priority_queue<cInt> ClipperLib::ClipperBase::ScanbeamList [protected]
```

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ClipperBase()

```
ClipperLib::ClipperBase::ClipperBase ( )
```

6.5.2.2 ~ClipperBase()

```
ClipperLib::ClipperBase::~~ClipperBase ( ) [virtual]
```

6.5.3 Member Function Documentation

6.5.3.1 AddBoundsToLML()

```
TEdge* ClipperLib::ClipperBase::AddBoundsToLML (
    TEdge * e,
    bool IsClosed ) [protected]
```

6.5.3.2 AddPath()

```
bool ClipperLib::ClipperBase::AddPath (
    const Path & pg,
    PolyType PolyTyp,
    bool Closed ) [virtual]
```

6.5.3.3 AddPaths()

```
bool ClipperLib::ClipperBase::AddPaths (
    const Paths & ppg,
    PolyType PolyTyp,
    bool Closed )
```


6.5.3.4 Clear()

```
void ClipperLib::ClipperBase::Clear ( ) [virtual]
```

6.5.3.5 CreateOutRec()

```
OutRec * ClipperLib::ClipperBase::CreateOutRec ( ) [protected]
```

6.5.3.6 DeleteFromAEL()

```
void ClipperLib::ClipperBase::DeleteFromAEL (
    TEdge * e ) [protected]
```

6.5.3.7 DisposeAllOutRecs()

```
void ClipperLib::ClipperBase::DisposeAllOutRecs ( ) [protected]
```

6.5.3.8 DisposeLocalMinimaList()

```
void ClipperLib::ClipperBase::DisposeLocalMinimaList ( ) [protected]
```

6.5.3.9 DisposeOutRec()

```
void ClipperLib::ClipperBase::DisposeOutRec (
    PolyOutList::size_type index ) [protected]
```

6.5.3.10 GetBounds()

```
IntRect ClipperLib::ClipperBase::GetBounds ( )
```

6.5.3.11 InsertScanbeam()

```
void ClipperLib::ClipperBase::InsertScanbeam (
    const cInt Y ) [protected]
```

6.5.3.12 LocalMinimaPending()

```
bool ClipperLib::ClipperBase::LocalMinimaPending ( ) [protected]
```

6.5.3.13 PopLocalMinima()

```
bool ClipperLib::ClipperBase::PopLocalMinima (
    cInt Y,
    const LocalMinimum *& locMin ) [protected]
```

6.5.3.14 PopScanbeam()

```
bool ClipperLib::ClipperBase::PopScanbeam (
    cInt & Y ) [protected]
```

6.5.3.15 PreserveCollinear() [1/2]

```
bool ClipperLib::ClipperBase::PreserveCollinear ( ) [inline]
```

6.5.3.16 PreserveCollinear() [2/2]

```
void ClipperLib::ClipperBase::PreserveCollinear (
    bool value ) [inline]
```

6.5.3.17 ProcessBound()

```
TEdge * ClipperLib::ClipperBase::ProcessBound (
    TEdge * E,
    bool IsClockwise ) [protected]
```

6.5.3.18 Reset()

```
void ClipperLib::ClipperBase::Reset ( ) [protected], [virtual]
```

6.5.3.19 SwapPositionsInAEL()

```
void ClipperLib::ClipperBase::SwapPositionsInAEL (
    TEdge * edge1,
    TEdge * edge2 ) [protected]
```

6.5.3.20 UpdateEdgeIntoAEL()

```
void ClipperLib::ClipperBase::UpdateEdgeIntoAEL (
    TEdge *& e ) [protected]
```

6.5.4 Member Data Documentation

6.5.4.1 m_ActiveEdges

```
TEdge* ClipperLib::ClipperBase::m_ActiveEdges [protected]
```

6.5.4.2 m_CurrentLM

```
MinimalList::iterator ClipperLib::ClipperBase::m_CurrentLM [protected]
```

6.5.4.3 m_edges

```
EdgeList ClipperLib::ClipperBase::m_edges [protected]
```

6.5.4.4 m_HasOpenPaths

```
bool ClipperLib::ClipperBase::m_HasOpenPaths [protected]
```

6.5.4.5 m_MinimalList

`MinimalList` `ClipperLib::ClipperBase::m_MinimalList` [protected]

6.5.4.6 m_PolyOuts

`PolyOutList` `ClipperLib::ClipperBase::m_PolyOuts` [protected]

6.5.4.7 m_PreserveCollinear

`bool` `ClipperLib::ClipperBase::m_PreserveCollinear` [protected]

6.5.4.8 m_Scanbeam

`ScanbeamList` `ClipperLib::ClipperBase::m_Scanbeam` [protected]

6.5.4.9 m_UseFullRange

`bool` `ClipperLib::ClipperBase::m_UseFullRange` [protected]

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

- `src/include/clipper.hh`
- `src/clipper.cc`

6.6 ClipperLib::clipperException Class Reference

```
#include <clipper.hh>
```

Inherits exception.

Public Member Functions

- `clipperException` (`const char *description`)
- virtual `~clipperException` () `throw ()`
- virtual `const char * what` () `const throw ()`

6.6.1 Constructor & Destructor Documentation

6.6.1.1 clipperException()

```
ClipperLib::clipperException::clipperException (
    const char * description ) [inline]
```

6.6.1.2 ~clipperException()

```
virtual ClipperLib::clipperException::~clipperException ( ) throw ( ) [inline], [virtual]
```

6.6.2 Member Function Documentation

6.6.2.1 what()

```
virtual const char* ClipperLib::clipperException::what ( ) const throw ( ) [inline], [virtual]
```

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

- [src/include/clipper.hh](#)

6.7 ClipperLib::ClipperOffset Class Reference

```
#include <clipper.hh>
```

Public Member Functions

- [ClipperOffset](#) (double miterLimit=2.0, double roundPrecision=0.25)
- [~ClipperOffset](#) ()
- void [AddPath](#) (const [Path](#) &path, [JoinType](#) joinType, [EndType](#) endType)
- void [AddPaths](#) (const [Paths](#) &paths, [JoinType](#) joinType, [EndType](#) endType)
- void [Execute](#) ([Paths](#) &solution, double delta)
- void [Execute](#) ([PolyTree](#) &solution, double delta)
- void [Clear](#) ()

Public Attributes

- double [MiterLimit](#)
- double [ArcTolerance](#)

6.7.1 Constructor & Destructor Documentation

6.7.1.1 ClipperOffset()

```
ClipperLib::ClipperOffset::ClipperOffset (
    double miterLimit = 2.0,
    double roundPrecision = 0.25 )
```

6.7.1.2 ~ClipperOffset()

```
ClipperLib::ClipperOffset::~~ClipperOffset ( )
```

6.7.2 Member Function Documentation

6.7.2.1 AddPath()

```
void ClipperLib::ClipperOffset::AddPath (
    const Path & path,
    JoinType joinType,
    EndType endType )
```

6.7.2.2 AddPaths()

```
void ClipperLib::ClipperOffset::AddPaths (
    const Paths & paths,
    JoinType joinType,
    EndType endType )
```

6.7.2.3 Clear()

```
void ClipperLib::ClipperOffset::Clear ( )
```

6.7.2.4 Execute() [1/2]

```
void ClipperLib::ClipperOffset::Execute (
    Paths & solution,
    double delta )
```

6.7.2.5 Execute() [2/2]

```
void ClipperLib::ClipperOffset::Execute (
    PolyTree & solution,
    double delta )
```

6.7.3 Member Data Documentation

6.7.3.1 ArcTolerance

```
double ClipperLib::ClipperOffset::ArcTolerance
```

6.7.3.2 MiterLimit

```
double ClipperLib::ClipperOffset::MiterLimit
```

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

- [src/include/clipper.hh](#)
- [src/clipper.cc](#)

6.8 Configuration2< T1 > Class Template Reference

This class stores a configuration, that is a point and an angle.

```
#include <maths.hh>
```

Public Member Functions

- [Configuration2](#) ()
Default constructor that use as point (0,0) and as angle 0 RAD.
- [Configuration2](#) (const T1 _x, const T1 _y, const [Angle](#) _th)
Default constructor that takes the coordinates, the angle, and stores them.
- [Configuration2](#) (const [Point2](#)< T1 > P, const [Angle](#) _th)
Default constructor that takes the point, the angle, and stores them.
- [Point2](#)< T1 > [point](#) () const
- T1 [x](#) () const
- T1 [y](#) () const
- [Angle](#) [angle](#) () const
- int [x](#) (const T1 _x)
This function stores a new value for the abscissa.
- int [y](#) (const T1 _y)
This function stores a new value for the ordinate.
- void [angle](#) (const [Angle](#) _th)
This function stores a new value for the angle.
- template<class T2 >
int [offset](#) (const T2 _offset, const [Angle](#) phi, const [Angle](#) _th)
This function compute the offset of the point given a vector, that is the lenght of the vector and its angle. The angle must be an [Angle](#) variable. It takes also another [Angle](#) to change the [Angle](#) in the configuration.
- int [offset](#) ([Configuration2](#)< T1 > p)
This function compute the offset of the point given another [Configuration2](#).
- int [offset](#) ([Point2](#)< T1 > p, const [Angle](#) _th=[Angle](#)())
This function compute the offset of the point given a [Point2](#) containing the offsets for the abscissa and the ordiante and an [Angle](#) to change the [Angle](#) in the configuration.
- int [offset_x](#) (const T1 _offset)
Function to add an offset to the abscissa.
- int [offset_y](#) (const [Angle](#) _offset)
Function to add an offset to the ordinate.
- void [offset_angle](#) (const [Angle](#) _th)
Function to add an offset to the angle.
- template<class T2 >
[Tuple](#)< double > [distance](#) ([Configuration2](#)< T2 > B, [DISTANCE_TYPE](#) dist_type=[EUCLIDEAN](#))
Wrapper to compute different distances. \tparam T2 The type of the elements in the second [Configuration2](#).
- template<class T2 >
[Tuple](#)< double > [EuDistance](#) ([Configuration2](#)< T2 > B)
Function that compute the Euclidean Distance between two configurations. \tparam T2 The type of the elements in the second [Configuration2](#).
- template<class T2 >
[Tuple](#)< double > [MaDistance](#) ([Configuration2](#)< T2 > B)
Function that compute the Manhattan Distance between two configurations. \tparam T2 The type of the elements in the second [Configuration2](#).
- stringstream [to_string](#) () const
Function to create a stringstream containing the detail of the configuration.
- template<class T2 >
[operator](#) [Point2](#)< T2 > () const
Cast of Configuration to [Point2](#).
- [Configuration2](#)< T1 > [copy](#) (const [Configuration2](#)< T1 > &A)
Copy a configuration into another one.
- [Configuration2](#)< T1 > [operator=](#) (const [Configuration2](#)< T1 > &A)
Overload of the = operatore. Just calls [copy](#).

- bool `equal` (const `Configuration2< T1 >` &A)
Equalize two configurations.
- bool `operator==` (const `Configuration2< T1 >` &A)
Overload of the == operator. Just calls `equal`.
- template<class T2 >
`operator Configuration2< T2 > ()` const

Friends

- ostream & `operator<<` (ostream &out, const `Configuration2< T1 >` &data)
Overload of operator << to output the content of a `Configuration2`.

6.8.1 Detailed Description

```
template<class T1>
class Configuration2< T1 >
```

This class stores a configuration, that is a point and an angle.

Template Parameters

<i>T1</i>	The type of the coordinates.
-----------	------------------------------

6.8.2 Constructor & Destructor Documentation

6.8.2.1 Configuration2() [1/3]

```
template<class T1>
Configuration2< T1 >::Configuration2 ( ) [inline]
```

Default constructor that use as point (0,0) and as angle 0 RAD.

6.8.2.2 Configuration2() [2/3]

```
template<class T1>
Configuration2< T1 >::Configuration2 (
    const T1 _x,
    const T1 _y,
    const Angle _th ) [inline]
```

Default constructor that takes the coordinates, the angle, and stores them.

Parameters

in	\leftrightarrow $_ \leftrightarrow$ x	The abscissa coordinate.
in	\leftrightarrow $_ \leftrightarrow$ y	The ordinate coordinate.
in	\leftrightarrow $_ \leftrightarrow$ th	The angle.

6.8.2.3 Configuration2() [3/3]

```
template<class T1>
Configuration2< T1 >::Configuration2 (
    const Point2< T1 > P,
    const Angle _th ) [inline]
```

Default constructor that takes the point, the angle, and stores them.

Parameters

in	P	The coordinates.
in	\leftrightarrow $_ \leftrightarrow$ th	The angle.

6.8.3 Member Function Documentation**6.8.3.1 angle()** [1/2]

```
template<class T1>
Angle Configuration2< T1 >::angle ( ) const [inline]
```

Returns

The angle.

6.8.3.2 angle() [2/2]

```
template<class T1>
void Configuration2< T1 >::angle (
    const Angle _th ) [inline]
```

This function stores a new value for the angle.

Parameters

in	\leftrightarrow	The value to be stored.
	\overleftarrow{th}	

Returns

1 if everything went ok, 0 otherwise.

6.8.3.3 copy()

```
template<class T1>
Configuration2<T1> Configuration2< T1 >::copy (
    const Configuration2< T1 > & A ) [inline]
```

Copy a configuration into another one.

Parameters

in	<i>A</i>	Configuration to be copied.
----	----------	-----------------------------

Returns

this.

6.8.3.4 distance()

```
template<class T1>
template<class T2 >
Tuple<double> Configuration2< T1 >::distance (
    Configuration2< T2 > B,
    DISTANCE_TYPE dist_type = EUCLIDEAN ) [inline]
```

Wrapper to compute different distances. \tparam T2 The type of the elements in the second [Configuration2](#).

Parameters

in	<i>B</i>	The second Configuration2 to use for computing the distance.
in	<i>dist</i>	The type of distance to be computed.

Returns

The distance between the two configurations.

6.8.3.5 equal()

```
template<class T1>
bool Configuration2< T1 >::equal (
    const Configuration2< T1 > & A ) [inline]
```

Equalize two configurations.

Parameters

in	<i>A</i>	Configuration to be equalized.
----	----------	--------------------------------

Returns

true if the two configurations are equal.

6.8.3.6 EuDistance()

```
template<class T1>
template<class T2 >
Tuple<double> Configuration2< T1 >::EuDistance (
    Configuration2< T2 > B ) [inline]
```

Function that compute the Euclidean Distance between two configurations. \tparan T2 The type of the elements in the second [Configuration2](#).

Parameters

in	<i>B</i>	the second Configuration2 to use for computing the distance.
----	----------	--

Returns

The Euclidean distance between the two configurations.

6.8.3.7 MaDistance()

```
template<class T1>
template<class T2 >
Tuple<double> Configuration2< T1 >::MaDistance (
    Configuration2< T2 > B ) [inline]
```

Function that compute the Manhattan Distance between two configurations. \tparan T2 The type of the elements in the second [Configuration2](#).

Parameters

in	B	the second Configuration2 to use for computing the distance.
----	-----	--

Returns

The Manhattan distance between the two configurations.

6.8.3.8 `offset()` [1/3]

```
template<class T1>
template<class T2 >
int Configuration2< T1 >::offset (
    const T2 _offset,
    const Angle phi,
    const Angle _th ) [inline]
```

This function compute the offset of the point given a vector, that is the lenght of the vector and its angle. The angle must be an [Angle](#) variable. It takes also another [Angle](#) to change the [Angle](#) in the configuration.

Template Parameters

--	--

6.8.3.9 `offset()` [2/3]

```
template<class T1>
int Configuration2< T1 >::offset (
    Configuration2< T1 > p ) [inline]
```

This function compute the offset of the point given another [Configuration2](#).

Parameters

in	p	The configuration containing the offsets.
----	-----	---

Returns

1 if everything went fine, 0 otherwise.

6.8.3.10 `offset()` [3/3]

```
template<class T1>
int Configuration2< T1 >::offset (
```

```
Point2< T1 > p,
const Angle _th = Angle() ) [inline]
```

This function compute the offset of the point given a `Point2` containing the offsets for the abscissa and the ordiante and an `Angle` to change the `Angle` in the configuration.

Parameters

in	p	The point containing the offsets.
in	\leftarrow \leftarrow th	The offset for the <code>Angle</code> in the configuration. It's set to 0 as default so to easily change just the coordinates.

Returns

1 if everything went fine, 0 otherwise.

6.8.3.11 offset_angle()

```
template<class T1>
void Configuration2< T1 >::offset_angle (
    const Angle _th ) [inline]
```

Function to add an offset to the angle.

Parameters

in	<code>_offset</code>	The offset.
----	----------------------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.8.3.12 offset_x()

```
template<class T1>
int Configuration2< T1 >::offset_x (
    const T1 _offset ) [inline]
```

Function to add an offset to the abscissa.

Parameters

in	<code>_offset</code>	The offset.
----	----------------------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.8.3.13 offset_y()

```
template<class T1>
int Configuration2< T1 >::offset_y (
    const Angle _offset ) [inline]
```

Function to add an offset to the ordinate.

Parameters

in	_offset	The offset.
----	---------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.8.3.14 operator Configuration2< T2 >()

```
template<class T1>
template<class T2 >
Configuration2< T1 >::operator Configuration2< T2 > ( ) const [inline]
```

6.8.3.15 operator Point2< T2 >()

```
template<class T1>
template<class T2 >
Configuration2< T1 >::operator Point2< T2 > ( ) const [inline]
```

Cast of Configuration to [Point2](#).

Template Parameters

T2	Type of Point2 to be casted to.
----	---

Returns

A [Point2](#) of type T2.

6.8.3.16 operator=()

```
template<class T1>
Configuration2<T1> Configuration2< T1 >::operator= (
    const Configuration2< T1 > & A ) [inline]
```

Overload of the = operator. Just calls `copy`.

Parameters

in	A	Configuration to be copied.
----	---	-----------------------------

Returns

this.

6.8.3.17 operator==()

```
template<class T1>
bool Configuration2< T1 >::operator== (
    const Configuration2< T1 > & A ) [inline]
```

Overload of the == operator. Just calls `equal`.

Parameters

in	A	Configuration to be equalized.
----	---	--------------------------------

Returns

true if the two configurations are equal.

6.8.3.18 point()

```
template<class T1>
Point2<T1> Configuration2< T1 >::point ( ) const [inline]
```

Returns

A `Point2` variable containing the coordinates.

6.8.3.19 to_string()

```
template<class T1>
stringstream Configuration2< T1 >::to_string ( ) const [inline]
```

Function to create a stringstream containing the detail of the configuration.

Returns

A stringstream.

6.8.3.20 x() [1/2]

```
template<class T1>
T1 Configuration2< T1 >::x ( ) const [inline]
```

Returns

The abscissa coordinate.

6.8.3.21 x() [2/2]

```
template<class T1>
int Configuration2< T1 >::x (
    const T1 _x ) [inline]
```

This function stores a new value for the abscissa.

Parameters

in	↔	The value to be stored.
	↔	
	x	

Returns

1 if everything went ok, 0 otherwise.

6.8.3.22 y() [1/2]

```
template<class T1>
T1 Configuration2< T1 >::y ( ) const [inline]
```

Returns

The ordinate coordinate.

6.8.3.23 `y()` [2/2]

```
template<class T1>
int Configuration2< T1 >::y (
    const T1 _y ) [inline]
```

This function stores a new value for the ordinate.

Parameters

in	↔	The value to be stored.
	↔ _y	

Returns

1 if everything went ok, 0 otherwise.

6.8.4 Friends And Related Function Documentation**6.8.4.1** `operator<<`

```
template<class T1>
ostream& operator<< (
    ostream & out,
    const Configuration2< T1 > & data ) [friend]
```

Overload of operator << to output the content of a [Configuration2](#).

Parameters

in	<i>out</i>	The output stream.
in	<i>data</i>	The Configuration2 to print.

Returns

An output stream to be printed.

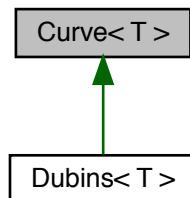
The documentation for this class was generated from the following file:

- [src/include/maths.hh](#)

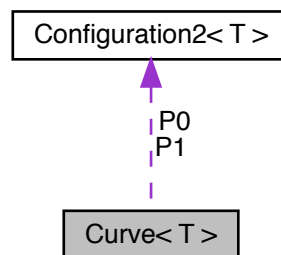
6.9 Curve< T > Class Template Reference

```
#include <dubins.hh>
```

Inheritance diagram for Curve< T >:



Collaboration diagram for Curve< T >:



Public Member Functions

- [Curve](#) ()
Default plain constructor which creates two plain [Configuration2](#)s.
- [Curve](#) (const [Configuration2](#)< T > _P0, const [Configuration2](#)< T > _P1)
- [Curve](#) (const [Point2](#)< T > _P0, const [Point2](#)< T > _P1, const [Angle](#) _th0, const [Angle](#) _th1)
- [Curve](#) (const T x0, const T y0, const [Angle](#) _th0, const T x1, const T y1, const [Angle](#) _th1)
- [Configuration2](#)< T > [begin](#) () const
Returns the starting [Configuration2](#) of the [Curve](#).
- [Configuration2](#)< T > [end](#) () const
Returns the ending [Configuration2](#) of the [Curve](#).
- void [begin](#) ([Configuration2](#)< T > _P0)
- void [end](#) ([Configuration2](#)< T > _P1)
- stringstream [to_string](#) () const

Protected Attributes

- [Configuration2](#)< T > [P0](#)
Start [Configuration2](#).
- [Configuration2](#)< T > [P1](#)
End [Configuration2](#).

Friends

- ostream & [operator](#)<< (ostream &out, const [Curve](#) &data)

6.9.1 Detailed Description

```
template<class T>
class Curve< T >
```

Class that defines a general curve. It just contains a start [Configuration2](#) and an end [Configuration2](#).

Template Parameters

T	The type of the Configuration2s
-------------------	---

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Curve() [1/4]

```
template<class T>
Curve< T >::Curve ( ) [inline]
```

Default plain constructor which creates two plain [Configuration2s](#).

6.9.2.2 Curve() [2/4]

```
template<class T>
Curve< T >::Curve (
    const Configuration2< T > _P0,
    const Configuration2< T > _P1 ) [inline]
```

Constructor that takes two [Configuration2s](#) and stores them.

Parameters

<code>_P0</code>	Start Configuration2 .
<code>_P1</code>	End Configuration2 .

6.9.2.3 Curve() [3/4]

```
template<class T>
Curve< T >::Curve (
    const Point2< T > _P0,
    const Point2< T > _P1,
    const Angle _th0,
    const Angle _th1 ) [inline]
```

Constructor that takes two [Point2](#)s and two [Angles](#) and stores them as [Configuration2](#)s.

Parameters

<code>_P0</code>	Start Point2 .
<code>_P1</code>	End Point2 .
<code>_th0</code>	Starting Angle
<code>_th1</code>	Ending Angle

6.9.2.4 Curve() [4/4]

```
template<class T>
Curve< T >::Curve (
    const T x0,
    const T y0,
    const Angle _th0,
    const T x1,
    const T y1,
    const Angle _th1 ) [inline]
```

Constructor that takes the bare coordinates of two points and their [Angles](#) and stores them as [Configuration2](#)s.

Parameters

<code>x0</code>	Start abscissa coordinate.
<code>y0</code>	Start ordinate coordinate.
<code>_th0</code>	Start Angle .
<code>x1</code>	End abscissa coordinate.
<code>y1</code>	End ordinate coordinate.
<code>_th1</code>	End Angle .

6.9.3 Member Function Documentation

6.9.3.1 begin() [1/2]

```
template<class T>
Configuration2<T> Curve< T >::begin ( ) const [inline]
```

Returns the starting [Configuration2](#) of the [Curve](#).

6.9.3.2 begin() [2/2]

```
template<class T>
void Curve< T >::begin (
    Configuration2< T > _P0 ) [inline]
```

Function that stores the starting [Configuration2](#).

Parameters

<code>_P0</code>	Starting Configuration2 .
------------------	---

6.9.3.3 end() [1/2]

```
template<class T>
Configuration2<T> Curve< T >::end ( ) const [inline]
```

Returns the ending [Configuration2](#) of the [Curve](#).

6.9.3.4 end() [2/2]

```
template<class T>
void Curve< T >::end (
    Configuration2< T > _P1 ) [inline]
```

Function that stores the ending [Configuration2](#).

Parameters

<code>_P0</code>	Ending Configuration2 .
------------------	---

6.9.3.5 to_string()

```
template<class T>
stringstream Curve< T >::to_string ( ) const [inline]
```

This function create a strinstream object containing infos about the [Curve](#).

Returns

A string stream.

6.9.4 Friends And Related Function Documentation

6.9.4.1 operator<<

```
template<class T>
ostream& operator<< (
    ostream & out,
    const Curve< T > & data ) [friend]
```

This function overload the << operator so to print with `std::cout` the values of the [Curve](#).

Parameters

in	<i>out</i>	The out stream.
in	<i>data</i>	The Curve to print.

Returns

An output stream to be printed.

6.9.5 Member Data Documentation

6.9.5.1 P0

```
template<class T>
Configuration2<T> Curve< T >::P0 [protected]
```

Start [Configuration2](#).

6.9.5.2 P1

```
template<class T>
Configuration2<T> Curve< T >::P1 [protected]
```

End `Configuration2`.

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

- `src/include/dubins.hh`

6.10 ClipperLib::DoublePoint Struct Reference

```
#include <clipper.hh>
```

Public Member Functions

- `DoublePoint` (double x=0, double y=0)
- `DoublePoint` (IntPoint ip)

Public Attributes

- double `X`
- double `Y`

6.10.1 Constructor & Destructor Documentation

6.10.1.1 DoublePoint() [1/2]

```
ClipperLib::DoublePoint::DoublePoint (
    double x = 0,
    double y = 0 ) [inline]
```

6.10.1.2 DoublePoint() [2/2]

```
ClipperLib::DoublePoint::DoublePoint (
    IntPoint ip ) [inline]
```

6.10.2 Member Data Documentation

6.10.2.1 X

```
double ClipperLib::DoublePoint::X
```

6.10.2.2 Y

```
double ClipperLib::DoublePoint::Y
```

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

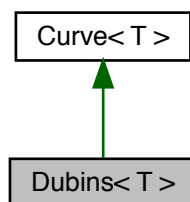
- [src/include/clipper.hh](#)

6.11 Dubins< T > Class Template Reference

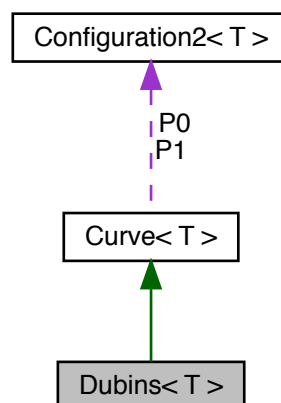
Class to store a [Dubins](#) curve. This class inherits from [Curve](#) and is composed of three [DubinsArc](#).

```
#include <dubins.hh>
```

Inheritance diagram for Dubins< T >:



Collaboration diagram for Dubins< T >:



Public Member Functions

- [Dubins](#) ()
- [Dubins](#) (const [Configuration2](#)< T > _P0, const [Configuration2](#)< T > _P1, const double _K=[KMAX](#))
- [Dubins](#) (const [Point2](#)< T > _P0, const [Point2](#)< T > _P1, const [Angle](#) _th0, const [Angle](#) _th1, const double _K=[KMAX](#))
- [Dubins](#) (const T x0, const T y0, const [Angle](#) _th0, const T x1, const T y1, const [Angle](#) _th1, const double _K=[KMAX](#))
- double [getKMax](#) () const
Returns the maximum curvature of the [Dubins](#).
- double [length](#) () const
Returns the length of the [Dubins](#).
- double [getId](#) ()
Returns the id of the [Dubins](#), that is the set of three maneuvers that creates the curve.
- [DubinsArc](#)< T > [getA1](#) () const
Returns the first [DubinsArc](#).
- [DubinsArc](#)< T > [getA2](#) () const
Returns the second [DubinsArc](#).
- [DubinsArc](#)< T > [getA3](#) () const
Returns the third [DubinsArc](#).
- double * [LSL](#) (double th0, double th1, double _kmax)
- double * [RSR](#) (double th0, double th1, double _kmax)
- double * [LSR](#) (double th0, double th1, double _kmax)
- double * [RSL](#) (double th0, double th1, double _kmax)
- double * [RLR](#) (double th0, double th1, double _kmax)
- double * [LRL](#) (double th0, double th1, double _kmax)
- [Tuple](#)< double > [scaleToStandard](#) ()
Function to compute standardize the parameters. This function computes the initial and final angles as if the reference system is P0(-1,0), P1(0,1). This allows to simplify the calculations to find the best set of maneuvers.
- [Tuple](#)< double > [scaleFromStandard](#) (double lambda, double sc_s1, double sc_s2, double sc_s3)
- int [shortest_path](#) ()
This function computes the shortest path for the [Dubins](#) constructed. First the values are scaled. Then the six sets of maneuvers are computed and their lengths are stored. Once the set that gives the [Dubins](#) with the minimum length is found, the lengths are rescaled and the [DubinsArc](#) are created. In the process length is also computed.
- bool [check](#) (double s1, double k0, double s2, double k1, double s3, double k2, [Angle](#) th0, [Angle](#) th1) const
- [Tuple](#)< [Tuple](#)< [Point2](#)< double > > > [splitIt](#) (int _arch=0, double _L=[PIECE_LENGTH](#))
- stringstream [to_string](#) () const
- void [draw](#) (double dimX, double dimY, double inc, Scalar scl, Mat &image, double SHIFT=0)

Static Public Member Functions

- static double [rangeSymm](#) (double ang)

Friends

- ostream & [operator<<](#) (ostream &out, const [Dubins](#) &data)

Additional Inherited Members

6.11.1 Detailed Description

```
template<class T>
class Dubins< T >
```

Class to store a [Dubins](#) curve. This class inherits from [Curve](#) and is composed of three [DubinsArc](#).

Template Parameters

<i>T</i>	The type of the classes Curve and DubinsArc .
----------	---

6.11.2 Constructor & Destructor Documentation

6.11.2.1 Dubins() [1/4]

```
template<class T>
Dubins< T >::Dubins ( ) [inline]
```

Plain constructor for [Dubins](#) that calls the plain constructor of [Curve](#) and [DubinsArc](#).

6.11.2.2 Dubins() [2/4]

```
template<class T>
Dubins< T >::Dubins (
    const Configuration2< T > _P0,
    const Configuration2< T > _P1,
    const double _K = KMAX ) [inline]
```

Constructor that takes an initial and a final [Configuration2](#), a curvature and compute the [Dubins](#) that connect the two configurations.

Parameters

<i>_P0</i>	Initial Configuration2 .
<i>_P1</i>	Final Configuration2 .
<i>_K</i>	Curvature.

6.11.2.3 Dubins() [3/4]

```
template<class T>
Dubins< T >::Dubins (
    const Point2< T > _P0,
    const Point2< T > _P1,
    const Angle _th0,
    const Angle _th1,
    const double _K = KMAX ) [inline]
```

Constructor that takes an initial and a final [Point2](#), the two respectively [Angles](#) and the curvature and computes the [Dubins](#).

Parameters

<code>_P0</code>	Initial Point2 .
<code>_P1</code>	Final Point2 .
<code>_th0</code>	Initial Angle
<code>_th1</code>	Final Angle
<code>_K</code>	Curvature.

6.11.2.4 `Dubins()` [4/4]

```
template<class T>
Dubins< T >::Dubins (
    const T x0,
    const T y0,
    const Angle _th0,
    const T x1,
    const T y1,
    const Angle _th1,
    const double _K = KMAX ) [inline]
```

Constructor that takes the initial and final coordinates, the respective [Angles](#) and the curvature and compute a [Dubins](#).

Parameters

<code>x0</code>	Initial abscissa coordinate.
<code>y0</code>	Initial ordinate coordinate.
<code>_th0</code>	Initial Angle .
<code>x1</code>	Final abscissa coordinate.
<code>y1</code>	Final ordinate coordinate.
<code>_th1</code>	Final Angle .
<code>_K</code>	Curvature of the curve.

6.11.3 Member Function Documentation

6.11.3.1 `check()`

```
template<class T>
bool Dubins< T >::check (
    double s1,
    double k0,
    double s2,
    double k1,
    double s3,
```

```
double k2,
Angle th0,
Angle th1 ) const [inline]
```

Function that checks that the values got in `shortest_path()` are right.

Parameters

<i>s1</i>	Length for the first DubinsArc .
<i>k0</i>	Curvature for the first DubinsArc .
<i>s2</i>	Length for the second DubinsArc .
<i>k1</i>	Curvature for the second DubinsArc .
<i>s3</i>	Length for the third DubinsArc .
<i>k2</i>	Curvature for the third DubinsArc .
<i>th0</i>	Initial angles (standardised).
<i>th1</i>	Final angles (standardised).

Returns

`true` if the values where correct, `false` otherwise.

6.11.3.2 draw()

```
template<class T>
void Dubins< T >::draw (
    double dimX,
    double dimY,
    double inc,
    Scalar scl,
    Mat & image,
    double SHIFT = 0 ) [inline]
```

Function to draw the [Dubins](#).

Parameters

<i>dimX</i>	The dimension X of the Mat.
<i>dimY</i>	The dimension Y of the Mat.
<i>inc</i>	The value to scale each point.
<i>scl</i>	The Scalar that defines the color to use.
<i>image</i>	The Mat where to draw the points.
<i>SHIFT</i>	The value to use to shift the points to make them stay inside the matrix.

6.11.3.3 getA1()

```
template<class T>
DubinsArc<T> Dubins< T >::getA1 ( ) const [inline]
```

Returns the first [DubinsArc](#).

6.11.3.4 getA2()

```
template<class T>
DubinsArc<T> Dubins< T >::getA2 ( ) const [inline]
```

Returns the second [DubinsArc](#).

6.11.3.5 getA3()

```
template<class T>
DubinsArc<T> Dubins< T >::getA3 ( ) const [inline]
```

Returns the third [DubinsArc](#).

6.11.3.6 getId()

```
template<class T>
double Dubins< T >::getId ( ) [inline]
```

Returns the id of the [Dubins](#), that is the set of three maneuvers that creates the curve.

6.11.3.7 getKMax()

```
template<class T>
double Dubins< T >::getKMax ( ) const [inline]
```

Returns the maximum curvature of the [Dubins](#).

6.11.3.8 length()

```
template<class T>
double Dubins< T >::length ( ) const [inline]
```

Returns the length of the [Dubins](#).

6.11.3.9 LRL()

```
template<class T>
double* Dubins< T >::LRL (
    double th0,
    double th1,
    double _kmax ) [inline]
```

Function to compute the set of maneuvers Left Right Left.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.10 LSL()

```
template<class T>
double* Dubins< T >::LSL (
    double th0,
    double th1,
    double _kmax ) [inline]
```

Function to compute the set of maneuvers Left Straight Left.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.11 LSR()

```
template<class T>
double* Dubins< T >::LSR (
    double th0,
    double th1,
    double _kmax ) [inline]
```

Function to compute the set of maneuvers Left Straight Right.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.12 rangeSymm()

```
template<class T>
static double Dubins< T >::rangeSymm (
    double ang ) [inline], [static]
```

Normalize an angular difference $(-\pi, \pi]$.

Parameters

<i>ang</i>	The value of the angle to be normalized.
------------	--

Returns

The normalized angle.

6.11.3.13 RLR()

```
template<class T>
double* Dubins< T >::RLR (
    double th0,
    double th1,
    double _kmax ) [inline]
```

Function to compute the set of maneuvers Right Left Right.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.14 RSL()

```
template<class T>
double* Dubins< T >::RSL (
```



```
double th0,
double th1,
double _kmax ) [inline]
```

Function to compute the set of maneuvers Right Straight Left.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.15 RSR()

```
template<class T>
double* Dubins< T >::RSR (
    double th0,
    double th1,
    double _kmax ) [inline]
```

Function to compute the set of maneuvers Right Straight Right.

Parameters

<i>th0</i>	The initial angle standardized.
<i>th1</i>	The final angle standardized.
<i>_kmax</i>	The maximum curvature.

Returns

An array of dimension 3 containing the length of the 3 maneuvers.

6.11.3.16 scaleFromStandard()

```
template<class T>
Tuple<double> Dubins< T >::scaleFromStandard (
    double lambda,
    double sc_s1,
    double sc_s2,
    double sc_s3 ) [inline]
```

6.11.3.17 scaleToStandard()

```
template<class T>
Tuple<double> Dubins< T >::scaleToStandard ( ) [inline]
```

Function to compute standardize the parameters. This function computes the initial and final angles as if the reference system is P0(-1,0), P1(0,1). This allows to simplify the calculations to find the best set of maneuvers.

Returns

A [Tuple](#) of [double](#) containing the standardised initial and final angle, the new curvature and the parameter lambda that allows to compute the real dimension lengths.

6.11.3.18 shortest_path()

```
template<class T>
int Dubins< T >::shortest_path ( ) [inline]
```

This function computes the shortest path for the [Dubins](#) constructed. First the values are scaled. Then the six sets of maneuvers are computed and their lengths are stored. Once the set that gives the [Dubins](#) with the minimum length is found, the lengths are rescaled and the [DubinsArc](#) are created. In the process length is also computed.

Returns

The id of the set of maneuvers.

6.11.3.19 splitIt()

```
template<class T>
Tuple<Tuple<Point2<double> > > Dubins< T >::splitIt (
    int _arch = 0,
    double _L = PIECE\_LENGTH ) [inline]
```

Function to split a [Dubins](#) in points.

Parameters

<code>_arch</code>	If defined returns only the points for a single DubinsArc .
<code>_L</code>	The distance from one point to another.

Returns

A [Tuple](#) containing three [Tuple](#) of [Point2](#) (one for each arc) containing the computed points.

6.11.3.20 to_string()

```
template<class T>
stringstream Dubins< T >::to_string ( ) const [inline]
```

This function create a stringstream object containing infos about the [Dubins](#).

Returns

A string stream.

6.11.4 Friends And Related Function Documentation

6.11.4.1 operator<<

```
template<class T>
ostream& operator<< (
    ostream & out,
    const Dubins< T > & data ) [friend]
```

This function overload the << operator so to print with `std::cout` the values of the [Dubins](#), that is printing the 3 DubinsArcs.

Parameters

in	<i>out</i>	The out stream.
in	<i>data</i>	The Dubins to print.

Returns

An output stream to be printed.

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

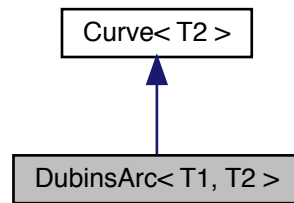
- [src/include/dubins.hh](#)

6.12 DubinsArc< T1, T2 > Class Template Reference

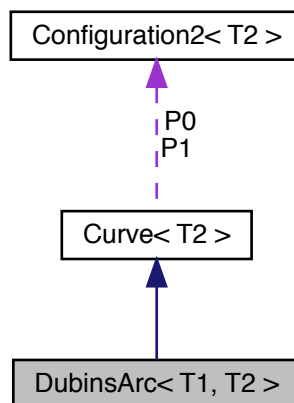
Class to store a maneuver of [Dubins](#). It inherits from [Curve](#). Since each [Dubins](#) is formed of atmost 3 maneuvers, this class is meant to store one of this maneuver, which can be L, R or S respectively Left, Right, Straight.

```
#include <dubins.hh>
```

Inheritance diagram for DubinsArc< T1, T2 >:



Collaboration diagram for DubinsArc< T1, T2 >:



Public Member Functions

- `DubinsArc ()`
- `DubinsArc (const Configuration2< T2 > _P0, const T1 _k, const T1 _l)`
- `T1 getK () const`
Returns the curvature of the arc.
- `T1 length () const`
Returns the length of the arc.
- `Tuple< Point2< T2 > > splitIt (double _L=PIECE_LENGTH)`
Splits the DubinsArc in pieces of _L length. This function starts from the beginning of the arc and computes n new arcs through the `circline()` function using the curvature of the DubinsArc and _L as the length.
- `stringstream to_string () const`
- `void draw (double dimX, double dimY, double inc, Scalar scl, Mat &image, double SHIFT)`

Friends

- ostream & [operator<<](#) (ostream &out, const [DubinsArc](#) &data)

Additional Inherited Members

6.12.1 Detailed Description

```
template<class T1 = double, class T2 = double>
class DubinsArc< T1, T2 >
```

Class to store a maneuver of [Dubins](#). It inherits from [Curve](#). Since each [Dubins](#) is formed of atmost 3 maneuvers, this class is meant to store one of this maneuver, which can be L, R or S respectively Left, Right, Straight.

Template Parameters

<i>T1</i>	The type of Length and Curvature.
<i>T2</i>	The type of the class Curve .

6.12.2 Constructor & Destructor Documentation

6.12.2.1 DubinsArc() [1/2]

```
template<class T1 = double, class T2 = double>
DubinsArc< T1, T2 >::DubinsArc ( ) [inline]
```

Plain constructor of [DubinsArc](#) that sets L and K to 0 and creates a plain [Curve](#).

6.12.2.2 DubinsArc() [2/2]

```
template<class T1 = double, class T2 = double>
DubinsArc< T1, T2 >::DubinsArc (
    const Configuration2< T2 > _P0,
    const T1 _k,
    const T1 _l ) [inline]
```

Creates a new [DubinsArc](#) given a start [Configuration2](#), the curvature and the length of the arc calling [circline\(\)](#).

Parameters

<i>_P0</i>	The starting Configuration2 .
<i>_k</i>	The curvature of the DubinsArc .
<i>_l</i>	The length of the DubinsArc .

6.12.3 Member Function Documentation

6.12.3.1 draw()

```
template<class T1 = double, class T2 = double>
void DubinsArc< T1, T2 >::draw (
    double dimX,
    double dimY,
    double inc,
    Scalar scl,
    Mat & image,
    double SHIFT ) [inline]
```

This function draws the [DubinsArc](#).

Parameters

<i>dimX</i>	The dimension X of the Mat.
<i>dimY</i>	The dimension Y of the Mat.
<i>inc</i>	The value to scale each point.
<i>scl</i>	The Scalar that defines the color to use.
<i>image</i>	The Mat where to draw the points.
<i>SHIFT</i>	The value to use to shift the points to make them stay inside the matrix.

6.12.3.2 getK()

```
template<class T1 = double, class T2 = double>
T1 DubinsArc< T1, T2 >::getK ( ) const [inline]
```

Returns the curvature of the arc.

6.12.3.3 length()

```
template<class T1 = double, class T2 = double>
T1 DubinsArc< T1, T2 >::length ( ) const [inline]
```

Returns the length of the arc.

6.12.3.4 splitt()

```
template<class T1 = double, class T2 = double>
Tuple<Point2<T2> > DubinsArc< T1, T2 >::splitIt (
    double _L = PIECE_LENGTH ) [inline]
```

Splits the [DubinsArc](#) in pieces of `_L` length. This function starts from the beginning of the arc and computes n new arcs through the [circline\(\)](#) function using the curvature of the [DubinsArc](#) and `_L` as the length.

Parameters

\leftrightarrow	The length that each points should have.
$_ \leftrightarrow$	
L	

Returns

A [Tuple](#) of [Configuration2](#)s representing the points along the arc.

6.12.3.5 to_string()

```
template<class T1 = double, class T2 = double>
stringstream DubinsArc< T1, T2 >::to_string ( ) const [inline]
```

This function create a stringstream object containing infos about the [DubinsArc](#).

Returns

A string stream.

6.12.4 Friends And Related Function Documentation

6.12.4.1 operator<<

```
template<class T1 = double, class T2 = double>
ostream& operator<< (
    ostream & out,
    const DubinsArc< T1, T2 > & data ) [friend]
```

This function overload the << operator so to print with `std::cout` the values of the [DubinsArc](#), that is [Curve](#) values more the length and the curvature.

Parameters

in	<i>out</i>	The out stream.
in	<i>data</i>	The DubinsArc to print.

Returns

An output stream to be printed.

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

- [src/include/dubins.hh](#)

6.13 DubinsSet< T > Class Template Reference

Given a set of point, compute the shortest set of [Dubins](#) that allows to go from start to end through all points.

```
#include <dubins.hh>
```

Public Member Functions

- [DubinsSet](#) ([Tuple](#)< [Dubins](#)< T > > _dubinses, double _kmax=[KMAX](#))
- [DubinsSet](#) ([Tuple](#)< [Configuration2](#)< T > > _confs, double _kmax=[KMAX](#))
- [DubinsSet](#) ([Configuration2](#)< T > start, [Configuration2](#)< T > end, [Tuple](#)< [Point2](#)< T > > _points, double _kmax=[KMAX](#))
Constructor that given a start [Configuration2](#), an end [Configuration2](#) and a [Tuple](#) of [Point2](#), computes the best path from start to end through all points by brute forcing all possible angles. Since this approach is based on a brute force algorithm, it's best not to use this on too many points.
- [DubinsSet](#) ([Tuple](#)< [Point2](#)< T > > _points, double _kmax=[KMAX](#))
Constructor that computes a series of [Dubins](#) given only [Point2](#) points via brute force. Since this approach is based on a brute force algorithm, it's best not to use this on too many points.
- void [find_best](#) ([Tuple](#)< [Point2](#)< T > > _points, [Tuple](#)< [Angle](#) > &_angles, [Angle](#) area=[A_2PI](#), double tries=4.0, double _kmax=[KMAX](#))
- double [getLength](#) ()
Returns the Length of the set of [Dubins](#).
- double [getKmax](#) ()
Returns the maximum curvature.
- double [getSize](#) ()
Returns the number of [Dubins](#) stored.
- [Tuple](#)< [Dubins](#)< T > > [getDubinses](#) ()
Returns a [Tuple](#) containing all the [Dubins](#).
- [Dubins](#)< T > [getDubins](#) (int id)
- stringstream [to_string](#) ()

Friends

- ostream & [operator](#)<< (ostream &out, [DubinsSet](#) &data)

6.13.1 Detailed Description

```
template<class T>
class DubinsSet< T >
```

Given a set of point, compute the shortest set of [Dubins](#) that allows to go from start to end through all points.

Template Parameters

<i>T</i>	Type for class Dubins .
----------	---

6.13.2 Constructor & Destructor Documentation

6.13.2.1 DubinsSet() [1/4]

```
template<class T >
DubinsSet< T >::DubinsSet (
    Tuple< Dubins< T > > _dubinses,
    double _kmax = KMAX ) [inline]
```

Constructor that given a [Tuple](#) of [Dubins](#) computes stores all of them.

Parameters

<code>_dubinses</code>	The Tuple of Dubins .
<code>_kmax</code>	The maximum curvature.

6.13.2.2 DubinsSet() [2/4]

```
template<class T >
DubinsSet< T >::DubinsSet (
    Tuple< Configuration2< T > > _confs,
    double _kmax = KMAX ) [inline]
```

Constructor that takes a [Tuple](#) of [Configuration2](#)s and computes the [Dubins](#) between them.

Parameters

<code>_confs</code>	The Tuple of Configuration2 s.
<code>_kmax</code>	The maximum curvature to be used.

6.13.2.3 DubinsSet() [3/4]

```
template<class T >
DubinsSet< T >::DubinsSet (
    Configuration2< T > start,
    Configuration2< T > end,
    Tuple< Point2< T > > _points,
    double _kmax = KMAX ) [inline]
```

Constructor that given a start [Configuration2](#), an end [Configuration2](#) and a [Tuple](#) of [Point2](#), computes the best path from start to end through all points by brute forcing all possible angles. Since this approach is based on a brute force algorithm, it's best not to use this on too many points.

Parameters

<i>start</i>	Configuration2 of start.
<i>end</i>	Configuration2 of end.
<i>_points</i>	Tuple of Point2 containing all the intermediate points.
<i>_kmax</i>	The maximum curvature of the system.

6.13.2.4 [DubinsSet\(\)](#) [4/4]

```
template<class T >
DubinsSet< T >::DubinsSet (
    Tuple< Point2< T > > _points,
    double _kmax = KMAX ) [inline]
```

Constructor that computes a series of [Dubins](#) given only [Point2](#) points via brute force. Since this approach is based on a brute force algorithm, it's best not to use this on too many points.

Parameters

<i>_points</i>	A Tuple containing all points.
<i>_kmax</i>	The maximum curvature to be used for all Dubins .

6.13.3 Member Function Documentation

6.13.3.1 [find_best\(\)](#)

```
template<class T >
void DubinsSet< T >::find_best (
    Tuple< Point2< T > > _points,
    Tuple< Angle > & _angles,
    Angle area = A\_2PI,
    double tries = 4.0,
    double _kmax = KMAX ) [inline]
```

Function to compute the best path. This function calls [disp\(\)](#) in order to calculate all possible angles, and then creates a [Dubins](#) for each possibility choosing the one with the minimum length.

Parameters

<i>_points</i>	A Tuple of Point2 through which the path should flow.
<i>_angles</i>	A Tuple of Angle containing all base Angle .
<i>area</i>	This is the angle around each angle to be "scanned".
<i>tries</i>	The number of discretizations that should be made.
<i>_kmax</i>	The maximum curvature to be used.

6.13.3.2 getDubins()

```
template<class T >
Dubins<T> DubinsSet< T >::getDubins (
    int id ) [inline]
```

This function returns a specific [Dubins](#) from the set.

Parameters

<i>id</i>	The position of the Dubins in the set.
-----------	--

Returns

The *id*-th [Dubins](#).

6.13.3.3 getDubinses()

```
template<class T >
Tuple<Dubins<T> > DubinsSet< T >::getDubinses ( ) [inline]
```

Returns a [Tuple](#) containing all the [Dubins](#).

6.13.3.4 getKmax()

```
template<class T >
double DubinsSet< T >::getKmax ( ) [inline]
```

Returns the maximum curvature.

6.13.3.5 getLength()

```
template<class T >
double DubinsSet< T >::getLength ( ) [inline]
```

Returns the Length of the set of [Dubins](#).

6.13.3.6 `getSize()`

```
template<class T >
double DubinsSet< T >::getSize ( ) [inline]
```

Returns the number of `Dubins` stored.

6.13.3.7 `to_string()`

```
template<class T >
stringstream DubinsSet< T >::to_string ( ) [inline]
```

This function create a strinstream object containing infos about the `DubinsSet`.

Returns

A string stream.

6.13.4 Friends And Related Function Documentation

6.13.4.1 `operator<<`

```
template<class T >
ostream& operator<< (
    ostream & out,
    DubinsSet< T > & data ) [friend]
```

This function overload the `<<` operator so to print with `std::cout` the values of the `DubinsSet`, that is printing all the `Dubins` stored.

Parameters

in	<i>out</i>	The out stream.
in	<i>data</i>	The <code>DubinsSet</code> to print.

Returns

An output stream to be printed.

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

- `src/include/dubins.hh`

6.14 Filter Class Reference

```
#include <filter.hh>
```

Public Member Functions

- [Filter](#) ()
Default constructor: it set all values to 0.
- [Filter](#) (int _low_h, int _low_s, int _low_v, int _high_h, int _high_s, int _high_v)
Constructor that sets all the values.
- [Filter](#) (vector< [int](#) > v)
Constructor from a vector.
- Scalar [Low](#) ()
Returns a Scalar containing the lower boudary.
- Scalar [High](#) ()
Returns a Scalar containing the lower boudary.
- stringstream [to_string](#) () const
Save value in a stringstream.
- [Filter copy](#) (const [Filter](#) &fil)
A function to copy a filter to this.
- [Filter operator=](#) (const [Filter](#) &filt)
Overload of operator =. It just calls the copy function.
- [operator vector< int >](#) () const
Overload of operator cast to vector<int>.

Public Attributes

- [int low_h](#)
Lower value for hue.
- [int low_s](#)
Lower value for saturation.
- [int low_v](#)
Lower value for value.
- [int high_h](#)
Higher value for hue.
- [int high_s](#)
Higher value for saturation.
- [int high_v](#)
Higher value for value.

Friends

- ostream & [operator<<](#) (ostream &out, const [Filter](#) &data)

6.14.1 Detailed Description

A class to store the values for an HSV filter with lower and higher boundary.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 Filter() [1/3]

```
Filter::Filter ( ) [inline]
```

Default constructor: it set all values to 0.

6.14.2.2 Filter() [2/3]

```
Filter::Filter (
    int _low_h,
    int _low_s,
    int _low_v,
    int _high_h,
    int _high_s,
    int _high_v ) [inline]
```

Constructor that sets all the values.

Parameters

<code>_low↔ _h</code>	Lower value for hue
<code>_low↔ _s</code>	Lower value for saturation
<code>_low↔ _v</code>	Lower value for value
<code>_high↔ _h</code>	Higher value for hue
<code>_high↔ _s</code>	Higher value for saturation
<code>_high↔ _v</code>	Higher value for value

6.14.2.3 Filter() [3/3]

```
Filter::Filter (
    vector< int > v ) [inline]
```

Constructor from a vector.

Parameters

<i>v</i>	The vector containing the 6 values. Mind that they must be 6.
----------	---

6.14.3 Member Function Documentation

6.14.3.1 copy()

```
Filter Filter::copy (
    const Filter & fil ) [inline]
```

A function to copy a filter to this.

Parameters

<i>fil</i>	The filter to be copied.
------------	--------------------------

Returns

this filter with the new values copied.

6.14.3.2 High()

```
Scalar Filter::High ( ) [inline]
```

Returns a Scalar containing the lower boudary.

6.14.3.3 Low()

```
Scalar Filter::Low ( ) [inline]
```

Returns a Scalar containing the lower boudary.

6.14.3.4 operator vector< int >()

```
Filter::operator vector< int > ( ) const [inline]
```

Overload of operator cast to vector<int>.

Returns

A vector containing the 6 values.

6.14.3.5 operator=()

```
Filter Filter::operator= (
    const Filter & filt ) [inline]
```

Overload of operator =. It just calls the copy function.

Parameters

<i>filt</i>	The filter to be copied.
-------------	--------------------------

Returns

this filter with the new values copied.

6.14.3.6 to_string()

```
stringstream Filter::to_string ( ) const [inline]
```

Save value in a stringstream.

Returns

A stringstream containing the values of both boundaries.

6.14.4 Friends And Related Function Documentation

6.14.4.1 operator<<

```
ostream& operator<< (
    ostream & out,
    const Filter & data ) [friend]
```

This function overload the << operator so to print with `std::cout`.

Parameters

in	<i>out</i>	The out stream.
in	<i>data</i>	The filter to print.

Returns

An output stream to be printed.

6.14.5 Member Data Documentation

6.14.5.1 high_h

```
int Filter::high_h
```

Higher value for hue.

6.14.5.2 high_s

```
int Filter::high_s
```

Higher value for saturation.

6.14.5.3 high_v

```
int Filter::high_v
```

Higher value for value.

6.14.5.4 low_h

```
int Filter::low_h
```

Lower value for hue.

6.14.5.5 low_s

```
int Filter::low_s
```

Lower value for saturation.

6.14.5.6 low_v

```
int Filter::low_v
```

Lower value for value.

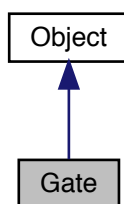
The documentation for this class was generated from the following file:

- [src/include/filter.hh](#)

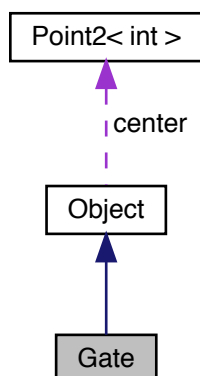
6.15 Gate Class Reference

```
#include <objects.hh>
```

Inheritance diagram for Gate:



Collaboration diagram for Gate:



Public Member Functions

- [Gate](#) (vector< [Point2](#)< [int](#) > > &vp)
Constructor of the gate class and automatically compute center and radius.
- string [toString](#) ()
Generate a string that describe the gate.
- void [print](#) ()
Print the describing string of the gate.

Additional Inherited Members

6.15.1 Constructor & Destructor Documentation

6.15.1.1 Gate()

```
Gate::Gate (
    vector< Point2< int > > & vp )
```

Constructor of the gate class and automatically compute center and radius.

Parameters

in	vp	Vector of points that is the convex hull of the gate.
--------------------	--------------------	---

Returns

Return the created gate.

6.15.2 Member Function Documentation

6.15.2.1 print()

```
void Gate::print ( )
```

Print the describing string of the gate.

6.15.2.2 toString()

```
string Gate::toString ( )
```

Generate a string that describe the gate.

Returns

The generated string.

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

- [src/include/objects.hh](#)
- [src/objects.cc](#)

6.16 CameraCapture::input_options_t Struct Reference

Structure for store the input option for the class [CameraCapture](#).

```
#include <camera_capture.hh>
```

Public Member Functions

- [input_options_t](#) ()
- [input_options_t](#) (const uint32_t frameHeight_px_, const uint32_t frameWidth_px_, const uint32_t cameraFPS_, const uint32_t cameraId_)↩
- [input_options_t](#) (const [input_options_t](#) &inpOpt)

Public Attributes

- uint32_t [frameHeight_px](#)
- uint32_t [frameWidth_px](#)
- uint32_t [cameraFPS](#)
- char [nameCamera](#) [20]

6.16.1 Detailed Description

Structure for store the input option for the class [CameraCapture](#).

[frameHeight_px](#) desidered height of the camera

[frameWidth_px](#) desidered width of the frame of the camera

[cameraFPS](#) desidered FPS of the camera

[nameCamera](#) is the camera filedescriptor (max 20 char)

6.16.2 Constructor & Destructor Documentation

6.16.2.1 input_options_t() [1/3]

```
CameraCapture::input_options_t::input_options_t ( )
```

6.16.2.2 input_options_t() [2/3]

```
CameraCapture::input_options_t::input_options_t (
    const uint32_t frameHeight_px_,
    const uint32_t frameWidth_px_,
    const uint32_t cameraFPS_,
    const uint32_t cameraId_ )
```

6.16.2.3 input_options_t() [3/3]

```
CameraCapture::input_options_t::input_options_t (
    const input\_options\_t & inpOpt )
```

6.16.3 Member Data Documentation

6.16.3.1 cameraFPS

```
uint32_t CameraCapture::input_options_t::cameraFPS
```

6.16.3.2 frameHeight_px

```
uint32_t CameraCapture::input_options_t::frameHeight_px
```

6.16.3.3 frameWidth_px

```
uint32_t CameraCapture::input_options_t::frameWidth_px
```

6.16.3.4 nameCamera

```
char CameraCapture::input_options_t::nameCamera[20]
```

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

- [src/include/camera_capture.hh](#)
- [src/camera_capture.cc](#)

6.17 ClipperLib::Int128 Class Reference

Public Member Functions

- [Int128](#) ([long64](#) _lo=0)
- [Int128](#) (const [Int128](#) &val)
- [Int128](#) (const [long64](#) &_hi, const [ulong64](#) &_lo)
- [Int128](#) & [operator=](#) (const [long64](#) &val)
- bool [operator==](#) (const [Int128](#) &val) const
- bool [operator !=](#) (const [Int128](#) &val) const
- bool [operator >](#) (const [Int128](#) &val) const
- bool [operator <](#) (const [Int128](#) &val) const
- bool [operator >=](#) (const [Int128](#) &val) const
- bool [operator <=](#) (const [Int128](#) &val) const
- [Int128](#) & [operator+=](#) (const [Int128](#) &rhs)
- [Int128](#) [operator+](#) (const [Int128](#) &rhs) const
- [Int128](#) & [operator -=](#) (const [Int128](#) &rhs)
- [Int128](#) [operator -](#) (const [Int128](#) &rhs) const
- [Int128](#) [operator-](#) () const
- [operator double](#) () const

Public Attributes

- [ulong64](#) lo
- [long64](#) hi

6.17.1 Constructor & Destructor Documentation

6.17.1.1 Int128() [1/3]

```
ClipperLib::Int128::Int128 (
    long64 _lo = 0 ) [inline]
```

6.17.1.2 Int128() [2/3]

```
ClipperLib::Int128::Int128 (
    const Int128 & val ) [inline]
```

6.17.1.3 Int128() [3/3]

```
ClipperLib::Int128::Int128 (
    const long64 & _hi,
    const ulong64 & _lo ) [inline]
```

6.17.2 Member Function Documentation

6.17.2.1 operator !=()

```
bool ClipperLib::Int128::operator != (
    const Int128 & val ) const [inline]
```

6.17.2.2 operator -()

```
Int128 ClipperLib::Int128::operator - (
    const Int128 & rhs ) const [inline]
```

6.17.2.3 operator -=()

```
Int128& ClipperLib::Int128::operator -= (
    const Int128 & rhs ) [inline]
```

6.17.2.4 operator >()

```
bool ClipperLib::Int128::operator > (
    const Int128 & val ) const [inline]
```

6.17.2.5 operator >=()

```
bool ClipperLib::Int128::operator >= (
    const Int128 & val ) const [inline]
```

6.17.2.6 operator double()

```
ClipperLib::Int128::operator double ( ) const [inline]
```

6.17.2.7 operator+()

```
Int128 ClipperLib::Int128::operator+ (
    const Int128 & rhs ) const [inline]
```

6.17.2.8 operator+=()

```
Int128& ClipperLib::Int128::operator+= (
    const Int128 & rhs ) [inline]
```

6.17.2.9 operator-()

```
Int128 ClipperLib::Int128::operator- ( ) const [inline]
```

6.17.2.10 operator<()

```
bool ClipperLib::Int128::operator< (
    const Int128 & val ) const [inline]
```

6.17.2.11 operator<=()

```
bool ClipperLib::Int128::operator<= (
    const Int128 & val ) const [inline]
```


6.17.2.12 operator=()

```
Int128& ClipperLib::Int128::operator= (
    const long64 & val ) [inline]
```

6.17.2.13 operator==()

```
bool ClipperLib::Int128::operator== (
    const Int128 & val ) const [inline]
```

6.17.3 Member Data Documentation

6.17.3.1 hi

```
long64 ClipperLib::Int128::hi
```

6.17.3.2 lo

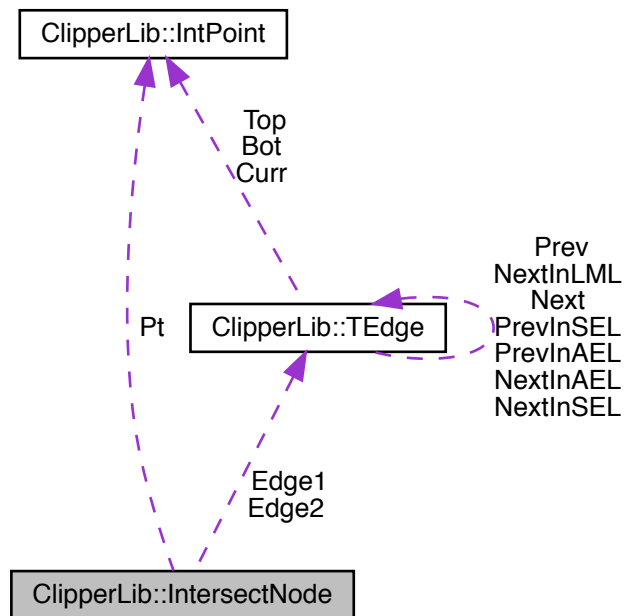
```
ulong64 ClipperLib::Int128::lo
```

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

- [src/clipper.cc](#)

6.18 ClipperLib::IntersectNode Struct Reference

Collaboration diagram for ClipperLib::IntersectNode:



Public Attributes

- `TEdge * Edge1`
- `TEdge * Edge2`
- `IntPoint Pt`

6.18.1 Member Data Documentation

6.18.1.1 Edge1

`TEdge*` `ClipperLib::IntersectNode::Edge1`

6.18.1.2 Edge2

`TEdge*` `ClipperLib::IntersectNode::Edge2`

6.18.1.3 Pt

`IntPoint` ClipperLib::IntersectNode::Pt

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

- [src/clipper.cc](#)

6.19 ClipperLib::IntPoint Struct Reference

```
#include <clipper.hh>
```

Public Member Functions

- `IntPoint` (`cInt` x=0, `cInt` y=0)

Public Attributes

- `cInt` X
- `cInt` Y

Friends

- `bool operator==` (const `IntPoint` &a, const `IntPoint` &b)
- `bool operator!=` (const `IntPoint` &a, const `IntPoint` &b)

6.19.1 Constructor & Destructor Documentation

6.19.1.1 IntPoint()

```
ClipperLib::IntPoint::IntPoint (
    cInt x = 0,
    cInt y = 0 ) [inline]
```

6.19.2 Friends And Related Function Documentation

6.19.2.1 operator!=

```
bool operator!= (
    const IntPoint & a,
    const IntPoint & b ) [friend]
```

6.19.2.2 operator==

```
bool operator== (
    const IntPoint & a,
    const IntPoint & b ) [friend]
```

6.19.3 Member Data Documentation

6.19.3.1 X

```
cInt ClipperLib::IntPoint::X
```

6.19.3.2 Y

```
cInt ClipperLib::IntPoint::Y
```

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

- [src/include/clipper.hh](#)

6.20 ClipperLib::IntRect Struct Reference

```
#include <clipper.hh>
```

Public Attributes

- [cInt](#) left
- [cInt](#) top
- [cInt](#) right
- [cInt](#) bottom

6.20.1 Member Data Documentation

6.20.1.1 bottom

`cInt` ClipperLib::IntRect::bottom

6.20.1.2 left

`cInt` ClipperLib::IntRect::left

6.20.1.3 right

`cInt` ClipperLib::IntRect::right

6.20.1.4 top

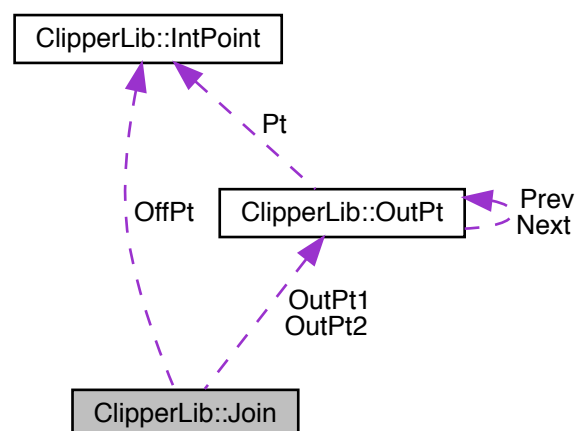
`cInt` ClipperLib::IntRect::top

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

- [src/include/clipper.hh](#)

6.21 ClipperLib::Join Struct Reference

Collaboration diagram for ClipperLib::Join:



Public Attributes

- [OutPt](#) * [OutPt1](#)
- [OutPt](#) * [OutPt2](#)
- [IntPoint](#) [OffPt](#)

6.21.1 Member Data Documentation

6.21.1.1 OffPt

[IntPoint](#) [ClipperLib::Join::OffPt](#)

6.21.1.2 OutPt1

[OutPt](#)* [ClipperLib::Join::OutPt1](#)

6.21.1.3 OutPt2

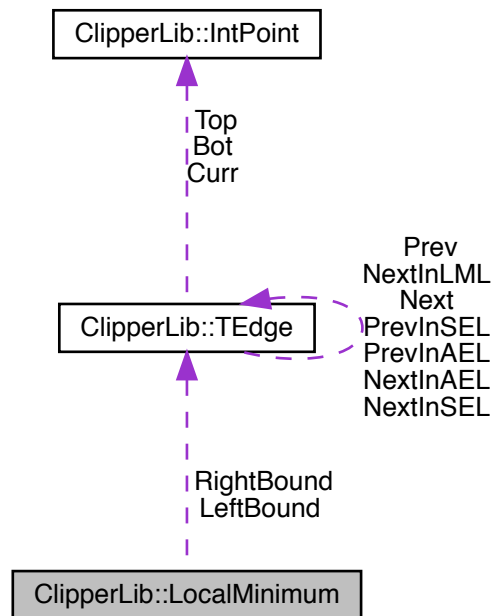
[OutPt](#)* [ClipperLib::Join::OutPt2](#)

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

- [src/clipper.cc](#)

6.22 ClipperLib::LocalMinimum Struct Reference

Collaboration diagram for ClipperLib::LocalMinimum:



Public Attributes

- `clnt Y`
- `TEdge * LeftBound`
- `TEdge * RightBound`

6.22.1 Member Data Documentation

6.22.1.1 LeftBound

`TEdge*` `ClipperLib::LocalMinimum::LeftBound`

6.22.1.2 RightBound

`TEdge*` `ClipperLib::LocalMinimum::RightBound`

6.22.1.3 Y

```
cInt ClipperLib::LocalMinimum::Y
```

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

- [src/clipper.cc](#)

6.23 ClipperLib::LocMinSorter Struct Reference

Public Member Functions

- `bool operator()` (const [LocalMinimum](#) &locMin1, const [LocalMinimum](#) &locMin2)

6.23.1 Member Function Documentation

6.23.1.1 operator>()

```
bool ClipperLib::LocMinSorter::operator() (
    const LocalMinimum & locMin1,
    const LocalMinimum & locMin2 ) [inline]
```

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

- [src/clipper.cc](#)

6.24 Mapp Class Reference

```
#include <map.hh>
```


Public Member Functions

- **Mapp** (const `int` _lengthX=1000, const `int` _lengthY=1500, const `int` _pixX=cellSize, const `int` _pixY=cellSize, const `int` _borderSize=borderSizeDefault, const `vector`< `vector`< `Point2`< `int` > > > &vvp=`vector`< `vector`< `Point2`< `int` > > >())
Constructor of the class.
- void **addObject** (const `vector`< `Point2`< `int` > > &vp, const `OBJ_TYPE` type)
Given an obstacle it is added to the map.
- void **addObjects** (const `vector`< `vector`< `Point2`< `int` > > > &vvp, const `OBJ_TYPE` type)
Given a vector objects it is added them to the map.
- void **addObjects** (const `vector`< `Obstacle` > &objs)
Given a vector of obstacles adds them to the map.
- void **addObjects** (const `vector`< `Victim` > &objs)
Given a vector of victims adds them to the map.
- void **addObjects** (const `vector`< `Gate` > &objs)
Given a vector of gates (tipically this vector contain only one element) adds it to the map.
- void **getVictimCenters** (`vector`< `Point2`< `int` > > &vp)
Add to the given vector the set of centers of the victims of the map.
- void **getGateCenter** (`vector`< `Point2`< `int` > > &vp)
Add to the given vector the center of the gate of the map.
- `OBJ_TYPE` **getPointType** (const `Point2`< `int` > &p)
Given a point return the type (status) of the cell in the map that contain it.
- bool **checkSegment** (const `Point2`< `int` > &p0, const `Point2`< `int` > &p1)
Given a segment, the function answer if that segment cross a cell with obstacles.
- bool **checkSegmentCollisionWithType** (const `Point2`< `int` > &p0, const `Point2`< `int` > &p1, const `OBJ_TYPE` type)
Given a segment and a type, the function answer if that segment cross a cell with the given type.
- `vector`< `vector`< `Point2`< `int` > > > **minPathNPoints** (const `vector`< `Point2`< `int` > > &vp)
Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.
- `vector`< `Point2`< `int` > > **minPathTwoPoints** (const `Point2`< `int` > &p0, const `Point2`< `int` > &p1)
Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.
- `vector`< `Point2`< `int` > > **sampleNPoints** (const `vector`< `vector`< `Point2`< `int` > > > &vvp, const `int` n=nPoints)
It extracts from the given vector of vector of points, a subset of points that always contains the first one and the last one of each vector.
- `vector`< `Point2`< `int` > > **sampleNPoints** (const `vector`< `Point2`< `int` > > &points, const `int` n)
It extracts from the given vector of points, a subset of points that always contains the first one and the last one.
- `vector`< `Point2`< `int` > > **samplePointsEachNCells** (const `vector`< `Point2`< `int` > > &points, const `int` step)
It extracts from the given vector of points, a subset of points that always contains the first one and the last one.
- `Mat` **createMapRepresentation** ()
*The function create an image (Mat) with the dimensions of the **Mapp** and all its objects inside.*
- void **imageAddSegments** (`Mat` &image, const `vector`< `Point2`< `int` > > &vp, const `int` thickness=3)
It add to the given image a set of (n-1) segments specified by the n points given.
- void **imageAddSegment** (`Mat` &image, const `Point2`< `int` > &p0, const `Point2`< `int` > &p1, const `int` thickness)
It add to the given image the segment defined from p0 to p1.
- void **imageAddPoints** (`Mat` &image, const `vector`< `Point2`< `int` > > &vp, const `int` radius=7)
It add to the given image a vector of points.
- void **imageAddPoint** (`Mat` &image, const `Point2`< `int` > &p, const `int` radius=7)
It add to the given image a point.
- void **printMap** ()

Print to the terminal the main informations of the Map, and its grid representation.

- string `matrixToString ()`

Generate a string (a grid of pixels) that represent the matrix.

- void `printDimensions ()`

Print to the terminal the main informations of the Map.

Protected Member Functions

- set< pair< int, int > > `cellsFromSegment` (const `Point2< int >` &p0, const `Point2< int >` &p1)

Given a segment (from p0 to p1) it return a set of all the cells that are partly cover from that segment.

- vector< `Point2< int >` > `minPathTwoPointsInternal` (const `Point2< int >` &startP, const `Point2< int >` &endP, double **distances, `Point2< int >` **parents)

Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.

- void `resetDistanceMap` (double **distances, const double value=`baseDistance`)

It reset, to the given value, the matrix of distances, to compute again the minPath search.

Protected Attributes

- `OBJ_TYPE` ** map
- const int range = 3
- const int foundLimit = 5
- const int offsetValue = 50
- int lengthX
- int lengthY
- int dimX
- int dimY
- int pixX
- int pixY
- int borderSize
- vector< `Obstacle` > vObstacles
- vector< `Victim` > vVictims
- vector< `Gate` > vGates

Static Protected Attributes

- static constexpr double `baseDistance` = -1.0
- static const int `borderSizeDefault` = 4
- static const int `cellSize` = 5
- static const int `nPoints` = 20

6.24.1 Constructor & Destructor Documentation

6.24.1.1 Mapp()

```
Mapp::Mapp (
    const int _lengthX = 1000,
    const int _lengthY = 1500,
    const int _pixX = cellSize,
    const int _pixY = cellSize,
    const int _borderSize = borderSizeDefault,
    const vector< vector< Point2< int > > > & vvp = vector< vector<Point2<int> > > ()
)
```

Constructor of the class.

Parameters

in	<i>_lengthX</i>	It is the size in pixel of the horizontal dimension.
in	<i>_lengthY</i>	It is the size in pixel of the vertical dimension.
in	<i>_pixX</i>	It is the horizontal granularity of a cell (how many pixels for each cell).
in	<i>_pixY</i>	It is the vertical granularity of a cell (how many pixels for each cell).
in	<i>_borderSize</i>	It is the dimension (defined based on cells of the map) of the border of each obstascles.
in	<i>vvp</i>	It is a vector, of vector, of point that delimit, as a convex hull, a set of obstacles in the map.

6.24.2 Member Function Documentation

6.24.2.1 addObject()

```
void Mapp::addObject (
    const vector< Point2< int > > & vp,
    const OBJ_TYPE type )
```

Given an obstacle it is added to the map.

This means that all the cells of the map that are partly cover from this obstacle will be set to its type.

Parameters

in	<i>vp</i>	It is the vector of points (convex hull) that delimit the object of interest.
in	<i>type</i>	It is the type of the given object. Defined as a OBJ_TYPE.

6.24.2.2 addObjectes() [1/4]

```
void Mapp::addObjects (
    const vector< vector< Point2< int > > > & vvp,
    const OBJ_TYPE type )
```

Given a vector objects it is added them to the map.

This means that all the cells of the map that are partly cover from these obstacles will be set to its type. It is a wrapper function of addObject.

Parameters

in	<i>vvp</i>	It is the vector of vector of points (set of convex hull) that delimit the objects of interest.
in	<i>type</i>	It is the type of the given object. Defined as a OBJ_TYPE.

6.24.2.3 addObjects() [2/4]

```
void Mapp::addObjects (
    const vector< Obstacle > & objs )
```

Given a vector of obstacles adds them to the map.

This means that all the cells of the map that are partly cover from these obstacles will be set to its type. It is a wrapper function of addObject.

Parameters

<i>in</i>	<i>objs</i>	It is the vector of obstacles to be loaded in the map structure.
-----------	-------------	--

6.24.2.4 addObjects() [3/4]

```
void Mapp::addObjects (
    const vector< Victim > & objs )
```

Given a vector of victims adds them to the map.

This means that all the cells of the map that are partly cover from these victims will be set to its type. It is a wrapper function of addObject.

Parameters

<i>in</i>	<i>objs</i>	It is the vector of victims to be loaded in the map structure.
-----------	-------------	--

6.24.2.5 addObjects() [4/4]

```
void Mapp::addObjects (
    const vector< Gate > & objs )
```

Given a vector of gates (typically this vector contain only one element) adds it to the map.

This means that all the cells of the map that are partly cover from this gate will be set to its type. It is a wrapper function of addObject.

Parameters

<i>in</i>	<i>objs</i>	It is the vector of gates to be loaded in the map structure.
-----------	-------------	--

6.24.2.6 cellsFromSegment()

```
set< pair< int, int > > Mapp::cellsFromSegment (
    const Point2< int > & p0,
    const Point2< int > & p1 ) [protected]
```

Given a segment (from p0 to p1) it return a set of all the cells that are partly cover from that segment.

Parameters

in	<i>p0</i>	First point of the segment.
in	<i>p1</i>	Second point of the segment.

Returns

A set containing all the cells, identified by their row(i or y) and column(j or x).

6.24.2.7 checkSegment()

```
bool Mapp::checkSegment (
    const Point2< int > & p0,
    const Point2< int > & p1 )
```

Given a segment, the function answer if that segment cross a cell with obstacles.

It is a wrapper for the function 'checkSegmentCollisionWithType'.

Parameters

in	<i>p0</i>	First point of the segment.
in	<i>p1</i>	Second point of the segment.

Returns

True if the obstacles were crossed, false otherwise.

6.24.2.8 checkSegmentCollisionWithType()

```
bool Mapp::checkSegmentCollisionWithType (
    const Point2< int > & p0,
    const Point2< int > & p1,
    const OBJ_TYPE type )
```

Given a segment and a type, the function answer if that segment cross a cell with the given type.

Parameters

in	<i>p0</i>	First point of the segment.
in	<i>p1</i>	Second point of the segment.
in	<i>type</i>	The type to be detected.

Returns

True if the type was found, false otherwise.

6.24.2.9 createMapRepresentation()

```
Mat Mapp::createMapRepresentation ( )
```

The function create an image (Mat) with the dimensions of the [Mapp](#) and all its objects inside.

Returns

The generated image is returned.

6.24.2.10 getGateCenter()

```
void Mapp::getGateCenter (
    vector< Point2< int > > & vp )
```

Add to the given vector the center of the gate of the map.

Parameters

out	<i>vp</i>	A vector where the requested center will be added.
-----	-----------	--

6.24.2.11 getPointType()

```
OBJ\_TYPE Mapp::getPointType (
    const Point2< int > & p )
```

Given a point return the type (status) of the cell in the map that contain it.

Parameters

in	<i>p</i>	The point of which we want to know the informations.
----	----------	--

Returns

The type (OBJ_TYPE) of the cell.

6.24.2.12 getVictimCenters()

```
void Mapp::getVictimCenters (
    vector< Point2< int > > & vp )
```

Add to the given vector the set of centers of the victims of the map.

Parameters

out	<i>vp</i>	A vector where the requested centers will be added.
-----	-----------	---

6.24.2.13 imageAddPoint()

```
void Mapp::imageAddPoint (
    Mat & image,
    const Point2< int > & p,
    const int radius = 7 )
```

It add to the given image a point.

Parameters

	<i>[in/out]</i>	map The image where the points will be added.
in	<i>p</i>	The point to add.
in	<i>radius</i>	The radius of the point to be drawn.

6.24.2.14 imageAddPoints()

```
void Mapp::imageAddPoints (
    Mat & image,
    const vector< Point2< int > > & vp,
    const int radius = 7 )
```

It add to the given image a vector of points.

Parameters

	<i>[in/out]</i>	map The image where the point will be added.
in	<i>vp</i>	The vecotor of points to add.
in	<i>radius</i>	The radius of the points to be drawn.

6.24.2.15 imageAddSegment()

```
void Mapp::imageAddSegment (
    Mat & image,
    const Point2< int > & p0,
    const Point2< int > & p1,
    const int thickness )
```

It add to the given image the segment defined from p0 to p1.

Parameters

	<i>[in/out]</i>	map The image where the segment will be added.
in	<i>p0</i>	The first point of the segment.
in	<i>p1</i>	The end point of the segment.
in	<i>thickness</i>	The thickness of the line to be drawn.

6.24.2.16 imageAddSegments()

```
void Mapp::imageAddSegments (
    Mat & image,
    const vector< Point2< int > > & vp,
    const int thickness = 3 )
```

It add to the given image a set of (n-1) segments specified by the n points given.

Parameters

	<i>[in/out]</i>	map The image where the segments will be added.
in	<i>vp</i>	The vector of points that identify the segments.
in	<i>thickness</i>	The thickness of the lines to be drawn.

6.24.2.17 matrixToString()

```
string Mapp::matrixToString ( )
```

Generate a string (a grid of pixels) that represent the matrix.

Returns

The generated string.

6.24.2.18 minPathNPoints()

```
vector< vector< Point2< int > > > Mapp::minPathNPoints (
    const vector< Point2< int > > & vp )
```

Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.

The function is based on a Breadth-first search (BFS).

Parameters

in	<i>p0</i>	The source point.
in	<i>p1</i>	The destination point.

Returns

A vector of vector of points along the path (one for each cell of the grid of the map). Each vector is the best path for one connection, given n points there are n-1 connections.

6.24.2.19 minPathTwoPoints()

```
vector< Point2< int > > Mapp::minPathTwoPoints (
    const Point2< int > & p0,
    const Point2< int > & p1 )
```

Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.

The function is based on a Breadth-first search (BFS).

Parameters

in	<i>p0</i>	The source point.
in	<i>p1</i>	The destination point.

Returns

A vector of points along the path (one for each cell of the grid of the map).

6.24.2.20 minPathTwoPointsInternal()

```
vector< Point2< int > > Mapp::minPathTwoPointsInternal (
    const Point2< int > & startP,
    const Point2< int > & endP,
```

```
double ** distances,
Point2< int > ** parents ) [protected]
```

Given a couple of points the function compute the minimum path that connect them avoiding the intersection of OBST and BODA.

The function is based on a Breadth-first search (BFS).

Parameters

in	<i>startP</i>	The source point.
in	<i>endP</i>	The destination point.
in	<i>distances</i>	A matrix that is needed to store the distances of the visited cells.
in	<i>parents</i>	A matrix that is needed to store the parent of each cell (AKA the one that have discovered that cell with the minimum distance).

Returns

A vector of points along the path (one for each cell of the grid of the map).

6.24.2.21 printDimensions()

```
void Mapp::printDimensions ( )
```

Print to the terminal the main informations of the Map.

6.24.2.22 printMap()

```
void Mapp::printMap ( )
```

Print to the terminal the main informations of the Map, and its grid representation.

6.24.2.23 resetDistanceMap()

```
void Mapp::resetDistanceMap (
    double ** distances,
    const double value = baseDistance ) [protected]
```

It reset, to the given value, the matrix of distances, to compute again the minPath search.

Parameters

in	<i>value</i>	The value to be set.
----	--------------	----------------------

6.24.2.24 sampleNPoints() [1/2]

```
vector< Point2< int > > Mapp::sampleNPoints (
    const vector< vector< Point2< int > > > & vvp,
    const int n = nPoints )
```

It extracts from the given vector of vector of points, a subset of points that always contains the first one and the last one of each vector.

Parameters

in	<i>n</i>	The n number of points to sample.
in	<i>points</i>	The vector of vector of points to be selected.

Returns

The vector containing the subset of n points.

6.24.2.25 sampleNPoints() [2/2]

```
vector< Point2< int > > Mapp::sampleNPoints (
    const vector< Point2< int > > & points,
    const int n )
```

It extracts from the given vector of points, a subset of points that always contains the first one and the last one.

Parameters

in	<i>n</i>	The number of points to select except the extremes, it must be greater or equal than 2.
in	<i>points</i>	The vector of points to be selected.

Returns

The vector containing the subset of n points.

6.24.2.26 samplePointsEachNCells()

```
vector< Point2< int > > Mapp::samplePointsEachNCells (
    const vector< Point2< int > > & points,
    const int step )
```

It extracts from the given vector of points, a subset of points that always contains the first one and the last one.

Parameters

in	<i>step</i>	The distance (counted as cells) from the previous to the next cell, it must but ≥ 2 to have a reason.
in	<i>points</i>	The vector of points to be selected.

Returns

The vector containing the subset of points, each step cells.

6.24.3 Member Data Documentation

6.24.3.1 baseDistance

```
constexpr double Mapp::baseDistance = -1.0 [static], [protected]
```

6.24.3.2 borderSize

```
int Mapp::borderSize [protected]
```

6.24.3.3 borderSizeDefault

```
const int Mapp::borderSizeDefault = 4 [static], [protected]
```

6.24.3.4 cellSize

```
const int Mapp::cellSize = 5 [static], [protected]
```

6.24.3.5 dimX

```
int Mapp::dimX [protected]
```

6.24.3.6 dimY

```
int Mapp::dimY [protected]
```

6.24.3.7 foundLimit

```
const int Mapp::foundLimit = 5 [protected]
```

6.24.3.8 lengthX

```
int Mapp::lengthX [protected]
```

6.24.3.9 lengthY

```
int Mapp::lengthY [protected]
```

6.24.3.10 map

```
OBJ_TYPE** Mapp::map [protected]
```

6.24.3.11 nPoints

```
const int Mapp::nPoints = 20 [static], [protected]
```

6.24.3.12 offsetValue

```
const int Mapp::offsetValue = 50 [protected]
```

6.24.3.13 pixX

```
int Mapp::pixX [protected]
```

6.24.3.14 pixY

```
int Mapp::pixY [protected]
```

6.24.3.15 range

```
const int Mapp::range = 3 [protected]
```

6.24.3.16 vGates

```
vector<Gate> Mapp::vGates [protected]
```

6.24.3.17 vObstacles

```
vector<Obstacle> Mapp::vObstacles [protected]
```

6.24.3.18 vVictims

```
vector<Victim> Mapp::vVictims [protected]
```

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

- [src/include/map.hh](#)
- [src/map.cc](#)

6.25 MyException< T > Class Template Reference

```
#include <utils.hh>
```

Inherits exception.

Public Member Functions

- [MyException](#) ([EXCEPTION_TYPE](#) _type, T _a, [int](#) _b)
- const char * [what](#) () const throw ()

Public Attributes

- [EXCEPTION_TYPE](#) type
- [T](#) [a](#)
- [int](#) [b](#)

6.25.1 Constructor & Destructor Documentation

6.25.1.1 MyException()

```
template<class T >
MyException< T >::MyException (
    EXCEPTION_TYPE _type,
    T _a,
    int _b ) [inline]
```

6.25.2 Member Function Documentation

6.25.2.1 what()

```
template<class T >
const char* MyException< T >::what ( ) const throw ( ) [inline]
```

6.25.3 Member Data Documentation

6.25.3.1 a

```
template<class T >
T MyException< T >::a
```

6.25.3.2 b

```
template<class T >
int MyException< T >::b
```


6.25.3.3 type

```
template<class T >  
EXCEPTION_TYPE MyException< T >::type
```

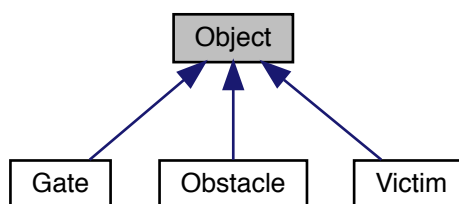
The documentation for this class was generated from the following file:

- [src/include/utils.hh](#)

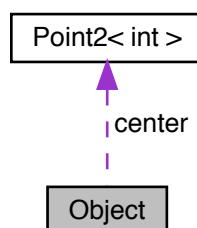
6.26 Object Class Reference

```
#include <objects.hh>
```

Inheritance diagram for Object:



Collaboration diagram for Object:



Public Member Functions

- string `toString ()`
Generate a string that describe the object.
- unsigned int `size ()`
Return the number of points of the object.
- unsigned int `nPoints ()`
Return the number of points of the object.
- vector< Point2< int > > `getPoints ()`
Return the of points of the object.
- Point2< int > `getCenter ()`
Retrieve the center of the object.
- double `getRadius ()`
Retrieve the radius of the object.
- void `computeCenter ()`
Find the representative center of the object.
- void `computeRadius ()`
Compute the radius of the object.
- void `offsetting (const int offset, const int limitX, const int limitY)`
Enlarge the object of the given offset (defined as pixels=mm in our scenario).
- bool `insidePolyApprox (Point2< int > pt)`
Check if the given point is inside the approximation shape of the object (a circle).
- bool `insidePoly (Point2< int > pt)`
Exact check if a point is inside the object (no approximation).

Protected Attributes

- vector< Point2< int > > `points`
- Point2< int > `center`
- double `radius`

6.26.1 Member Function Documentation

6.26.1.1 computeCenter()

```
void Object::computeCenter ( )
```

Find the representative center of the object.

The center is computed as the mean of the minimum and maximum x and y.

6.26.1.2 computeRadius()

```
void Object::computeRadius ( )
```

Compute the radius of the object.

This function assume that the center of the object is already computed and consistent.

6.26.1.3 getCenter()

```
Point2< int > Object::getCenter ( )
```

Retrieve the center of the object.

Returns

The center.

6.26.1.4 getPoints()

```
vector< Point2< int > > Object::getPoints ( )
```

Return the of points of the object.

Returns

The vector of points.

6.26.1.5 getRadius()

```
double Object::getRadius ( )
```

Retrieve the radius of the object.

Returns

The radius.

6.26.1.6 insidePoly()

```
bool Object::insidePoly (
    Point2< int > pt )
```

Exact check if a point is inside the object (no approximation).

Parameters

in	<i>pt</i>	The point to be checked.
----	-----------	--------------------------

Returns

True if the point is inside the object, false otherwise.

6.26.1.7 insidePolyApprox()

```
bool Object::insidePolyApprox (
    Point2< int > pt )
```

Check if the given point is inside the approximation shape of the object (a circle).

Parameters

in	<i>pt</i>	The point to be checked.
----	-----------	--------------------------

Returns

True if the point is inside the object, false otherwise.

6.26.1.8 nPoints()

```
unsigned int Object::nPoints ( )
```

Return the number of points of the object.

Returns

The number of points.

6.26.1.9 offsetting()

```
void Object::offsetting (
    const int offset,
    const int limitX,
    const int limitY )
```

Enlarge the object of the given offset (defined as pixels=mm in our scenario).

The function automatically update even the center and the radius.

Parameters

in	<i>offset</i>	The size of the offset.
in	<i>limitX</i>	The the maximum x that the point can have.
in	<i>limitY</i>	The the maximum y that the point can have.

6.26.1.10 size()

```
unsigned int Object::size ( )
```

Return the number of points of the object.

Returns

The number of points.

6.26.1.11 toString()

```
string Object::toString ( )
```

Generate a string that describe the object.

Returns

The generated string.

6.26.2 Member Data Documentation

6.26.2.1 center

```
Point2<int> Object::center [protected]
```

6.26.2.2 points

```
vector<Point2<int> > Object::points [protected]
```

6.26.2.3 radius

```
double Object::radius [protected]
```

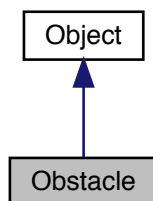
The documentation for this class was generated from the following files:

- [src/include/objects.hh](#)
- [src/objects.cc](#)

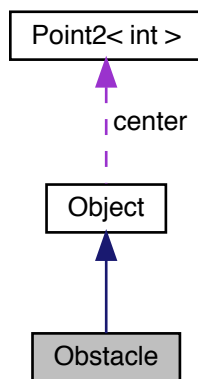
6.27 Obstacle Class Reference

```
#include <objects.hh>
```

Inheritance diagram for Obstacle:



Collaboration diagram for Obstacle:



Public Member Functions

- **Obstacle** (vector< **Point2**< **int** > > &vp)
Constructor of the obstacle class and automatically compute center and radius.
- string **toString** ()
Generate a string that describe the obstacle.
- void **print** ()
Print the describing string of the obstacle.

Additional Inherited Members

6.27.1 Constructor & Destructor Documentation

6.27.1.1 Obstacle()

```
Obstacle::Obstacle (
    vector< Point2< int > > & vp )
```

Constructor of the obstacle class and automatically compute center and radius.

Parameters

in	vp	Vector of points that is the convex hull of the obstacle.
----	----	---

Returns

Return the created obstacle.

6.27.2 Member Function Documentation

6.27.2.1 print()

```
void Obstacle::print ( )
```

Print the describing string of the obstacle.

6.27.2.2 toString()

```
string Obstacle::toString ( )
```

Generate a string that describe the obstacle.

Returns

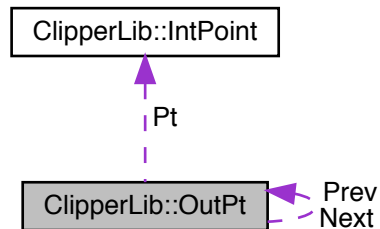
The generated string.

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

- [src/include/objects.hh](#)
- [src/objects.cc](#)

6.28 ClipperLib::OutPt Struct Reference

Collaboration diagram for ClipperLib::OutPt:



Public Attributes

- `int Idx`
- `IntPoint Pt`
- `OutPt * Next`
- `OutPt * Prev`

6.28.1 Member Data Documentation

6.28.1.1 Idx

`int` `ClipperLib::OutPt::Idx`

6.28.1.2 Next

`OutPt*` `ClipperLib::OutPt::Next`

6.28.1.3 Prev

`OutPt*` `ClipperLib::OutPt::Prev`

6.28.1.4 Pt

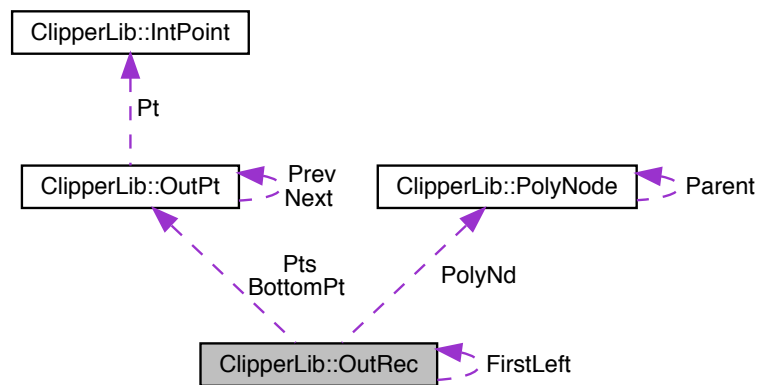
```
IntPoint ClipperLib::OutPt::Pt
```

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

- [src/clipper.cc](#)

6.29 ClipperLib::OutRec Struct Reference

Collaboration diagram for ClipperLib::OutRec:



Public Attributes

- `int` **Idx**
- `bool` **IsHole**
- `bool` **IsOpen**
- `OutRec *` **FirstLeft**
- `PolyNode *` **PolyNd**
- `OutPt *` **Pts**
- `OutPt *` **BottomPt**

6.29.1 Member Data Documentation

6.29.1.1 BottomPt

```
OutPt* ClipperLib::OutRec::BottomPt
```

6.29.1.2 FirstLeft

```
OutRec* ClipperLib::OutRec::FirstLeft
```

6.29.1.3 Idx

```
int ClipperLib::OutRec::Idx
```

6.29.1.4 IsHole

```
bool ClipperLib::OutRec::IsHole
```

6.29.1.5 IsOpen

```
bool ClipperLib::OutRec::IsOpen
```

6.29.1.6 PolyNd

```
PolyNode* ClipperLib::OutRec::PolyNd
```

6.29.1.7 Pts

```
OutPt* ClipperLib::OutRec::Pts
```

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

- [src/clipper.cc](#)

6.30 Point2< T > Class Template Reference

Class that stores two value to construct a point in 2D. The value is saved in a [Tuple](#).

```
#include <maths.hh>
```

Public Member Functions

- [Point2](#) ()
Default constructor to build an empty [Tuple](#).
- [Point2](#) (const T _x, const T _y)
Constructor that takes elements and builds a point.
- [Point2](#) (const cv::Point p)
Constructor that takes a cv::Point and returns a [Point2](#).
- [T x](#) () const
- [T y](#) () const
- void [x](#) (const T _x)
Set the abscissa value.
- void [y](#) (const T _y)
Set the ordinate value.
- template<class T1 >
void [offset](#) (const T1 _offset, const [Angle](#) th)
This function compute the offset of the point given a vector, that is the length of the vector and its angle. The angle must be an [Angle](#) variable.
- void [offset](#) (const [Point2](#)< T > p)
This function compute an offset given another point made of the abscissa offset and the ordinate offset.
- void [offset](#) (const [Tuple](#)< T > p)
This function compute an offset given a [Tuple](#) made of the abscissa offset and the ordinate offset.
- void [offset_x](#) (const T _offset)
This function compute an offset for the abscissa.
- void [offset_y](#) (const T _offset)
This function compute an offset for the ordinate.
- template<class T1 >
double [distance](#) ([Point2](#)< T1 > B, [DISTANCE_TYPE](#) dist=[EUCLIDEAN](#))
Wrapper to compute different distances. \tparam T1 The type of the elements in the second [Point2](#).
- template<class T1 >
double [MaDistance](#) ([Point2](#)< T1 > B)
Function that compute the Manhattan Distance between two points. \tparam T1 The type of the elements in the second [Point2](#).
- template<class T1 >
double [EuDistance](#) ([Point2](#)< T1 > B)
Function that compute the Euclidean Distance between two points. \tparam T1 The type of the elements in the second [Point2](#).
- stringstream [to_string](#) () const
- [Point2](#)< T > [copy](#) (const [Point2](#)< T > &A)
Copy a point into another one.
- [Point2](#)< T > [operator=](#) (const [Point2](#)< T > &A)
Overload of the = operator. Just calls [copy](#).
- bool [equal](#) (const [Point2](#)< T > &A)
Equalize two points.
- bool [operator==](#) (const [Point2](#)< T > &A)
Overload of the == operator. Just calls [equal](#).
- bool [operator!=](#) (const [Point2](#)< T > &A)
Overload of the != operator. Just calls [equal](#) and negates it.
- [operator cv::Point](#) () const
Cast to cv::Point.
- bool [operator<](#) (const [Point2](#)< T > &A)
- [Angle](#) th ([Point2](#) P1, [Angle](#)::[ANGLE_TYPE](#) type=[Angle](#)::[RAD](#))

Friends

- ostream & [operator<<](#) (ostream &out, const [Point2](#)< T > &data)
Overload of operator << to output the content of a [Point2](#).

6.30.1 Detailed Description

```
template<class T>
class Point2< T >
```

Class that stores two value to construct a point in 2D. The value is saved in a [Tuple](#).

Template Parameters

T	The type of the coordinates to be stored.
-------------------	---

6.30.2 Constructor & Destructor Documentation

6.30.2.1 [Point2\(\)](#) [1/3]

```
template<class T>
Point2< T >::Point2 ( ) [inline]
```

Default constructor to build an empty [Tuple](#).

6.30.2.2 [Point2\(\)](#) [2/3]

```
template<class T>
Point2< T >::Point2 (
    const T _x,
    const T _y ) [inline]
```

Constructor that taked to elements and builds a point.

Parameters

in	↔ _↔ x	The abscissa coordinate.
in	↔ _↔ y	The ordinate coordinate.

6.30.2.3 Point2() [3/3]

```
template<class T>
Point2< T >::Point2 (
    const cv::Point p ) [inline]
```

Constructor that takes a cv::Point and returns a [Point2](#).

Parameters

in	<i>p</i>	The cv::Point to be copied.
----	----------	-----------------------------

6.30.3 Member Function Documentation**6.30.3.1 copy()**

```
template<class T>
Point2<T> Point2< T >::copy (
    const Point2< T > & A ) [inline]
```

Copy a point into another one.

Parameters

in	<i>A</i>	point to be copied.
----	----------	---------------------

Returns

this.

6.30.3.2 distance()

```
template<class T>
template<class T1 >
double Point2< T >::distance (
    Point2< T1 > B,
    DISTANCE_TYPE dist = EUCLIDEAN ) [inline]
```

Wrapper to compute different distances. \tparam T1 The type of the elements in the second [Point2](#).

Parameters

in	<i>B</i>	The second Point2 to use for computing the distance.
in	<i>dist</i>	The type of distance to be computed.

Returns

The distance between the two points.

6.30.3.3 equal()

```
template<class T>
bool Point2< T >::equal (
    const Point2< T > & A ) [inline]
```

Equalize two points.

Parameters

in	<i>A</i>	point to be compared to.
----	----------	--------------------------

Returns

true if the two points are equal.

6.30.3.4 EuDistance()

```
template<class T>
template<class T1 >
double Point2< T >::EuDistance (
    Point2< T1 > B ) [inline]
```

Function that compute the Euclidean Distance between two points. \tparan T1 The type of the elements in the second [Point2](#).

Parameters

in	<i>B</i>	the second Point2 to use for computing the distance.
----	----------	--

Returns

The Euclidean distance between the two points.

6.30.3.5 MaDistance()

```
template<class T>
template<class T1 >
double Point2< T >::MaDistance (
    Point2< T1 > B ) [inline]
```

Function that compute the Manhattan Distance between two points. \tparan T1 The type of the elements in the second Point2.

Parameters

in	B	the second Point2 to use for computing the distance.
----	---	--

Returns

The Manhattan distance between the two points.

6.30.3.6 offset() [1/3]

```
template<class T>
template<class T1 >
void Point2< T >::offset (
    const T1 _offset,
    const Angle th ) [inline]
```

This function compute the offset of the point given a vector, that is the lenght of the vector and its angle. The angle must be an Angle variable.

Template Parameters

--	--

6.30.3.7 offset() [2/3]

```
template<class T>
void Point2< T >::offset (
    const Point2< T > p ) [inline]
```

This function compute an offset given another point made of the abscissa offset and the ordinate offset.

Parameters

in	p	The point with the offsets.
----	---	-----------------------------

Returns

1 if everything went fine, 0 otherwise.

6.30.3.8 offset() [3/3]

```
template<class T>
void Point2< T >::offset (
    const Tuple< T > p ) [inline]
```

This function compute an offset given a [Tuple](#) made of the abscissa offset and the ordinate offset.

Parameters

in	<i>p</i>	The Tuple with the offsets. Its dimension must be 2.
----	----------	--

Returns

1 if everything went fine, 0 otherwise.

6.30.3.9 offset_x()

```
template<class T>
void Point2< T >::offset_x (
    const T _offset ) [inline]
```

This function compute an offset for the abscissa.

Parameters

in	<i>_offset</i>	The offset.
----	----------------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.30.3.10 offset_y()

```
template<class T>
void Point2< T >::offset_y (
    const T _offset ) [inline]
```

This function compute an offset for the ordinate.

Parameters

in	<code>_offset</code>	The offset.
----	----------------------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.30.3.11 operator cv::Point()

```
template<class T>
Point2< T >::operator cv::Point ( ) const [inline]
```

Cast to cv::Point.

Returns

The value casted to point

6.30.3.12 operator!=(=)

```
template<class T>
bool Point2< T >::operator!= (
    const Point2< T > & A ) [inline]
```

Overload of the != operator. Just calls `equal` and negates it.

Parameters

in	<code>A</code>	point to be compared to.
----	----------------	--------------------------

Returns

true if the two configurations are different.

6.30.3.13 operator<()

```
template<class T>
bool Point2< T >::operator< (
    const Point2< T > & A ) [inline]
```

6.30.3.14 operator=()

```
template<class T>
Point2<T> Point2< T >::operator= (
    const Point2< T > & A ) [inline]
```

Overload of the = operator. Just calls copy.

Parameters

in	A	point to be copied.
----	---	---------------------

Returns

this.

6.30.3.15 operator==()

```
template<class T>
bool Point2< T >::operator== (
    const Point2< T > & A ) [inline]
```

Overload of the == operator. Just calls equal.

Parameters

in	A	point to be compared to.
----	---	--------------------------

Returns

true if the two configurations are equal.

6.30.3.16 th()

```
template<class T>
Angle Point2< T >::th (
    Point2< T > P1,
    Angle::ANGLE_TYPE type = Angle::RAD ) [inline]
```

6.30.3.17 to_string()

```
template<class T>
stringstream Point2< T >::to_string ( ) const [inline]
```

6.30.3.18 `x()` [1/2]

```
template<class T>
T Point2< T >::x ( ) const [inline]
```

Returns

The abscissa coordinate

6.30.3.19 `x()` [2/2]

```
template<class T>
void Point2< T >::x (
    const T _x ) [inline]
```

Set the abscissa value.

Parameters

in	\leftrightarrow	The new abscissa value
	\leftarrow	
	x	

Returns

1 if it was successful, 0 otherwise.

6.30.3.20 `y()` [1/2]

```
template<class T>
T Point2< T >::y ( ) const [inline]
```

Returns

The ordinate coordinate

6.30.3.21 `y()` [2/2]

```
template<class T>
void Point2< T >::y (
    const T _y ) [inline]
```

Set the ordinate value.

Parameters

in	↔	The new ordinate value
	↔ x	

Returns

1 if it was successful, 0 otherwise.

6.30.4 Friends And Related Function Documentation**6.30.4.1 operator<<**

```
template<class T>
ostream& operator<< (
    ostream & out,
    const Point2< T > & data ) [friend]
```

Overload of operator << to output the content of a [Point2](#).

Parameters

in	<i>out</i>	The output stream.
in	<i>data</i>	The Point2 to print.

Returns

An output stream to be printed.

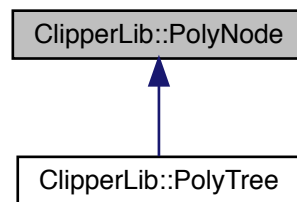
The documentation for this class was generated from the following file:

- [src/include/maths.hh](#)

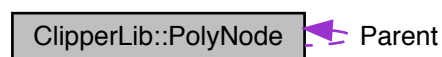
6.31 ClipperLib::PolyNode Class Reference

```
#include <clipper.hh>
```

Inheritance diagram for ClipperLib::PolyNode:



Collaboration diagram for ClipperLib::PolyNode:



Public Member Functions

- [PolyNode](#) ()
- virtual [~PolyNode](#) ()
- [PolyNode](#) * [GetNext](#) () const
- bool [IsHole](#) () const
- bool [IsOpen](#) () const
- int [ChildCount](#) () const

Public Attributes

- [Path Contour](#)
- [PolyNodes](#) Childs
- [PolyNode](#) * [Parent](#)

Friends

- class [Clipper](#)
- class [ClipperOffset](#)

6.31.1 Constructor & Destructor Documentation

6.31.1.1 PolyNode()

```
ClipperLib::PolyNode::PolyNode ( )
```

6.31.1.2 ~PolyNode()

```
virtual ClipperLib::PolyNode::~~PolyNode ( ) [inline], [virtual]
```

6.31.2 Member Function Documentation

6.31.2.1 ChildCount()

```
int ClipperLib::PolyNode::ChildCount ( ) const
```

6.31.2.2 GetNext()

```
PolyNode * ClipperLib::PolyNode::GetNext ( ) const
```

6.31.2.3 IsHole()

```
bool ClipperLib::PolyNode::IsHole ( ) const
```

6.31.2.4 IsOpen()

```
bool ClipperLib::PolyNode::IsOpen ( ) const
```

6.31.3 Friends And Related Function Documentation

6.31.3.1 Clipper

```
friend class Clipper [friend]
```

6.31.3.2 ClipperOffset

```
friend class ClipperOffset [friend]
```

6.31.4 Member Data Documentation

6.31.4.1 Childs

```
PolyNodes ClipperLib::PolyNode::Childs
```

6.31.4.2 Contour

```
Path ClipperLib::PolyNode::Contour
```

6.31.4.3 Parent

```
PolyNode* ClipperLib::PolyNode::Parent
```

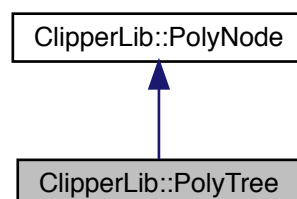
The documentation for this class was generated from the following files:

- `src/include/clipper.hh`
- `src/clipper.cc`

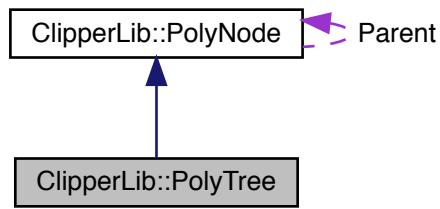
6.32 ClipperLib::PolyTree Class Reference

```
#include <clipper.hh>
```

Inheritance diagram for ClipperLib::PolyTree:



Collaboration diagram for ClipperLib::PolyTree:



Public Member Functions

- [~PolyTree](#) ()
- [PolyNode * GetFirst](#) () const
- void [Clear](#) ()
- [int Total](#) () const

Friends

- class [Clipper](#)

Additional Inherited Members

6.32.1 Constructor & Destructor Documentation

6.32.1.1 ~PolyTree()

```
ClipperLib::PolyTree::~~PolyTree ( ) [inline]
```

6.32.2 Member Function Documentation

6.32.2.1 Clear()

```
void ClipperLib::PolyTree::Clear ( )
```


6.32.2.2 GetFirst()

```
PolyNode * ClipperLib::PolyTree::GetFirst ( ) const
```

6.32.2.3 Total()

```
int ClipperLib::PolyTree::Total ( ) const
```

6.32.3 Friends And Related Function Documentation

6.32.3.1 Clipper

```
friend class Clipper [friend]
```

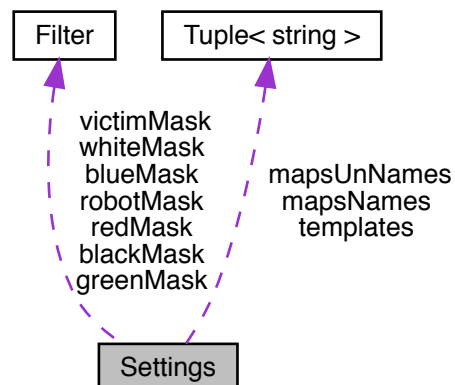
The documentation for this class was generated from the following files:

- [src/include/clipper.hh](#)
- [src/clipper.cc](#)

6.33 Settings Class Reference

```
#include <settings.hh>
```

Collaboration diagram for Settings:



Public Types

- enum `COLOR` {
`BLACK`, `RED`, `GREEN`, `VICTIMS`,
`BLUE`, `WHITE`, `ROBOT` }

Public Member Functions

- `Settings` (string `mapsFolder`="data/map", string `_templatesFolder`="data/num_template/", vector< string > `_mapsNames`={}, vector< string > `_mapsUnNames`={}, string `_calibrationFile`="data/calib_config.xml", string `_intrinsicCalibrationFile`="data/intrinsic_calibration.xml", `Filter` `_blackMask`=`Filter`(0, 0, 0, 179, 255, 70), `Filter` `_redMask`=`Filter`(15, 100, 140, 160, 255, 255), `Filter` `_greenMask`=`Filter`(54, 74, 25, 119, 255, 88), `Filter` `_victimMask`=`Filter`(0, 0, 0, 179, 255, 80), `Filter` `_blueMask`=`Filter`(100, 100, 40, 140, 200, 170), `Filter` `_whiteMask`=`Filter`(100, 100, 40, 140, 200, 170), `Filter` `_roboteMask`=`Filter`(100, 100, 40, 140, 200, 170), int `_kernelSide`=9, string `_convexHullFile`="data/convexHull.xml", vector< string > `_templates`={})
Constructor of class `Settings`. The value are all set by default. The constructor does NOT read from or write to file.
- `~Settings` ()
Destructor.
- void `save` (string `mapsFolder`="data/map", string `_templatesFolder`="data/num_template/", vector< string > `_mapsNames`={}, vector< string > `_mapsUnNames`={}, string `_calibrationFile`="data/calib_config.xml", string `_intrinsicCalibrationFile`="data/intrinsic_calibration.xml", `Filter` `_blackMask`=`Filter`(0, 0, 0, 179, 255, 70), `Filter` `_redMask`=`Filter`(15, 100, 140, 160, 255, 255), `Filter` `_greenMask`=`Filter`(54, 74, 25, 119, 255, 88), `Filter` `_victimMask`=`Filter`(0, 0, 0, 179, 255, 80), `Filter` `_blueMask`=`Filter`(100, 100, 40, 140, 200, 170), `Filter` `_whiteMask`=`Filter`(100, 100, 40, 140, 200, 170), `Filter` `_roboteMask`=`Filter`(100, 100, 40, 140, 200, 170), int `_kernelSide`=9, string `_convexHullFile`="data/convexHull.xml", vector< string > `_templates`={})
Function to change values. The value are all set by default. This function does NOT read from or write to file.
- void `writeToFile` (string `_path`="data/settings.xml")
Function to write settings to file. Default is data/settings.xml.
- void `readFromFile` (string `_path`="data/settings.xml")
Function to read from file. The data found is going to be added to the settings. Default file is data/settings.xml.
- void `clean` ()
Function to clean all settings: number types are set to 0, string are set to "", Tuples are set to `Tuple<>()` and `Filter` are set to all 0s.
- void `cleanAndRead` (string `_path`="data/settings.xml")
Function to clean all settings and then read from file. Default is data/settings.xml.
- `Tuple`< string > `maps` (`Tuple`< int > `ids`=`Tuple`< int >())
Function to return the paths of maps. If ids are not specified all maps are returned.
- `Tuple`< string > `maps` (int `id`=-1)
Function to return the path of a map. If id is negative all maps are returned.
- string `maps` (string `_mapName`)
A function to return the path of a given map.
- `Tuple`< string > `maps` (`Tuple`< string > `_mapNames`)
A function to return the paths of a given `Tuple` of maps.
- bool `addUnMap` (string `unMap`)
- `Tuple`< string > `unMaps` (`Tuple`< int > `ids`=`Tuple`< int >())
Function to return the paths of undistorted maps. If ids are not specified all undistorted maps are returned.
- `Tuple`< string > `unMaps` (int `id`=-1)
Function to return the path of an undistorted map. If id is negative all undistorted maps are returned.
- string `unMaps` (string `_unMapName`)
A function to return the path of a given undistorted map.
- `Tuple`< string > `unMaps` (`Tuple`< string > `_unMapNames`)
A function to return the paths of a given `Tuple` of undistorted maps.

- `Tuple< string > getTemplates (int id=-1)`
Function to return the path of a template. If id is negative all templates are returned.
- `string getTemplates (string _template)`
A function to return the path of a given template.
- `Tuple< string > getTemplates (Tuple< string > _templates)`
A function to return the paths of a given *Tuple* of templates.
- `void changeMask (Tuple< COLOR > color, Tuple< Filter > fil)`
Change the values of *Tuple* of filters. Mind that no write function is called.
- `void changeMask (COLOR color, Filter fil)`
Change the values of a filter. Mind that no write function is called.
- `stringstream to_string () const`
A function that creates a stringstream to print the values stored in settings.

Public Attributes

- `string mapsFolder`
A string containing the path for mapsFolder. No certainty is given about the form of this string.
- `Tuple< string > mapsNames`
A *Tuple* containing the names of the maps. These are not paths but just names.
- `Tuple< string > mapsUnNames`
A *Tuple* containing the names of the undistorted maps. These are not paths but just names.
- `string intrinsicCalibrationFile`
A string containing the path to the file containing the values of the matrix for the calibration.
- `string calibrationFile`
A string containing the path to the file containing the data for the calibration.
- `Filter blackMask`
Filter for black.
- `Filter redMask`
Filter for red.
- `Filter greenMask`
Filter for green.
- `Filter victimMask`
Filter for the victims.
- `Filter blueMask`
Filter for blue.
- `Filter whiteMask`
Filter for white.
- `Filter robotMask`
Filter for the triangle above the robot.
- `int kernelSide`
- `string convexHullFile`
AString containing the path to file containing the points of the elements in the arena.
- `string templatesFolder`
A String containing the path of the folder containing the number templates.
- `Tuple< string > templates`
A *Tuple* containing the names of the templates. These are not paths but just names.

Friends

- `ostream & operator<< (ostream &out, const Settings &data)`

6.33.1 Detailed Description

Class that stores settings for the projects such as location of files, name of maps and filters to use. Mind that when created it does not read from file by default but the function must be invoked.

6.33.2 Member Enumeration Documentation

6.33.2.1 COLOR

```
enum Settings::COLOR
```

Enumerator

BLACK	
RED	
GREEN	
VICTIMS	
BLUE	
WHITE	
ROBOT	

6.33.3 Constructor & Destructor Documentation

6.33.3.1 Settings()

```
Settings::Settings (
    string _mapsFolder = "data/map",
    string _templatesFolder = "data/num_template/",
    vector< string > _mapsNames = {},
    vector< string > _mapsUnNames = {},
    string _intrinsicCalibrationFile = "data/calib_config.xml",
    string _calibrationFile = "data/intrinsic_calibration.xml",
    Filter _blackMask = Filter(0, 0, 0, 179, 255, 70),
    Filter _redMask = Filter(15, 100, 140, 160, 255, 255),
    Filter _greenMask = Filter(54, 74, 25, 119, 255, 88),
    Filter _victimMask = Filter(0, 0, 0, 179, 255, 80),
    Filter _blueMask = Filter(100, 100, 40, 140, 200, 170),
    Filter _whiteMask = Filter(100, 100, 40, 140, 200, 170),
    Filter _robotMask = Filter(100, 100, 40, 140, 200, 170),
    int _kernelSide = 9,
    string _convexHullFile = "data/convexHull.xml",
    vector< string > _templates = {} )
```

Constructor of class [Settings](#). The value are all set by default. The constructor does NOT read from or write to file.

Parameters

<i>mapsFolder</i>	A string containing the path for mapsFolder. No certainty is given about the form of this string
<i>_templatesFolder</i>	A String containing the path of the folder containing the number templates.
<i>_mapsNames</i>	A Tuple containing the names of the maps. These are not paths but just names.
<i>_mapsUnNames</i>	A Tuple containing the names of the undistorted maps. These are not paths but just names.
<i>_calibrationFile</i>	A string containing the path to the file containing the data for the calibration.
<i>_intrinsicCalibrationFile</i>	A string containing the path to the file containing the values of the matrix for the calibration.
<i>_blackMask</i>	Filter for black.
<i>_redMask</i>	Filter for red.
<i>_greenMask</i>	Filter for green.
<i>_victimMask</i>	Filter for the victims.
<i>_blueMask</i>	Filter for blue.
<i>_whiteMask</i>	Filter for white.
<i>_robotMask</i>	Filter for the triangle above the robot.
<i>_kernelSide</i>	
<i>_convexHullFile</i>	A String containing the path to file containing the points of the elements in the arena.
<i>_templates</i>	A Tuple containing the names of the templates. These are not paths but just names.

6.33.3.2 ~Settings()

```
Settings::~~Settings ( )
```

Destructor.

6.33.4 Member Function Documentation

6.33.4.1 addUnMap()

```
bool Settings::addUnMap (
    string unMap )
```

6.33.4.2 changeMask() [1/2]

```
void Settings::changeMask (
    Tuple< COLOR > color,
    Tuple< Filter > fil )
```

Change the values of [Tuple](#) of filters. Mind that no write function is called.

Parameters

<i>color</i>	A Tuple containing the colors of the filters to change.
<i>fil</i>	The new filters to be stored.

6.33.4.3 `changeMask()` [2/2]

```
void Settings::changeMask (
    COLOR color,
    Filter fil )
```

Change the values of a filter. Mind that no write function is called.

Parameters

<i>color</i>	The filter to change.
<i>fil</i>	The new filter to be stored.

6.33.4.4 `clean()`

```
void Settings::clean ( )
```

Function to clean all settings: number types are set to 0, string are set to "", Tuples are set to `Tuple<>()` and [Filter](#) are set to all 0s.

6.33.4.5 `cleanAndRead()`

```
void Settings::cleanAndRead (
    string _path = "data/settings.xml" )
```

Function to clean all settings and then read from file. Default is data/settings.xml.

6.33.4.6 `getTemplates()` [1/3]

```
Tuple< string > Settings::getTemplates (
    int id = -1 )
```

Function to return the path of a template. If id is negative all templates are returned.

Function to return the path of a template. If id is not specified all templates are returned.

Parameters

<i>id</i>	The positions in this.templates of the template to be retrieved
-----------	---

Returns

A [Tuple](#) containing the paths of the templates.

6.33.4.7 getTemplates() [2/3]

```
string Settings::getTemplates (
    string _template )
```

A function to return the path of a given template.

Parameters

<i>_templateName</i>	The name of the template to check in the Tuple .
----------------------	--

Returns

The path to the template if it is found, an empty string otherwise.

6.33.4.8 getTemplates() [3/3]

```
Tuple< string > Settings::getTemplates (
    Tuple< string > _templates )
```

A function to return the paths of a given [Tuple](#) of templates.

Parameters

<i>_templates</i>	A Tuple containing the names of the templates to check in the Tuple .
-------------------	---

Returns

The paths to the templates if they are found, an empty [Tuple](#) otherwise.

6.33.4.9 maps() [1/4]

```
Tuple< string > Settings::maps (
    Tuple< int > ids = Tuple<int>() )
```

Function to return the paths of maps. If ids are not specified all maps are returned.

Parameters

<i>ids</i>	A Tuple containing the ids (that is the positions in this.mapsNames) of the maps to be retrieved.
------------	---

Returns

A [Tuple](#) containing the paths of the maps.

6.33.4.10 maps() [2/4]

```
Tuple< string > Settings::maps (
    int id = -1 )
```

Function to return the path of a map. If id is negative all maps are returned.

Function to return the path of a map. If id is not specified all maps are returned.

Parameters

<i>id</i>	The positions in this.mapsNames of the map to be retrieved
-----------	--

Returns

A [Tuple](#) containing the paths of the maps.

Parameters

<i>id</i>	A the positions in this.mapsNames of the map to be retrieved
-----------	--

Returns

A [Tuple](#) containing the paths of the maps.

6.33.4.11 maps() [3/4]

```
string Settings::maps (
    string _mapName )
```

A function to return the path of a given map.

Parameters

<i>_mapName</i>	The name of the map to check in the Tuple .
-----------------	---

Returns

The path to the map if the map is found, an empty string otherwise.

6.33.4.12 maps() [4 / 4]

```
Tuple< string > Settings::maps (
    Tuple< string > _mapNames )
```

A function to return the paths of a given [Tuple](#) of maps.

Parameters

<code>_mapNames</code>	A Tuple containing the names of the maps to check in the Tuple .
------------------------	--

Returns

The paths to the maps if they are found, an empty [Tuple](#) otherwise.

6.33.4.13 readFromFile()

```
void Settings::readFromFile (
    string _path = "data/settings.xml" )
```

Function to read from file. The data found is going to be added to the settings. Default file is data/settings.xml.

Parameters

<code>_path</code>	The path of file to read from.
--------------------	--------------------------------

6.33.4.14 save()

```
void Settings::save (
    string _mapsFolder = "data/map",
    string _templatesFolder = "data/num_template/",
    vector< string > _mapsNames = {},
    vector< string > _mapsUnNames = {},
    string _intrinsicCalibrationFile = "data/calib_config.xml",
    string _calibrationFile = "data/intrinsic_calibration.xml",
    Filter _blackMask = Filter(0, 0, 0, 179, 255, 70),
    Filter _redMask = Filter(15, 100, 140, 160, 255, 255),
    Filter _greenMask = Filter(54, 74, 25, 119, 255, 88),
    Filter _victimMask = Filter(0, 0, 0, 179, 255, 80),
```

```

Filter _blueMask = Filter(100, 100, 40, 140, 200, 170),
Filter _whiteMask = Filter(100, 100, 40, 140, 200, 170),
Filter _robotMask = Filter(100, 100, 40, 140, 200, 170),
int _kernelSide = 9,
string _convexHullFile = "data/convexHull.xml",
vector< string > _templates = {} )

```

Function to change values. The value are all set by default. This function does NOT read from or write to file.

Parameters

<i>mapsFolder</i>	A string containing the path for mapsFolder. No certainty is given about the form of this string
<i>_templatesFolder</i>	A String containing the path of the folder containing the number templates.
<i>_mapsNames</i>	A Tuple containing the names of the maps. These are not paths but just names.
<i>_mapsUnNames</i>	A Tuple containing the names of the undistorted maps. These are not paths but just names.
<i>_calibrationFile</i>	A string containing the path to the file containing the data for the calibration.
<i>_intrinsicCalibrationFile</i>	A string containing the path to the file containing the values of the matrix for the calibration.
<i>_blackMask</i>	Filter for black.
<i>_redMask</i>	Filter for red.
<i>_greenMask</i>	Filter for green.
<i>_victimMask</i>	Filter for the victims.
<i>_blueMask</i>	Filter for blue.
<i>_whiteMask</i>	Filter for white.
<i>_robotMask</i>	Filter for the triangle above the robot.
<i>_kernelSide</i>	
<i>_convexHullFile</i>	A String containing the path to file containing the points of the elements in the arena.
<i>_templates</i>	A Tuple containing the names of the templates. These are not paths but just names.

6.33.4.15 to_string()

```
stringstream Settings::to_string ( ) const [inline]
```

A function that creates a stringstream to print the values stored in settings.

Returns

A strinstream containing the settings values.

6.33.4.16 unMaps() [1/4]

```

Tuple< string > Settings::unMaps (
    Tuple< int > ids = Tuple<int>() )

```

Function to return the paths of undistorted maps. If ids are not specified all undistorted maps are returned.

Parameters

<i>ids</i>	A Tuple containing the ids (that is the positions in <code>this.mapsUnNames</code>) of the undistorted maps to be retrieved.
------------	---

Returns

A [Tuple](#) containing the paths of the undistorted maps.

6.33.4.17 `unMaps()` [2/4]

```
Tuple< string > Settings::unMaps (
    int id = -1 )
```

Function to return the path of an undistorted map. If id is negative all undistorted maps are returned.

Function to return the path of an undistorted map. If id is not specified all undistorted maps are returned.

Parameters

<i>id</i>	The positions in <code>this.mapsUnNames</code> of the undistorted map to be retrieved
-----------	---

Returns

A [Tuple](#) containing the paths of the undistorted maps.

Parameters

<i>id</i>	A the positions in <code>this.mapsUnNames</code> of the undistorted map to be retrieved
-----------	---

Returns

A [Tuple](#) containing the paths of the undistorted maps.

6.33.4.18 `unMaps()` [3/4]

```
string Settings::unMaps (
    string _unMapName )
```

A function to return the path of a given undistorted map.

Parameters

<i>_unMapName</i>	The name of the undistorted map to check in the Tuple .
-------------------	---

Returns

The path to the undistorted map if it is found, an empty string otherwise.

6.33.4.19 unMaps() [4/4]

```
Tuple< string > Settings::unMaps (
    Tuple< string > _unMapNames )
```

A function to return the paths of a given [Tuple](#) of undistorted maps.

Parameters

<code>_unMapNames</code>	A Tuple containing the names of the undistorted maps to check in the Tuple .
--------------------------	--

Returns

The paths to the undistorted maps if they are found, an empty [Tuple](#) otherwise.

6.33.4.20 writeToFile()

```
void Settings::writeToFile (
    string _path = "data/settings.xml" )
```

Function to write settings to file. Default is data/settings.xml.

Parameters

<code>_path</code>	The path of the file to write to.
--------------------	-----------------------------------

6.33.5 Friends And Related Function Documentation**6.33.5.1 operator<<**

```
ostream& operator<< (
    ostream & out,
    const Settings & data ) [friend]
```

This function overload the << operator so to print with `std::cout`.

Parameters

<i>in</i>	<i>out</i>	The out stream.
<i>in</i>	<i>datThe</i>	settings to print.

Returns

An output stream to be printed.

6.33.6 Member Data Documentation**6.33.6.1 blackMask**

`Filter Settings::blackMask`

`Filter` for black.

6.33.6.2 blueMask

`Filter Settings::blueMask`

`Filter` for blue.

6.33.6.3 calibrationFile

`string Settings::calibrationFile`

A string containing the path to the file containing the data for the calibration.

6.33.6.4 convexHullFile

`string Settings::convexHullFile`

AString containing the path to file containing the points of the elements in the arena.

6.33.6.5 greenMask

`Filter Settings::greenMask`

`Filter` for green.

6.33.6.6 intrinsicCalibrationFile

`string Settings::intrinsicCalibrationFile`

A string containing the path to the file containing the values of the matrix for the calibration.

6.33.6.7 kernelSide

`int Settings::kernelSide`

6.33.6.8 mapsFolder

`string Settings::mapsFolder`

A string containing the path for mapsFolder. No certainty is given about the form of this string.

6.33.6.9 mapsNames

`Tuple<string> Settings::mapsNames`

A `Tuple` containing the names of the maps. These are not paths but just names.

6.33.6.10 mapsUnNames

`Tuple<string> Settings::mapsUnNames`

A `Tuple` containing the names of the undistorted maps. These are not paths but just names.

6.33.6.11 redMask

`Filter Settings::redMask`

`Filter` for red.

6.33.6.12 robotMask

`Filter Settings::robotMask`

`Filter` for the triangle above the robot.

6.33.6.13 templates

`Tuple<string> Settings::templates`

A `Tuple` containing the names of the templates. These are not paths but just names.

6.33.6.14 templatesFolder

`string Settings::templatesFolder`

A String containing the path of the folder containing the number templates.

6.33.6.15 victimMask

`Filter Settings::victimMask`

`Filter` for the victims.

6.33.6.16 whiteMask

`Filter Settings::whiteMask`

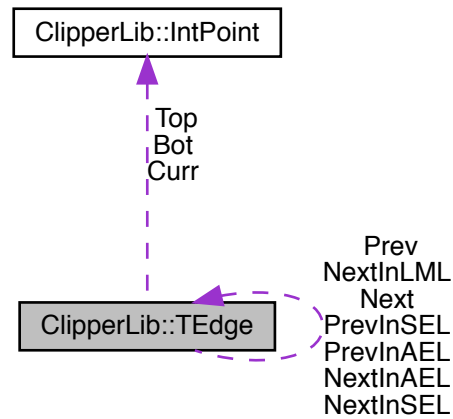
`Filter` for white.

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

- `src/include/settings.hh`
- `src/settings.cc`

6.34 ClipperLib::TEdge Struct Reference

Collaboration diagram for ClipperLib::TEdge:



Public Attributes

- [IntPoint Bot](#)
- [IntPoint Curr](#)
- [IntPoint Top](#)
- [double Dx](#)
- [PolyType PolyTyp](#)
- [EdgeSide Side](#)
- [int WindDelta](#)
- [int WindCnt](#)
- [int WindCnt2](#)
- [int OutIdx](#)
- [TEdge * Next](#)
- [TEdge * Prev](#)
- [TEdge * NextInLML](#)
- [TEdge * NextInAEL](#)
- [TEdge * PrevInAEL](#)
- [TEdge * NextInSEL](#)
- [TEdge * PrevInSEL](#)

6.34.1 Member Data Documentation

6.34.1.1 Bot

`IntPoint` ClipperLib::TEdge::Bot

6.34.1.2 Curr

`IntPoint` ClipperLib::TEdge::Curr

6.34.1.3 Dx

`double` ClipperLib::TEdge::Dx

6.34.1.4 Next

`TEdge*` ClipperLib::TEdge::Next

6.34.1.5 NextInAEL

`TEdge*` ClipperLib::TEdge::NextInAEL

6.34.1.6 NextInLML

`TEdge*` ClipperLib::TEdge::NextInLML

6.34.1.7 NextInSEL

`TEdge*` ClipperLib::TEdge::NextInSEL

6.34.1.8 OutIdx

`int` ClipperLib::TEdge::OutIdx

6.34.1.9 PolyTyp

`PolyType` ClipperLib::TEdge::PolyTyp

6.34.1.10 Prev

`TEdge*` ClipperLib::TEdge::Prev

6.34.1.11 PrevInAEL

`TEdge*` ClipperLib::TEdge::PrevInAEL

6.34.1.12 PrevInSEL

`TEdge*` ClipperLib::TEdge::PrevInSEL

6.34.1.13 Side

`EdgeSide` ClipperLib::TEdge::Side

6.34.1.14 Top

`IntPoint` ClipperLib::TEdge::Top

6.34.1.15 WindCnt

`int` ClipperLib::TEdge::WindCnt

6.34.1.16 WindCnt2

`int` ClipperLib::TEdge::WindCnt2

6.34.1.17 WindDelta

```
int ClipperLib::TEdge::WindDelta
```

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

- [src/clipper.cc](#)

6.35 Tuple< T > Class Template Reference

```
#include <maths.hh>
```

Public Member Functions

- [Tuple](#) ()
Default constructor.
- [Tuple](#) (int _n,...)
Constructors that takes the number of objectes to be stored, the objects and then stores them. For compatibility problem we strongly suggest to use this constructor only with standard types or types that can be promotted to one of the standard ones. For any other type we suggest to use an empty constructor and then use the [add\(\)](#) function.
- [int](#) size () const
- [T](#) get (const [int](#) _n) const
Gets the n-th element.
- [int](#) find (T _el)
- void [add](#) (const T _new)
Adds a value at the end of the list.
- void [addIfNot](#) (T _el)
- [int](#) remove (const T pos)
Removes a value from the list.
- void [eraseAll](#) ()
Removes all values from the [Tuple](#).
- [int](#) set (const [int](#) pos, const T _new)
Set a value in a certain position, or adds the element if the position equals the number of elements.
- void [ahead](#) (const T _new)
- bool [equal](#) (Tuple< T > _t)
- bool [operator==](#) (Tuple< T > _t)
- Tuple< T > [sum](#) (Tuple< T > t)
- Tuple< T > [sum](#) (T inc)
- Tuple< T > [operator+](#) (T inc)
- Tuple< T > & [operator+=](#) (T inc)
- Tuple< T > [mul](#) (Tuple< T > t)
- Tuple< T > [mul](#) (T inc)
- Tuple< T > [operator *](#) (T inc)
- Tuple< T > & [operator *=](#) (T inc)
- template<class T1 >
double [EuDistance](#) (const Tuple< T1 > B)
*Function that compute the Euclidean Distance between two tuples. They must have the same number of elements.
|tparam T1 The type of the elements in the second [Tuple](#).*
- template<class T1 >
double [MaDistance](#) (const Tuple< T1 > B)

Function that compute the Manhattan Distance between two tuples. They must have the same number of elements.
 $\text{\textbackslashtparam T1}$ The type of the elements in the second [Tuple](#).

- `template<class T1 >`
`double distance (const Tuple< T1 > B, const DISTANCE_TYPE dist=EUCLIDEAN)`
Wrapper to compute different distances. They must have the same number of elements. $\text{\textbackslashtparam T1}$ The type of the elements in the second [Tuple](#).
- `stringstream to_string (string _prefix="") const`
- `string to_std_string () const`
- `operator std::string () const`
- `template<class T1 >`
`operator vector< T1 > () const`
Overload of cast to vector.
- `int & operator[] (int index)`
Overloading [] operator to access elements in array style.
- `tuplelter begin ()`
Iterator.
- `tupleConstlter begin () const`
Const iterator.
- `tuplelter end ()`
Iterator.
- `tupleConstlter end () const`
Const iterator.

Friends

- `ostream & operator<< (ostream &out, const Tuple< T > &data)`
Overload of operator << to output the content of the tuple.

6.35.1 Detailed Description

```
template<class T>
class Tuple< T >
```

Brief This class allows the definition and storage of tuples of different dimensions. Functions to compute distance between tuples are also available.

Template Parameters

<i>T</i>	The type of elements to be stored.
----------	------------------------------------

6.35.2 Constructor & Destructor Documentation

6.35.2.1 Tuple() [1/2]

```
template<class T>
Tuple< T >::Tuple ( ) [inline]
```

Default constructor.

6.35.2.2 Tuple() [2/2]

```
template<class T>
Tuple< T >::Tuple (
    int _n,
    ... ) [inline]
```

Constructors that takes the number of objects to be stored, the objects and then stores them. For compatibility problem we strongly suggest to use this constructor only with standard types or types that can be promoted to one of the standard ones. For any other type we suggest to use an empty constructor and then use the `add()` function.

Parameters

in	\leftrightarrow \leftarrow n	Number of objects to store.
in	...	Objects to store.

6.35.3 Member Function Documentation

6.35.3.1 add()

```
template<class T>
void Tuple< T >::add (
    const T _new ) [inline]
```

Adds a value at the end of the list.

Parameters

in	<code>_new</code>	The new value to be added.
----	-------------------	----------------------------

6.35.3.2 addIfNot()

```
template<class T>
void Tuple< T >::addIfNot (
    T _el ) [inline]
```

6.35.3.3 ahead()

```
template<class T>
void Tuple< T >::ahead (
    const T _new ) [inline]
```

6.35.3.4 begin() [1/2]

```
template<class T>
tupleIter Tuple< T >::begin ( ) [inline]
```

Iterator.

Returns

the elements.begin() iterator.

6.35.3.5 begin() [2/2]

```
template<class T>
tupleConstIter Tuple< T >::begin ( ) const [inline]
```

Const iterator.

Returns

the elements.begin() iterator.

6.35.3.6 distance()

```
template<class T>
template<class T1 >
double Tuple< T >::distance (
    const Tuple< T1 > B,
    const DISTANCE_TYPE dist = EUCLIDEAN ) [inline]
```

Wrapper to compute different distances. They must have the same number of elements. \tparan T1 The type of the elements in the second Tuple.

Parameters

in	<i>B</i>	The second Tuple to use for computing the distance.
in	<i>dist</i>	The type of distance to be computed.

Returns

The distance between the two [Tuple](#).

6.35.3.7 end() [1/2]

```
template<class T>
tupleIter Tuple< T >::end ( ) [inline]
```

Iterator.

Returns

the `elements.end()` iterator.

6.35.3.8 end() [2/2]

```
template<class T>
tupleConstIter Tuple< T >::end ( ) const [inline]
```

Const iterator.

Returns

the `elements.begin()` iterator.

6.35.3.9 equal()

```
template<class T>
bool Tuple< T >::equal (
    Tuple< T > _t ) [inline]
```

6.35.3.10 eraseAll()

```
template<class T>
void Tuple< T >::eraseAll ( ) [inline]
```

Removes all values from the [Tuple](#).

6.35.3.11 EuDistance()

```
template<class T>
template<class T1 >
double Tuple< T >::EuDistance (
    const Tuple< T1 > B ) [inline]
```

Function that compute the Euclidean Distance between two tuples. They must have the same number of elements.
`\tparam T1` The type of the elements in the second [Tuple](#).

Parameters

in	<i>B</i>	the second Tuple to use for computing the distance.
----	----------	---

Returns

The Euclidean distance between the two [Tuple](#).

6.35.3.12 find()

```
template<class T>
int Tuple< T >::find (
    T _el ) [inline]
```

6.35.3.13 get()

```
template<class T>
T Tuple< T >::get (
    const int _n ) const [inline]
```

Gets the n-th element.

Parameters

in	\leftarrow	The position of the element to retrieve.
	\leftarrow <i>n</i>	

Returns

The element in the n-th position or an empty constructor if *_n* is greater than n or less than 0.

6.35.3.14 MaDistance()

```
template<class T>
template<class T1 >
double Tuple< T >::MaDistance (
    const Tuple< T1 > B ) [inline]
```

Function that compute the Manhattan Distance between two tuples. They must have the same number of elements.
\tparam T1 The type of the elements in the second [Tuple](#).

Parameters

<code>in</code>	<code>B</code>	the second Tuple to use for computing the distance.
-----------------	----------------	---

Returns

The Manhattan distance between the two [Tuple](#).

6.35.3.15 `mul()` [1/2]

```
template<class T>
Tuple<T> Tuple< T >::mul (
    Tuple< T > t ) [inline]
```

6.35.3.16 `mul()` [2/2]

```
template<class T>
Tuple<T> Tuple< T >::mul (
    T inc ) [inline]
```

6.35.3.17 `operator*()`

```
template<class T>
Tuple<T> Tuple< T >::operator * (
    T inc ) [inline]
```

6.35.3.18 `operator*=()`

```
template<class T>
Tuple<T>& Tuple< T >::operator *=(
    T inc ) [inline]
```

6.35.3.19 `operator std::string()`

```
template<class T>
Tuple< T >::operator std::string ( ) const [inline]
```

6.35.3.20 operator vector< T1 >()

```
template<class T>
template<class T1 >
Tuple< T >::operator vector< T1 > ( ) const [inline]
```

Overload of cast to vector.

Returns

A vector containing the values of elements.

6.35.3.21 operator+()

```
template<class T>
Tuple<T> Tuple< T >::operator+ (
    T inc ) [inline]
```

6.35.3.22 operator+=(())

```
template<class T>
Tuple<T>& Tuple< T >::operator+= (
    T inc ) [inline]
```

6.35.3.23 operator==(())

```
template<class T>
bool Tuple< T >::operator== (
    Tuple< T > _t ) [inline]
```

6.35.3.24 operator[]()

```
template<class T>
int& Tuple< T >::operator[] (
    int index ) [inline]
```

Overloading [] operator to access elements in array style.

Parameters

in	index	Id of value to get.
----	-------	---------------------

Returns

Value at id position.

6.35.3.25 remove()

```
template<class T>
int Tuple< T >::remove (
    const T pos ) [inline]
```

Removes a value from the list.

Parameters

in	<i>pos</i>	The position of the value to be removed.
----	------------	--

Returns

1 if everything went fine, 0 otherwise.

6.35.3.26 set()

```
template<class T>
int Tuple< T >::set (
    const int pos,
    const T _new ) [inline]
```

Set a value in a certain position, or adds the element if the position equals the number of elements.

Parameters

in	<i>pos</i>	Must be in $[0, n - 1]$. If $pos = n$ then the element is added at the end of the vector.
in	<i>_new</i>	The new element to be set.

Returns

1 if everything went right, 0 if the position was greater than n or less the 0.

6.35.3.27 size()

```
template<class T>
int Tuple< T >::size ( ) const [inline]
```

Returns

The number of stored elements. -1 if the [Tuple](#) has a different number of elements.

6.35.3.28 sum() [1/2]

```
template<class T>
Tuple<T> Tuple< T >::sum (
    Tuple< T > t ) [inline]
```

6.35.3.29 sum() [2/2]

```
template<class T>
Tuple<T> Tuple< T >::sum (
    T inc ) [inline]
```

6.35.3.30 to_std_string()

```
template<class T>
string Tuple< T >::to_std_string ( ) const [inline]
```

6.35.3.31 to_string()

```
template<class T>
stringstream Tuple< T >::to_string (
    string _prefix = "" ) const [inline]
```

This function create a stringstream object containing the values of the [Tuple](#).

Returns

A string stream.

6.35.4 Friends And Related Function Documentation**6.35.4.1 operator<<**

```
template<class T>
ostream& operator<< (
    ostream & out,
    const Tuple< T > & data ) [friend]
```

Overload of operator << to output the content of the tuple.

Parameters

<i>in</i>	<i>out</i>	The output stream.
<i>in</i>	<i>data</i>	The Tuple to print.

Returns

An output stream to be printed.

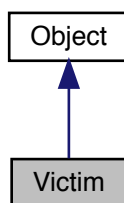
The documentation for this class was generated from the following file:

- [src/include/math.h](#)

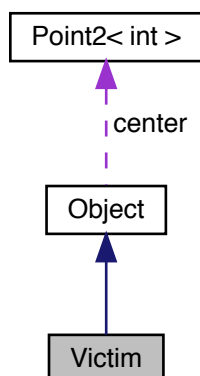
6.36 Victim Class Reference

```
#include <objects.hh>
```

Inheritance diagram for Victim:



Collaboration diagram for Victim:



Public Member Functions

- [Victim](#) (vector< [Point2](#)< [int](#) > > &vp, [int](#) _value)
Constructor of the victim class and automatically compute center and radius.
- string [toString](#) ()
Generate a string that describe the victim.
- void [print](#) ()
Print the describing string of the victim.
- [int](#) [getValue](#) ()
- void [setValue](#) ([int](#) v)

Protected Attributes

- [int](#) [value](#)

6.36.1 Constructor & Destructor Documentation

6.36.1.1 Victim()

```
Victim::Victim (
    vector< Point2< int > > & vp,
    int _value )
```

Constructor of the victim class and automatically compute center and radius.

Parameters

in	vp	Vector of points that is the convex hull of the victim.
in	_value	The representative number of the victim.

Returns

Return the created victim.

6.36.2 Member Function Documentation

6.36.2.1 getValue()

```
int Victim::getValue ( ) [inline]
```

6.36.2.2 print()

```
void Victim::print ( )
```

Print the describing string of the victim.

6.36.2.3 setValue()

```
void Victim::setValue (
    int v ) [inline]
```

6.36.2.4 toString()

```
string Victim::toString ( )
```

Generate a string that describe the victim.

Returns

The generated string.

6.36.3 Member Data Documentation

6.36.3.1 value

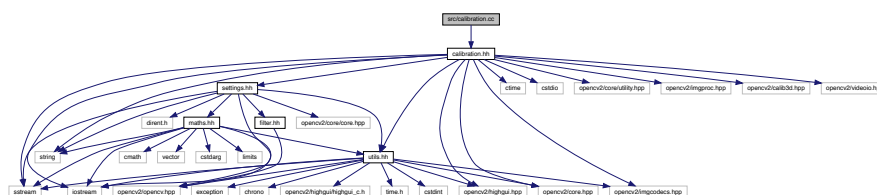
```
int Victim::value [protected]
```

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

- [src/include/objects.hh](#)
- [src/objects.cc](#)

File Documentation

```
#include "calibration.hh"
Include dependency graph for calibration.cc:
```



- `int calibration` (string inputFile)
Function to run the complete calibration.
- static void `read` (const FileNode &node, `CalSettings` &x, const `CalSettings` &default_value)
Reads `CalSettings` from file. If there is none then initiate a new `CalSettings`.
- static double `computeReprojectionErrors` (const vector< vector< Point3f > > &objectPoints, const vector< vector< Point2f > > &imagePoints, const vector< Mat > &rvecs, const vector< Mat > &tvecs, const Mat &cameraMatrix, const Mat &distCoeffs, vector< float > &perViewErrors, bool fisheye)
Compute the errors of the projection.
- void `calcBoardCornerPositions` (Size boardSize, float squareSize, vector< Point3f > &corners)
This function compute the position of the upper corners of every cell.
- static bool `runCalibration` (`CalSettings` &s, Size &imageSize, Mat &cameraMatrix, Mat &distCoeffs, vector< vector< Point2f > > &imagePoints, vector< Mat > &rvecs, vector< Mat > &tvecs, vector< float > &reprojErrs, double &totalAvgErr)
This function run the calibration creating the matrixed for the camera and the distortion coefficients.
- static void `saveCameraParams` (const `CalSettings` &s, const Size &imageSize, const Mat &cameraMatrix, const Mat &distCoeffs, const vector< Mat > &rvecs, const vector< Mat > &tvecs, const vector< float > &reprojErrs, const vector< vector< Point2f > > &imagePoints, const double totalAvgErr)
Function to save the computed parameters to a file.
- bool `runCalibrationAndSave` (`CalSettings` &s, Size imageSize, Mat &cameraMatrix, Mat &distCoeffs, vector< vector< Point2f > > &imagePoints)
Reads `CalSettings` from file. If there is none then initiate a new `CalSettings`.

7.1.1 Function Documentation

7.1.1.1 calcBoardCornerPositions()

```
void calcBoardCornerPositions (
    Size boardSize,
    float squareSize,
    vector< Point3f > & corners )
```

This function compute the position of the upper corners of every cell.

Parameters

in	<i>boardSiz</i>	The dimension of the chess board.
in	<i>squareSize</i>	The dimension of the edge of a cell.
out	<i>corners</i>	A vector of Point3fs which equals to the corners of the cells.

7.1.1.2 calibration()

```
int calibration (
    string inputFile )
```

Function to run the complete calibration.

Parameters

in	<i>inputFile</i>	Name of the setting.xml file. It's set to default to default.xml
----	------------------	--

Returns

- 2 if the [CalSettings](#) file could be load but the input was not well-formed
- 1 if the [CalSettings](#) file could not be opened.
- 0 if everything went fine.

7.1.1.3 computeReprojectionErrors()

```
static double computeReprojectionErrors (
    const vector< vector< Point3f > > & objectPoints,
    const vector< vector< Point2f > > & imagePoints,
    const vector< Mat > & rvecs,
    const vector< Mat > & tvecs,
    const Mat & cameraMatrix,
```

```

    const Mat & distCoeffs,
    vector< float > & perViewErrors,
    bool fisheye ) [static]

```

Compute the errors of the projection.

Parameters

in	<i>objectPoints</i>	The real image points which will be projected
in	<i>rvecs</i>	Input vector of rotation vectors estimated for each pattern view.
in	<i>tvecs</i>	Input vector of translation vectors estimated for each pattern view.
in	<i>cameraMatrix</i>	The matrix containing the parameters for the camera
in	<i>distCoeffs</i>	The matrix containing the distortion coefficients.
in	<i>fisheye</i>	A variable which says if a fish eye correction should be applied or no.
out	<i>perViewErrors</i>	A vector containing the error for each image.
out	<i>imagePoints</i>	The projected points for each image.

Returns

The total error.

7.1.1.4 read()

```

static void read (
    const FileNode & node,
    CalSettings & x,
    const CalSettings & default_value ) [inline], [static]

```

Reads [CalSettings](#) from file. If there is none then initiate a new [CalSettings](#).

Parameters

in	<i>node</i>	node to consider for getting CalSettings ;
in	<i>x</i>	CalSettings to configure;
in	<i>default_value</i>	CalSettings default value. Setted to CalSettings() .

7.1.1.5 runCalibration()

```

static bool runCalibration (
    CalSettings & s,
    Size & imageSize,
    Mat & cameraMatrix,
    Mat & distCoeffs,
    vector< vector< Point2f > > & imagePoints,
    vector< Mat > & rvecs,

```

```
vector< Mat > & tvecs,
vector< float > & reprojErrs,
double & totalAvgErr ) [static]
```

This function run the calibration creating the matrixed for the camera and the distorsion coefficients.

Parameters

in	<i>s</i>	The CalSettings read from the file and memorized.
in	<i>imageSize</i>	The size of the image used in <code>calibrateCamera()</code> to initialize the camera matrix.
in	<i>imagePoints</i>	The projected points for each image.
in	<i>reprojErrs</i>	The re-projection error, that is a geometric error corresponding to the image distance between a projected point and a measured one.
out	<i>cameraMatrix</i>	The matrix of the camera parameters
out	<i>distCoeffs</i>	The matrix of the distorsion coefficients.
out	<i>rvecs</i>	Output vector of rotation vectors estimated for each pattern view.
out	<i>tvecs</i>	Output vector of translation vectors estimated for each pattern view.
out	<i>totalAvgErr</i>	The total avarage error given from distorsion.

Returns

false if one or more elements in the `cameraMatrix` and `distCoeffs` are invalid.
true if all the elements are valid.

7.1.1.6 runCalibrationAndSave()

```
bool runCalibrationAndSave (
    CalSettings & s,
    Size imageSize,
    Mat & cameraMatrix,
    Mat & distCoeffs,
    vector< vector< Point2f > > imagePoints )
```

Reads [CalSettings](#) from file. If there is none then initiate a new [CalSettings](#).

Parameters

in	<i>s</i>	The CalSettings being used during the execution.
in	<i>imageSize</i>	The dimensions of the images.
in	<i>imagePoints</i>	The projected points for each image.
out	<i>cameraMatrix</i>	The matrix which is used to store the values for the camera parameters.
out	<i>distCoeffs</i>	The matrix which is used to store the distortion coefficients.

Returns

true if the calibration succeeded.
false otherwise.

7.1.1.7 saveCameraParams()

```
static void saveCameraParams (
    const CalSettings & s,
    const Size & imageSize,
    const Mat & cameraMatrix,
    const Mat & distCoeffs,
    const vector< Mat > & rvecs,
    const vector< Mat > & tvecs,
    const vector< float > & reprojErrs,
    const vector< vector< Point2f > > & imagePoints,
    const double totalAvgErr ) [static]
```

Function to save the computed parameters to a file.

Parameters

in	<i>s</i>	Use the <code>CalSettings</code> got at the beginning for information as the output file name, image and board size.
in	<i>imageSize</i>	The size of the image.
in	<i>cameraMatrix</i>	The camera matrix.
in	<i>distCoeffs</i>	The distortion coefficient matrix.
	<i>[int]</i>	rvecs Vector of rotation vectors estimated for each pattern view.
in	<i>tvecs</i>	Vector of translation vectors estimated for each pattern view.
in	<i>reprojErrs</i>	The re-projection error, that is a geometric error corresponding to the image distance between a projected point and a measured one.
in	<i>imagePoints</i>	The projected points for each image.
in	<i>totalAvgErr</i>	The total average error given from distortion.

Open file for writing

Stores time of calibration

Store infos about the images

7.2 src/camera_capture.cc File Reference

```
#include <iostream>
#include <utils.hh>
#include <cstring>
#include <camera_capture.hh>
Include dependency graph for camera_capture.cc:
```



Macros

- #define `SDEBUG(X)` {}

7.2.1 Macro Definition Documentation

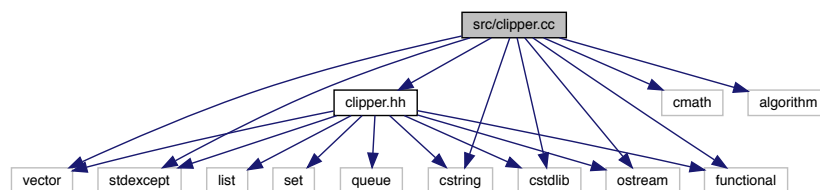
7.2.1.1 SDEBUG

```
#define SDEBUG(  
    X ) {}
```

7.3 src/clipper.cc File Reference

```
#include "clipper.hh"  
#include <cmath>  
#include <vector>  
#include <algorithm>  
#include <stdexcept>  
#include <cstring>  
#include <cstdlib>  
#include <ostream>  
#include <functional>
```

Include dependency graph for clipper.cc:



Classes

- struct [ClipperLib::TEdge](#)
- struct [ClipperLib::IntersectNode](#)
- struct [ClipperLib::LocalMinimum](#)
- struct [ClipperLib::OutRec](#)
- struct [ClipperLib::OutPt](#)
- struct [ClipperLib::Join](#)
- struct [ClipperLib::LocMinSorter](#)
- class [ClipperLib::Int128](#)

Namespaces

- [ClipperLib](#)

Macros

- #define `HORIZONTAL` (-1.0E+40)
- #define `TOLERANCE` (1.0e-20)
- #define `NEAR_ZERO`(val) (((val) > -`TOLERANCE`) && ((val) < `TOLERANCE`))

Enumerations

- enum `ClipperLib::Direction` { `ClipperLib::dRightToLeft`, `ClipperLib::dLeftToRight` }
- enum `ClipperLib::NodeType` { `ClipperLib::ntAny`, `ClipperLib::ntOpen`, `ClipperLib::ntClosed` }

Functions

- `clnt ClipperLib::Round` (double val)
- `clnt ClipperLib::Abs` (clnt val)
- `Int128 ClipperLib::Int128Mul` (long64 lhs, long64 rhs)
- `bool ClipperLib::Orientation` (const Path &poly)
- `double ClipperLib::Area` (const Path &poly)
- `double ClipperLib::Area` (const OutPt *op)
- `double ClipperLib::Area` (const OutRec &outRec)
- `bool ClipperLib::PointIsVertex` (const IntPoint &Pt, OutPt *pp)
- `int ClipperLib::PointInPolygon` (const IntPoint &pt, const Path &path)
- `int ClipperLib::PointInPolygon` (const IntPoint &pt, OutPt *op)
- `bool ClipperLib::Poly2ContainsPoly1` (OutPt *OutPt1, OutPt *OutPt2)
- `bool ClipperLib::SlopesEqual` (const TEdge &e1, const TEdge &e2, bool UseFullInt64Range)
- `bool ClipperLib::SlopesEqual` (const IntPoint pt1, const IntPoint pt2, const IntPoint pt3, bool UseFullInt64Range)
- `bool ClipperLib::SlopesEqual` (const IntPoint pt1, const IntPoint pt2, const IntPoint pt3, const IntPoint pt4, bool UseFullInt64Range)
- `bool ClipperLib::IsHorizontal` (TEdge &e)
- `double ClipperLib::GetDx` (const IntPoint pt1, const IntPoint pt2)
- `void ClipperLib::SetDx` (TEdge &e)
- `void ClipperLib::SwapSides` (TEdge &Edge1, TEdge &Edge2)
- `void ClipperLib::SwapPolyIndexes` (TEdge &Edge1, TEdge &Edge2)
- `clnt ClipperLib::TopX` (TEdge &edge, const clnt currentY)
- `void ClipperLib::IntersectPoint` (TEdge &Edge1, TEdge &Edge2, IntPoint &ip)
- `void ClipperLib::ReversePolyPtLinks` (OutPt *pp)
- `void ClipperLib::DisposeOutPts` (OutPt *&pp)
- `void ClipperLib::InitEdge` (TEdge *e, TEdge *eNext, TEdge *ePrev, const IntPoint &Pt)
- `void ClipperLib::InitEdge2` (TEdge &e, PolyType Pt)
- `TEdge * ClipperLib::RemoveEdge` (TEdge *e)
- `void ClipperLib::ReverseHorizontal` (TEdge &e)
- `void ClipperLib::SwapPoints` (IntPoint &pt1, IntPoint &pt2)
- `bool ClipperLib::GetOverlapSegment` (IntPoint pt1a, IntPoint pt1b, IntPoint pt2a, IntPoint pt2b, IntPoint &pt1, IntPoint &pt2)
- `bool ClipperLib::FirstIsBottomPt` (const OutPt *btmPt1, const OutPt *btmPt2)
- `OutPt * ClipperLib::GetBottomPt` (OutPt *pp)
- `bool ClipperLib::Pt2IsBetweenPt1AndPt3` (const IntPoint pt1, const IntPoint pt2, const IntPoint pt3)
- `bool ClipperLib::HorzSegmentsOverlap` (clnt seg1a, clnt seg1b, clnt seg2a, clnt seg2b)
- `void ClipperLib::RangeTest` (const IntPoint &Pt, bool &useFullRange)
- `TEdge * ClipperLib::FindNextLocMin` (TEdge *E)
- `OutRec * ClipperLib::GetLowermostRec` (OutRec *outRec1, OutRec *outRec2)
- `bool ClipperLib::OutRec1RightOfOutRec2` (OutRec *outRec1, OutRec *outRec2)

- bool [ClipperLib::IsMinima](#) (TEdge *e)
- bool [ClipperLib::IsMaxima](#) (TEdge *e, const cInt Y)
- bool [ClipperLib::IsIntermediate](#) (TEdge *e, const cInt Y)
- TEdge * [ClipperLib::GetMaximaPair](#) (TEdge *e)
- TEdge * [ClipperLib::GetMaximaPairEx](#) (TEdge *e)
- TEdge * [ClipperLib::GetNextInAEL](#) (TEdge *e, Direction dir)
- void [ClipperLib::GetHorzDirection](#) (TEdge &HorzEdge, Direction &Dir, cInt &Left, cInt &Right)
- bool [ClipperLib::IntersectListSort](#) (IntersectNode *node1, IntersectNode *node2)
- bool [ClipperLib::EdgesAdjacent](#) (const IntersectNode &inode)
- int [ClipperLib::PointCount](#) (OutPt *Pts)
- void [ClipperLib::SwapIntersectNodes](#) (IntersectNode &int1, IntersectNode &int2)
- bool [ClipperLib::E2InsertsBeforeE1](#) (TEdge &e1, TEdge &e2)
- bool [ClipperLib::GetOverlap](#) (const cInt a1, const cInt a2, const cInt b1, const cInt b2, cInt &Left, cInt &Right)
- void [ClipperLib::UpdateOutPtIdxs](#) (OutRec &outrec)
- OutPt * [ClipperLib::DupOutPt](#) (OutPt *outPt, bool InsertAfter)
- bool [ClipperLib::JoinHorz](#) (OutPt *op1, OutPt *op1b, OutPt *op2, OutPt *op2b, const IntPoint Pt, bool DiscardLeft)
- static OutRec * [ClipperLib::ParseFirstLeft](#) (OutRec *FirstLeft)
- DoublePoint [ClipperLib::GetUnitNormal](#) (const IntPoint &pt1, const IntPoint &pt2)
- void [ClipperLib::ReversePath](#) (Path &p)
- void [ClipperLib::ReversePaths](#) (Paths &p)
- void [ClipperLib::SimplifyPolygon](#) (const Path &in_poly, Paths &out_polys, PolyFillType fillType)
- void [ClipperLib::SimplifyPolygons](#) (const Paths &in_polys, Paths &out_polys, PolyFillType fillType)
- void [ClipperLib::SimplifyPolygons](#) (Paths &polys, PolyFillType fillType)
- double [ClipperLib::DistanceSqrD](#) (const IntPoint &pt1, const IntPoint &pt2)
- double [ClipperLib::DistanceFromLineSqrD](#) (const IntPoint &pt, const IntPoint &ln1, const IntPoint &ln2)
- bool [ClipperLib::SlopesNearCollinear](#) (const IntPoint &pt1, const IntPoint &pt2, const IntPoint &pt3, double distSqrD)
- bool [ClipperLib::PointsAreClose](#) (IntPoint pt1, IntPoint pt2, double distSqrD)
- OutPt * [ClipperLib::ExcludeOp](#) (OutPt *op)
- void [ClipperLib::CleanPolygon](#) (const Path &in_poly, Path &out_poly, double distance)
- void [ClipperLib::CleanPolygon](#) (Path &poly, double distance)
- void [ClipperLib::CleanPolygons](#) (const Paths &in_polys, Paths &out_polys, double distance)
- void [ClipperLib::CleanPolygons](#) (Paths &polys, double distance)
- void [ClipperLib::Minkowski](#) (const Path &poly, const Path &path, Paths &solution, bool isSum, bool isClosed)
- void [ClipperLib::MinkowskiSum](#) (const Path &pattern, const Path &path, Paths &solution, bool pathsIsClosed)
- void [ClipperLib::TranslatePath](#) (const Path &input, Path &output, const IntPoint delta)
- void [ClipperLib::MinkowskiSum](#) (const Path &pattern, const Paths &paths, Paths &solution, bool pathsIsClosed)
- void [ClipperLib::MinkowskiDiff](#) (const Path &poly1, const Path &poly2, Paths &solution)
- void [ClipperLib::AddPolyNodeToPaths](#) (const PolyNode &polynode, NodeType nodetype, Paths &paths)
- void [ClipperLib::PolyTreeToPaths](#) (const PolyTree &polytree, Paths &paths)
- void [ClipperLib::ClosedPathsFromPolyTree](#) (const PolyTree &polytree, Paths &paths)
- void [ClipperLib::OpenPathsFromPolyTree](#) (PolyTree &polytree, Paths &paths)
- std::ostream & [ClipperLib::operator<<](#) (std::ostream &s, const IntPoint &p)
- std::ostream & [ClipperLib::operator<<](#) (std::ostream &s, const Path &p)
- std::ostream & [ClipperLib::operator<<](#) (std::ostream &s, const Paths &p)

Variables

- static double const [ClipperLib::pi](#) = 3.141592653589793238
- static double const [ClipperLib::two_pi](#) = pi *2
- static double const [ClipperLib::def_arc_tolerance](#) = 0.25
- static int const [ClipperLib::Unassigned](#) = -1
- static int const [ClipperLib::Skip](#) = -2

7.3.1 Macro Definition Documentation

7.3.1.1 HORIZONTAL

```
#define HORIZONTAL (-1.0E+40)
```

7.3.1.2 NEAR_ZERO

```
#define NEAR_ZERO(  
    val ) (((val) > -TOLERANCE) && ((val) < TOLERANCE))
```

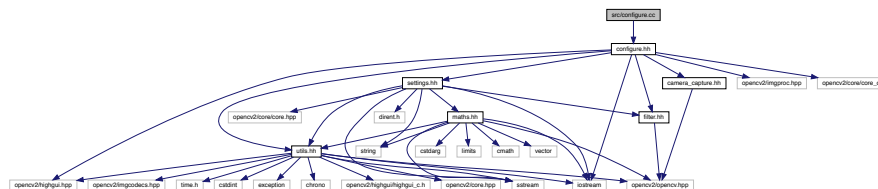
7.3.1.3 TOLERANCE

```
#define TOLERANCE (1.0e-20)
```

7.4 src/configure.cc File Reference

```
#include <configure.hh>
```

Include dependency graph for configure.cc:



Functions

- void [on_low_h_thresh_trackbar](#) (int, void *)
- void [on_high_h_thresh_trackbar](#) (int, void *)
- void [on_low_s_thresh_trackbar](#) (int, void *)
- void [on_high_s_thresh_trackbar](#) (int, void *)
- void [on_low_v_thresh_trackbar](#) (int, void *)
- void [on_high_v_thresh_trackbar](#) (int, void *)
- void [update_trackers](#) ()
- Mat [acquireImage](#) (const bool save)

It acquire a frame from the default camera of the pc.

- void [configure](#) (bool deploy, int img_id)

If DEPLOY is defined then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If DEPLOY is not defined then it takes a map from the folder set in [Settings](#) and ask for visual confirmation.

- bool [show_all_conditions](#) (const Mat &frame)

Variables

- `Filter filter = Filter(30, 30, 30, 100, 100, 100)`

7.4.1 Function Documentation

7.4.1.1 `acquireImage()`

```
Mat acquireImage (
    const bool save )
```

It acquire a frame from the default camera of the pc.

Parameters

<code>in</code>	<code>save</code>	If save, or not, the acquired image to a file.
-----------------	-------------------	--

Returns

The Mat of the acquired frame.

7.4.1.2 `configure()`

```
void configure (
    bool deploy,
    int img_id )
```

If DEPLOY is defined then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If DEPLOY is not defined then it takes a map from the folder set in [Settings](#) and ask for visual confirmation.

If deploy is true then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If deploy is false then it takes the `img_id`-th maps from the folder set in [Settings](#) and ask for visual confirmation.

7.4.1.3 `on_high_h_thresh_trackbar()`

```
void on_high_h_thresh_trackbar (
    int ,
    void * )
```

@function `on_high_h_thresh_trackbar`

7.4.1.4 on_high_s_thresh_trackbar()

```
void on_high_s_thresh_trackbar (
    int ,
    void * )
```

@function on_high_s_thresh_trackbar

7.4.1.5 on_high_v_thresh_trackbar()

```
void on_high_v_thresh_trackbar (
    int ,
    void * )
```

@function on_high_v_thresh_trackbar

7.4.1.6 on_low_h_thresh_trackbar()

```
void on_low_h_thresh_trackbar (
    int ,
    void * )
```

@function on_low_h_thresh_trackbar

7.4.1.7 on_low_s_thresh_trackbar()

```
void on_low_s_thresh_trackbar (
    int ,
    void * )
```

@function on_low_s_thresh_trackbar

7.4.1.8 on_low_v_thresh_trackbar()

```
void on_low_v_thresh_trackbar (
    int ,
    void * )
```

@function on_low_v_thresh_trackbar

7.4.1.9 show_all_conditions()

```
bool show_all_conditions (
    const Mat & frame )
```

Function to show a picture with various filters taken from [Settings](#). It then asks for visual confirmation.

Functions

- [int detection](#) ()
Loads some images and detects shapes according to different colors.
- void [computeConversionParameters](#) (Mat &transf)
- [Point2< int > localize](#) ()
- [Point2< int > localize](#) (const Mat &img)
- void [load_number_template](#) ()
Load some templates and save them in the global variable 'templates'.
- void [shape_detection](#) (const Mat &img, const [COLOR_TYPE](#) color)
Detect shapes inside the image according to the variable 'color'.
- void [erode_dilation](#) (Mat &img, const [COLOR_TYPE](#) color)
It apply some filtering function for isolate the subject and remove the noise.
- bool [_compare](#) (const pair< [int](#), [int](#) > &a, const pair< [int](#), [int](#) > &b)
- void [find_contours](#) (const Mat &img, const Mat &original, const [COLOR_TYPE](#) color)
Given an image, in black/white format, identify all the borders that delimit the shapes.
- void [save_convex_hull](#) (const vector< vector< [Point](#) >> &contours, const [COLOR_TYPE](#) color)
Given some vector save it in a xml file.
- [int number_recognition](#) (Rect blob, const Mat &base)
Detect a number on an image inside a region of interest.
- void [crop_number_section](#) (Mat &ROI)
Given an image identify the region of interest(ROI) and crop it out.

Variables

- vector< Mat > [templates](#)
- vector< [Point](#) > [robotShape](#)

7.5.1 Macro Definition Documentation

7.5.1.1 EPS_CURVE

```
#define EPS_CURVE 5
```

Given an image, in black/white format, identify all the borders that delimit the shapes.

Parameters

in	<i>img</i>	It is an image in HSV format at the base of the elaboration process.
out	<i>original</i>	It is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	It is the type of reference color.

7.5.1.2 MIN_AREA_SIZE

```
#define MIN_AREA_SIZE 1000
```

7.5.2 Function Documentation

7.5.2.1 _compare()

```
bool _compare (
    const pair< int, int > & a,
    const pair< int, int > & b )
```

7.5.2.2 computeConversionParameters()

```
void computeConversionParameters (
    Mat & transf )
```

7.5.2.3 crop_number_section()

```
void crop_number_section (
    Mat & ROI )
```

Given an image identify the region of interest(ROI) and crop it out.

Parameters

<i>in, out</i>	<i>ROI</i>	Is the image that the function will going to elaborate.
----------------	------------	---

7.5.2.4 detection()

```
int detection ( )
```

Loads some images and detects shapes according to different colors.

Returns

Return 0 if the function reach the end.

7.5.2.5 erode_dilation()

```
void erode_dilation (
    Mat & img,
    const COLOR_TYPE color )
```

It apply some filtering function for isolate the subject and remove the noise.

An example of the sub functions called are: GaussianBlur, Erosion, Dilation and Threshold.

Parameters

in, out	<i>img</i>	Is the image on which the function apply the filtering.
in	<i>color</i>	It is the type of reference color. According to the color the filtering functions apply can change in the type and in the order.

7.5.2.6 find_contours()

```
void find_contours (
    const Mat & img,
    const Mat & original,
    const COLOR_TYPE color )
```

Given an image, in black/white format, identify all the borders that delimit the shapes.

Parameters

in	<i>img</i>	Is an image in HSV format at the base of the elaboration process.
out	<i>original</i>	It is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	It is the type of reference color.

7.5.2.7 load_number_template()

```
void load_number_template ( )
```

Load some templates and save them in the global variable 'templates'.

7.5.2.8 localize() [1/2]

```
Point2<int> localize ( )
```

7.5.2.9 `localize()` [2/2]

```
Point2<int> localize (
    const Mat & img )
```

7.5.2.10 `number_recognition()`

```
int number_recognition (
    Rect blob,
    const Mat & base )
```

Detect a number on an image inside a region of interest.

Parameters

in	<i>blob</i>	Identify the region of interest inside the image 'base'.
in	<i>base</i>	Is the image where the function will going to search the number.

Returns

The number recognise, '-1' otherwise.

7.5.2.11 `save_convex_hull()`

```
void save_convex_hull (
    const vector< vector< Point >> & contours,
    const COLOR_TYPE color )
```

Given some vector save it in a xml file.

Parameters

in	<i>contours</i>	Is a vector that is saved in a xml file.
in	<i>color</i>	It is the type of reference color, according to which the function decide if saved ('color==GREEN') or not ('otherwise') the vector 'victims'.

7.5.2.12 `shape_detection()`

```
void shape_detection (
    const Mat & img,
    const COLOR_TYPE color )
```

Detect shapes inside the image according to the variable 'color'.

Parameters

in	<i>img</i>	Image on which the research will done.
in	<i>color</i>	It is the type of reference color. These color identify the possible spectrum that the function search on the image.

7.5.3 Variable Documentation

7.5.3.1 robotShape

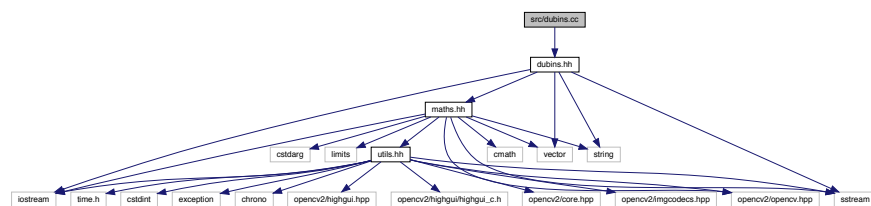
```
vector<Point> robotShape
```

7.5.3.2 templates

```
vector<Mat> templates
```

7.6 src/dubins.cc File Reference

```
#include "dubins.hh"
Include dependency graph for dubins.cc:
```



7.7 src/include/calibration.hh File Reference

Library for calibration.

```
#include <utils.hh>
#include <settings.hh>
#include <iostream>
#include <sstream>
#include <string>
#include <ctime>
#include <cstdio>
```


7.7.1 Detailed Description

Library for calibration.

7.7.2 Enumeration Type Documentation

7.7.2.1 anonymous enum

anonymous enum

Enumerator

DETECTION	
CAPTURING	
CALIBRATED	

7.7.3 Function Documentation

7.7.3.1 calibration()

```
int calibration (
    string inputFile )
```

Function to run the complete calibration.

Parameters

in	<i>inputFile</i>	Name of the setting.xml file. It's set to default to default.xml
----	------------------	--

Returns

- 2 if the [CalSettings](#) file could be load but the input was not well-formed
- 1 if the [CalSettings](#) file could not be opened.
- 0 if everything went fine.

7.7.3.2 runCalibrationAndSave()

```
bool runCalibrationAndSave (
    CalSettings & s,
```

```

    Size imageSize,
    Mat & cameraMatrix,
    Mat & distCoeffs,
    vector< vector< Point2f > > imagePoints )

```

Reads [CalSettings](#) from file. If there is none then initiate a new [CalSettings](#).

Parameters

in	<i>s</i>	The CalSettings being used during the execution.
in	<i>imageSize</i>	The dimensions of the images.
in	<i>imagePoints</i>	The projected points for each image.
out	<i>cameraMatrix</i>	The matrix which is used to store the values for the camera parameters.
out	<i>distCoeffs</i>	The matrix which is used to store the distortion coefficients.

Returns

```

    true if the calibration succeeded.
    false otherwise.

```

7.7.4 Variable Documentation

7.7.4.1 sett

```

Settings* sett

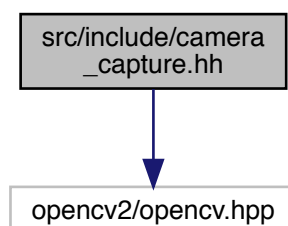
```

7.8 src/include/camera_capture.hh File Reference

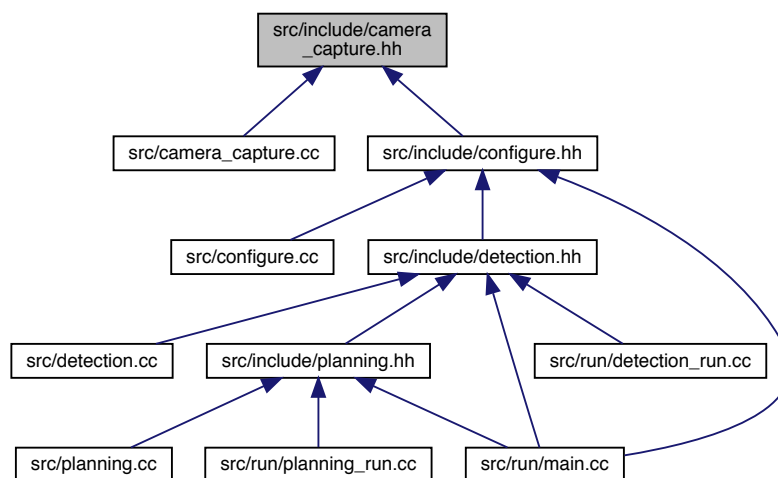
```

#include <opencv2/opencv.hpp>
Include dependency graph for camera_capture.hh:

```



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



Classes

- class [CameraCapture](#)
- struct [CameraCapture::input_options_t](#)

Structure for store the input option for the class [CameraCapture](#).

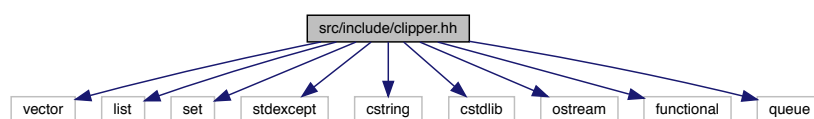
7.9 src/include/clipper.hh File Reference

```

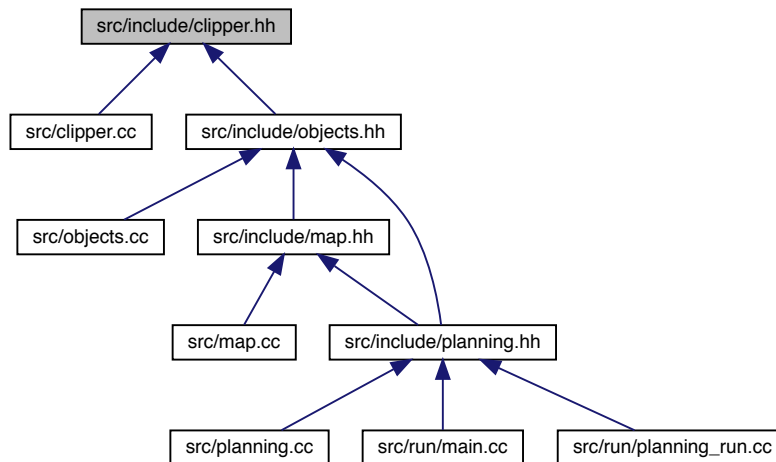
#include <vector>
#include <list>
#include <set>
#include <stdexcept>
#include <cstring>
#include <cstdlib>
#include <ostream>
#include <functional>
#include <queue>

```

Include dependency graph for clipper.hh:



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



Classes

- struct [ClipperLib::IntPoint](#)
- struct [ClipperLib::DoublePoint](#)
- class [ClipperLib::PolyNode](#)
- class [ClipperLib::PolyTree](#)
- struct [ClipperLib::IntRect](#)
- class [ClipperLib::ClipperBase](#)
- class [ClipperLib::Clipper](#)
- class [ClipperLib::ClipperOffset](#)
- class [ClipperLib::clipperException](#)

Namespaces

- [ClipperLib](#)

Macros

- `#define` [CLIPPER_VERSION](#) "6.4.2"
- `#define` [use_lines](#)

Typedefs

- typedef signed long long [ClipperLib::cInt](#)
- typedef signed long long [ClipperLib::long64](#)
- typedef unsigned long long [ClipperLib::ulong64](#)
- typedef std::vector< IntPoint > [ClipperLib::Path](#)
- typedef std::vector< Path > [ClipperLib::Paths](#)
- typedef std::vector< PolyNode * > [ClipperLib::PolyNodes](#)
- typedef std::vector< OutRec * > [ClipperLib::PolyOutList](#)
- typedef std::vector< TEdge * > [ClipperLib::EdgeList](#)
- typedef std::vector< Join * > [ClipperLib::JoinList](#)
- typedef std::vector< IntersectNode * > [ClipperLib::IntersectList](#)

Enumerations

- enum `ClipperLib::ClipType` { `ClipperLib::ctIntersection`, `ClipperLib::ctUnion`, `ClipperLib::ctDifference`, `ClipperLib::ctXor` }
- enum `ClipperLib::PolyType` { `ClipperLib::ptSubject`, `ClipperLib::ptClip` }
- enum `ClipperLib::PolyFillType` { `ClipperLib::pftEvenOdd`, `ClipperLib::pftNonZero`, `ClipperLib::pftPositive`, `ClipperLib::pftNegative` }
- enum `ClipperLib::InitOptions` { `ClipperLib::ioReverseSolution` = 1, `ClipperLib::ioStrictlySimple` = 2, `ClipperLib::ioPreserveCollinear` = 4 }
- enum `ClipperLib::JoinType` { `ClipperLib::jtSquare`, `ClipperLib::jtRound`, `ClipperLib::jtMiter` }
- enum `ClipperLib::EndType` { `ClipperLib::etClosedPolygon`, `ClipperLib::etClosedLine`, `ClipperLib::etOpenButt`, `ClipperLib::etOpenSquare`, `ClipperLib::etOpenRound` }
- enum `ClipperLib::EdgeSide` { `ClipperLib::esLeft` = 1, `ClipperLib::esRight` = 2 }

Functions

- `Path & ClipperLib::operator<<` (`Path &poly`, `const IntPoint &p`)
- `Paths & ClipperLib::operator<<` (`Paths &polys`, `const Path &p`)
- `std::ostream & ClipperLib::operator<<` (`std::ostream &s`, `const IntPoint &p`)
- `std::ostream & ClipperLib::operator<<` (`std::ostream &s`, `const Path &p`)
- `std::ostream & ClipperLib::operator<<` (`std::ostream &s`, `const Paths &p`)
- `bool ClipperLib::Orientation` (`const Path &poly`)
- `double ClipperLib::Area` (`const Path &poly`)
- `int ClipperLib::PointInPolygon` (`const IntPoint &pt`, `const Path &path`)
- `void ClipperLib::SimplifyPolygon` (`const Path &in_poly`, `Paths &out_polys`, `PolyFillType fillType`)
- `void ClipperLib::SimplifyPolygons` (`const Paths &in_polys`, `Paths &out_polys`, `PolyFillType fillType`)
- `void ClipperLib::SimplifyPolygons` (`Paths &polys`, `PolyFillType fillType`)
- `void ClipperLib::CleanPolygon` (`const Path &in_poly`, `Path &out_poly`, `double distance`)
- `void ClipperLib::CleanPolygon` (`Path &poly`, `double distance`)
- `void ClipperLib::CleanPolygons` (`const Paths &in_polys`, `Paths &out_polys`, `double distance`)
- `void ClipperLib::CleanPolygons` (`Paths &polys`, `double distance`)
- `void ClipperLib::MinkowskiSum` (`const Path &pattern`, `const Path &path`, `Paths &solution`, `bool pathsIsClosed`)
- `void ClipperLib::MinkowskiSum` (`const Path &pattern`, `const Paths &paths`, `Paths &solution`, `bool pathsIsClosed`)
- `void ClipperLib::MinkowskiDiff` (`const Path &poly1`, `const Path &poly2`, `Paths &solution`)
- `void ClipperLib::PolyTreeToPaths` (`const PolyTree &polytree`, `Paths &paths`)
- `void ClipperLib::ClosedPathsFromPolyTree` (`const PolyTree &polytree`, `Paths &paths`)
- `void ClipperLib::OpenPathsFromPolyTree` (`PolyTree &polytree`, `Paths &paths`)
- `void ClipperLib::ReversePath` (`Path &p`)
- `void ClipperLib::ReversePaths` (`Paths &p`)

Variables

- `static clnt const ClipperLib::loRange` = 0x3FFFFFFF
- `static clnt const ClipperLib::hiRange` = 0x3FFFFFFFFFFFFFFFLL

7.9.1 Macro Definition Documentation

Functions

- Mat [acquireImage](#) (const bool save=true)
It acquire a frame from the default camera of the pc.
- void [configure](#) (bool deploy=true, int img_id=0)
If deploy is true then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If deploy is false then it takes the img_id-th maps from the folder set in [Settings](#) and ask for visual confirmation.
- bool [show_all_conditions](#) (const Mat &frame)

7.10.1 Function Documentation

7.10.1.1 [acquireImage\(\)](#)

```
Mat acquireImage (
    const bool save )
```

It acquire a frame from the default camera of the pc.

Parameters

in	save	If save, or not, the acquired image to a file.
----	------	--

Returns

The Mat of the acquired frame.

7.10.1.2 [configure\(\)](#)

```
void configure (
    bool deploy,
    int img_id )
```

If deploy is true then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If deploy is false then it takes the img_id-th maps from the folder set in [Settings](#) and ask for visual confirmation.

If deploy is true then takes a photo from the camera, shows the various filters and asks if they are visually correct. If not then it allows to set the various filters through trackbars. If deploy is false then it takes the img_id-th maps from the folder set in [Settings](#) and ask for visual confirmation.

7.10.1.3 [show_all_conditions\(\)](#)

```
bool show_all_conditions (
    const Mat & frame )
```

Function to show a picture with various filters taken from [Settings](#). It then asks for visual confirmation.

Parameters

<i>frame</i>	The image to show.
--------------	--------------------

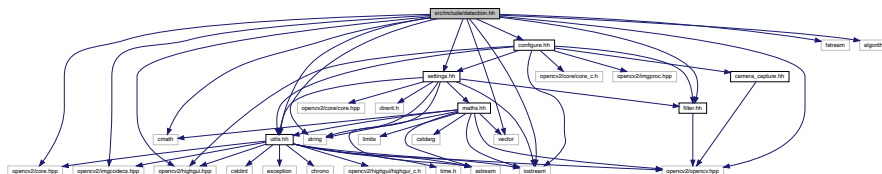
Returns

True if the filters are okay, false otherwise.

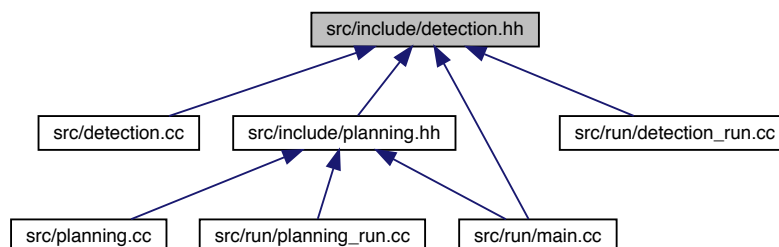
7.11 src/include/detection.hh File Reference

```
#include <utils.hh>
#include <settings.hh>
#include <filter.hh>
#include <configure.hh>
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#include <algorithm>
#include <vector>
#include <opencv2/highgui.hpp>
#include <opencv2/core.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/imgcodecs.hpp>
```

Include dependency graph for detection.hh:



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



Enumerations

- enum `COLOR_TYPE` {
`RED`, `GREEN`, `BLUE`, `BLACK`,
`CYAN` }

Functions

- `int detection ()`
Loads some images and detects shapes according to different colors.
- void `computeConversionParameters` (Mat &transf)
- `Point2< int > localize ()`
- `Point2< int > localize` (const Mat &img)
- void `shape_detection` (const Mat &img, const `COLOR_TYPE` color)
Detect shapes inside the image according to the variable 'color'.
- void `erode_dilation` (Mat &img, const `COLOR_TYPE` color)
It apply some filtering function for isolate the subject and remove the noise.
- void `find_contours` (const Mat &img, const Mat &original, const `COLOR_TYPE` color)
Given an image, in black/white format, identify all the borders that delimit the shapes.
- `int number_recognition` (Rect blob, const Mat &base)
Detect a number on an image inside a region of interest.
- void `save_convex_hull` (const vector< vector< Point >> &contours, const `COLOR_TYPE` color)
Given some vector save it in a xml file.
- void `load_number_template ()`
Load some templates and save them in the global variable 'templates'.
- void `crop_number_section` (Mat &processROI)
Given an image identify the region of interest(ROI) and crop it out.

7.11.1 Enumeration Type Documentation

7.11.1.1 COLOR_TYPE

enum `COLOR_TYPE`

Enumerator

RED	
GREEN	
BLUE	
BLACK	
CYAN	

7.11.2 Function Documentation

7.11.2.1 computeConversionParameters()

```
void computeConversionParameters (
    Mat & transf )
```

7.11.2.2 crop_number_section()

```
void crop_number_section (
    Mat & ROI )
```

Given an image identify the region of interest(ROI) and crop it out.

Parameters

<i>in, out</i>	<i>ROI</i>	Is the image that the function will going to elaborate.
----------------	------------	---

7.11.2.3 detection()

```
int detection ( )
```

Loads some images and detects shapes according to different colors.

Returns

Return 0 if the function reach the end.

7.11.2.4 erode_dilation()

```
void erode_dilation (
    Mat & img,
    const COLOR_TYPE color )
```

It apply some filtering function for isolate the subject and remove the noise.

An example of the sub functions called are: GaussianBlur, Erosion, Dilation and Threshold.

Parameters

<i>in, out</i>	<i>img</i>	Is the image on which the function apply the filtering.
<i>in</i>	<i>color</i>	It is the type of reference color. According to the color the filtering functions apply can change in the type and in the order.

7.11.2.5 find_contours()

```
void find_contours (
    const Mat & img,
    const Mat & original,
    const COLOR_TYPE color )
```

Given an image, in black/white format, identify all the borders that delimit the shapes.

Parameters

in	<i>img</i>	Is an image in HSV format at the base of the elaboration process.
out	<i>original</i>	It is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	It is the type of reference color.

7.11.2.6 load_number_template()

```
void load_number_template ( )
```

Load some templates and save them in the global variable 'templates'.

7.11.2.7 localize() [1/2]

```
Point2<int> localize ( )
```

7.11.2.8 localize() [2/2]

```
Point2<int> localize (
    const Mat & img )
```

7.11.2.9 number_recognition()

```
int number_recognition (
    Rect blob,
    const Mat & base )
```

Detect a number on an image inside a region of interest.

Parameters

in	<i>blob</i>	Identify the region of interest inside the image 'base'.
in	<i>base</i>	Is the image where the function will going to search the number.

Returns

The number recognise, '-1' otherwise.

7.11.2.10 save_convex_hull()

```
void save_convex_hull (
    const vector< vector< Point >> & contours,
    const COLOR_TYPE color )
```

Given some vector save it in a xml file.

Parameters

in	<i>contours</i>	Is a vector that is saved in a xml file.
in	<i>color</i>	It is the type of reference color, according to which the function decide if saved ('color==GREEN') or not ('otherwise') the vector 'victims'.

7.11.2.11 shape_detection()

```
void shape_detection (
    const Mat & img,
    const COLOR_TYPE color )
```

Detect shapes inside the image according to the variable 'color'.

Parameters

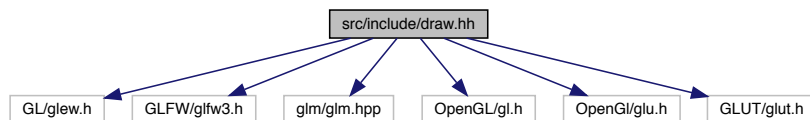
in	<i>img</i>	Image on which the research will done.
in	<i>color</i>	It is the type of reference color. These color identify the possible spectrum that the function search on the image.

7.12 src/include/draw.hh File Reference

```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <glm/glm.hpp>
```

```
#include <OpenGL/gl.h>
#include <OpenGL/glu.h>
#include <GLUT/glut.h>
```

Include dependency graph for draw.hh:



Namespaces

- [DW](#)

Typedefs

- typedef uint unsigned [int](#)

Functions

- void [DW::init](#) (x, y, GLfloat *vertices_buffer={0.0f})
- void [DW::changeBuffer](#) (GLfloat *vertices_buffer, uint dim)

Variables

- GLFWwindow * [DW::window](#)
- GLuint [DW::map_buffer](#)

7.12.1 Typedef Documentation

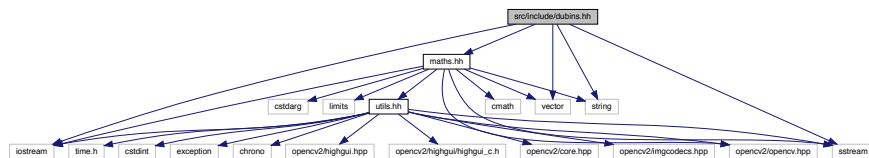
7.12.1.1 int

```
typedef uint unsigned int
```

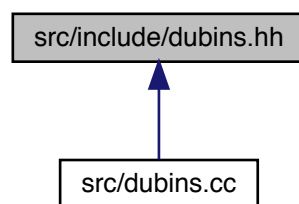
7.13 src/include/dubins.hh File Reference

```
#include <maths.hh>
#include <iostream>
#include <sstream>
#include <vector>
#include <string>
```

Include dependency graph for dubins.hh:



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



Classes

- class [Curve< T >](#)
- class [DubinsArc< T1, T2 >](#)

Class to store a maneuver of [Dubins](#). It inherits from [Curve](#). Since each [Dubins](#) is formed of atmost 3 maneuvers, this class is meant to store one of this maneuver, which can be L, R or S respectively Left, Right, Straight.

- class [Dubins< T >](#)

Class to store a [Dubins](#) curve. This class inherits from [Curve](#) and is composed of three [DubinsArc](#).

- class [DubinsSet< T >](#)

Given a set of point, compute the shortest set of [Dubins](#) that allows to go from start to end through all points.

Macros

- `#define` [PIECE_LENGTH](#) 2
- `#define` [PREC](#) 100000
- `#define` [KMAX](#) 0.5
- `#define` [DELTA](#) M_PI/180.0

Functions

- static double [sinc](#) (double t)
- [Configuration2](#)< double > [circline](#) (double _L, [Configuration2](#)< double > _P0, double _K)
- [Tuple](#)< [Angle](#) > [toBase](#) ([Tuple](#)< [Angle](#) > z, int n, int base, const [Angle](#) &inc, int startPos, int endPos)

*Convert a value in base 10 to base *base* in a [Tuple](#). To each value an *inc* is multiplied and the initial [Angle](#) is added.*
- void [disp](#) ([Tuple](#)< [Tuple](#)< [Angle](#) > > &t, [Tuple](#)< [Angle](#) > &z, int N, const [Angle](#) &inc, int startPos=0, int endPos=0)

Compute the arrangements. Since each arrangement can be computed as n_{parts} , where each values is then multiplied for the increment and is added to the initial values.
- template<class T >
 [vector](#)< [Point2](#)< T > > [reduce_points](#) ([Tuple](#)< [Point2](#)< T > > init_points)

Variables

- double [elapsedScale](#)
- double [elapsedPrimitives](#)
- double [elapsedBest](#)
- double [elapsedArcs](#)
- double [elapsedCheck](#)
- unsigned long [countTries](#)
- double [elapsedVar](#)
- double [elapsedCirc](#)
- double [elapsedSet](#)
- double [elapsedLSL](#)
- double [elapsedRSR](#)
- double [elapsedLSR](#)
- double [elapsedRSL](#)
- double [elapsedRLR](#)
- double [elapsedLRL](#)

7.13.1 Macro Definition Documentation

7.13.1.1 DELTA

```
#define DELTA M_PI/180.0
```

7.13.1.2 KMAX

```
#define KMAX 0.5
```

7.13.1.3 PIECE_LENGTH

```
#define PIECE_LENGTH 2
```

7.13.1.4 PREC

```
#define PREC 100000
```

7.13.2 Function Documentation

7.13.2.1 circline()

```
Configuration2<double> circline (
    double _L,
    Configuration2< double > _P0,
    double _K )
```

Computes an arrival point from an initial configuration through an arc of length `_L` and curvature `_K`.

Parameters

<code>_L</code>	The length of the arch.
<code>_P0</code>	The starting Configuration2 of the arc.
<code>_K</code>	The curvature of the arc.

Returns

The ending [Configuration2](#) of the arc.

7.13.2.2 disp()

```
void disp (
    Tuple< Tuple< Angle > > & t,
    Tuple< Angle > & z,
    int N,
    const Angle & inc,
    int startPos = 0,
    int endPos = 0 )
```

Compute the arrangements. Since each arrangement can be computed as n_{parts} , where each values is then multiplied for the increment and is added to the initial values.

Parameters

<i>t</i>	A Tuple containing all the Tuples containing the Angles .
<i>z</i>	A Tuple containing all the initial Angles .
<i>N</i>	The number of iterations. Each iteration is going to be converted in base parts.
<i>inc</i>	The increment to give each initial Angle .
<i>startPos</i>	The initial position to consider in Tuple .
<i>endPos</i>	The final position to consider in Tuple .

Parameters

<i>z</i>	Vector to use
<i>N</i>	Number of time to "iterate"
<i>inc</i>	Incrementation

7.13.2.3 `reduce_points()`

```
template<class T >
vector<Point2<T> > reduce_points (
    Tuple< Point2< T > > init_points )
```

7.13.2.4 `sinc()`

```
static double sinc (
    double t ) [static]
```

Compute the sinc of the function defined as:

$$\text{sinc}(t) = \frac{\sin(t)}{t} \quad t \neq 0 \quad t = 0$$

Parameters

<i>t</i>	The value of the angle to be used.
----------	------------------------------------

Returns

The result of the previous formula.

7.13.2.5 `toBase()`

```
Tuple<Angle> toBase (
    Tuple< Angle > z,
```

```

    int n,
    int base,
    const Angle & inc,
    int startPos,
    int endPos )

```

Convert a value in base 10 to base `base` in a [Tuple](#). To each value an `inc` is multiplied and the initial [Angle](#) is added.

Parameters

<code>z</code>	A Tuple containing all the initial Angles .
<code>n</code>	The value to be converted.
<code>base</code>	The base.
<code>inc</code>	The increment.
<code>startPos</code>	The starting position of the Tuple of Angles .
<code>endPos</code>	The ending position of the Tuple of Angles .

Returns

7.13.3 Variable Documentation

7.13.3.1 countTries

```
unsigned long countTries
```

7.13.3.2 elapsedArcs

```
double elapsedArcs
```

7.13.3.3 elapsedBest

```
double elapsedBest
```

7.13.3.4 elapsedCheck

```
double elapsedCheck
```

7.13.3.5 elapsedCirc

```
double elapsedCirc
```

7.13.3.6 elapsedLRL

```
double elapsedLRL
```

7.13.3.7 elapsedLSL

```
double elapsedLSL
```

7.13.3.8 elapsedLSR

```
double elapsedLSR
```

7.13.3.9 elapsedPrimitives

```
double elapsedPrimitives
```

7.13.3.10 elapsedRLR

```
double elapsedRLR
```

7.13.3.11 elapsedRSL

```
double elapsedRSL
```

7.13.3.12 elapsedRSR

```
double elapsedRSR
```

7.13.3.13 elapsedScale

```
double elapsedScale
```

7.13.3.14 elapsedSet

```
double elapsedSet
```

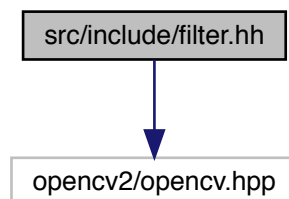
7.13.3.15 elapsedVar

```
double elapsedVar
```

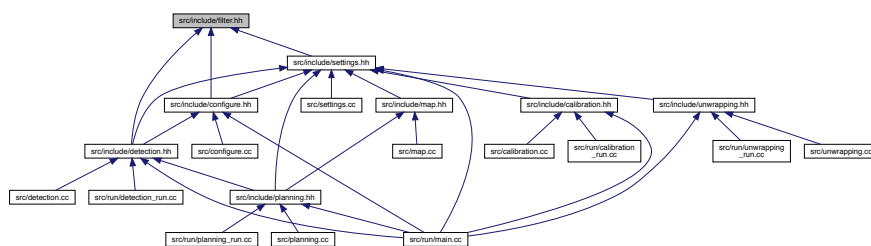
7.14 src/include/filter.hh File Reference

```
#include <opencv2/opencv.hpp>
```

Include dependency graph for filter.hh:



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



Classes

- class [Filter](#)

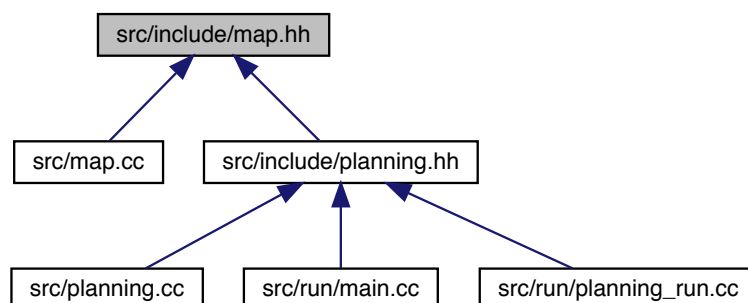
7.15 src/include/map.hh File Reference

```
#include <vector>
#include <set>
#include <queue>
#include <tuple>
#include <iostream>
#include <iomanip>
#include <maths.hh>
#include <settings.hh>
#include <utils.hh>
#include <objects.hh>
#include <opencv2/highgui.hpp>
#include <opencv2/core.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/imgcodecs.hpp>
```

Include dependency graph for map.hh:



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



Classes

- class [Mapp](#)

Enumerations

- enum [OBJ_TYPE](#) {
[FREE](#), [VICT](#), [OBST](#), [GATE](#),
[BODA](#) }

7.15.1 Enumeration Type Documentation

7.15.1.1 OBJ_TYPE

enum [OBJ_TYPE](#)

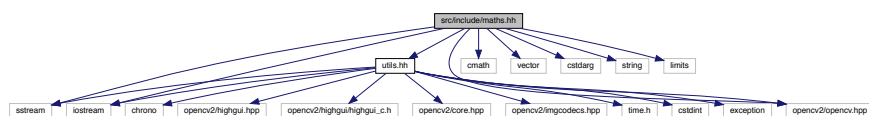
Enumerator

FREE	
VICT	
OBST	
GATE	
BODA	

7.16 src/include/maths.hh File Reference

```
#include "utils.hh"
#include <iostream>
#include <cmath>
#include <vector>
#include <cstdlib>
#include <sstream>
#include <string>
#include <opencv2/opencv.hpp>
#include <limits>
```

Include dependency graph for maths.hh:



Functions

- bool `equal` (const double &A, const double &B, const double E=`Epsi`)
Function to compare two doubles as $|A - B| < \varepsilon$.
- template<class T >
T `pow2` (const T x)

Variables

- const double `DEGTORAD` =(M_PI/180.0)
- const double `RADTODEG` =(180.0/M_PI)

7.16.1 Macro Definition Documentation

7.16.1.1 A_180

```
#define A_180 Angle(180, Angle::DEG)
```

Default `Angle` for 180 degree.

7.16.1.2 A_2PI

```
#define A_2PI Angle(6.2831853071-Epsi, Angle::RAD)
```

Default `Angle` for 2pi rad.

7.16.1.3 A_360

```
#define A_360 Angle(360.0-Epsi, Angle::DEG)
```

Default `Angle` for 360 degree.

7.16.1.4 A_90

```
#define A_90 Angle(90, Angle::DEG)
```

Default `Angle` for 90 degree.

7.16.1.5 A_DEG_NULL

```
#define A_DEG_NULL Angle(0, Angle::DEG)
```

Default `Angle` for 0 rad.

7.16.1.6 A_PI

```
#define A_PI Angle(M_PI, Angle::RAD)
```

Default `Angle` for pi rad.

7.16.1.7 A_PI2

```
#define A_PI2 Angle(M_PI/2.0, Angle::RAD)
```

Default `Angle` for pi/2 rad.

7.16.1.8 A_RAD_NULL

```
#define A_RAD_NULL Angle(0, Angle::RAD)
```

Default `Angle` for 0 degree.

7.16.1.9 DInf

```
#define DInf numeric_limits<double>::infinity()
```

7.16.1.10 Epsi

```
#define Epsi numeric_limits<double>::epsilon()
```

7.16.1.11 tupleConstIter

```
#define tupleConstIter const typename vector<T>::iterator
```

7.16.1.12 tupleIter

```
#define tupleIter typename vector<T>::iterator
```

7.16.2 Enumeration Type Documentation

7.16.2.1 DISTANCE_TYPE

```
enum DISTANCE\_TYPE
```

Enumerator

EUCLIDEAN	
MANHATTAN	

7.16.3 Function Documentation

7.16.3.1 equal()

```
bool equal (
    const double & A,
    const double & B,
    const double E = Epsi ) [inline]
```

Function to compare two doubles as $|A - B| < \varepsilon$.

Parameters

in	<i>A</i>	First number.
in	<i>B</i>	Second number.
in	<i>E</i>	ε , set at <code>std::numeric_limits<double>::epsilon()</code> as default.

Returns

true if $|A - B| < \varepsilon$, false otherwise.

7.16.3.2 pow2()

```
template<class T >
T pow2 (
    const T x ) [inline]
```

7.16.4 Variable Documentation

7.16.4.1 DEGTORAD

```
const double DEGTORAD =(M_PI/180.0)
```

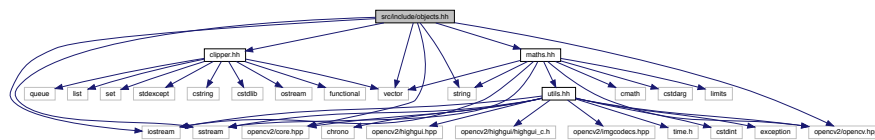
7.16.4.2 RADTODEG

```
const double RADTODEG =(180.0/M_PI)
```

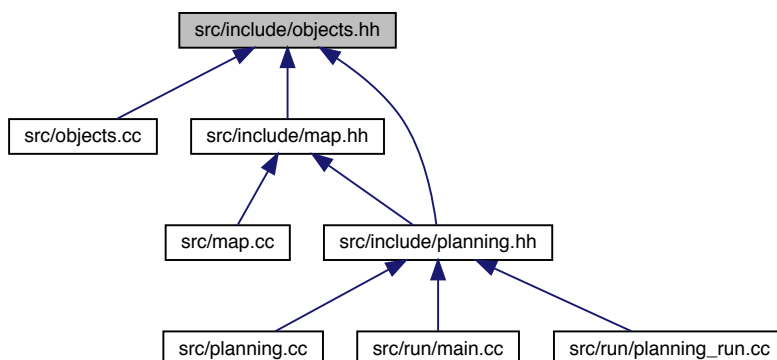
7.17 src/include/objects.hh File Reference

```
#include <iostream>
#include <vector>
#include <sstream>
#include <string>
#include <opencv2/core.hpp>
#include <opencv2/opencv.hpp>
#include "clipper.hh"
#include "maths.hh"
#include "utils.hh"
```

Include dependency graph for objects.hh:



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



Classes

- class [Object](#)
- class [Obstacle](#)
- class [Gate](#)
- class [Victim](#)

7.18.1.1 createMapp()

```
Mapp* createMapp ( )
```

The goal is to load, all the necessary data, from files and create a [Mapp](#) that store everything.

Returns

The created mapp.

7.18.1.2 loadVP()

```
void loadVP (
    vector< Point2< int > > & vp,
    FileNode fn )
```

The function load from the given fileNode a vector of [Point2<int>](#).

Parameters

out	<i>vp</i>	The location where to save the loaded vector.
in	<i>fn</i>	The fileNode from which to load the vector.

7.18.1.3 loadVVP()

```
void loadVVP (
    vector< vector< Point2< int > > > & vvp,
    FileNode fn )
```

The function load from the given fileNode a vector of vectors of [Point2<int>](#).

Parameters

out	<i>vvp</i>	The location where to save the loaded vector of vectors.
in	<i>fn</i>	The fileNode from which to load the vector of vectors.

7.18.1.4 planning()

```
pair< vector<Point2<int> >, Mapp* > planning ( )
```

The function plan a route from the actual position of the robot up to the final gate through all the victims.

All the data about the objects are loaded from the files previously saved. Then a [Mapp](#) is created and on that structure, thanks to a minPath function and a lot of dubin curves, the best route is computed.

7.20.1 Function Documentation

7.20.1.1 find_rect()

```
void find_rect (
    vector< Point > & _rect,
    const int & width,
    const int & height )
```

Since the border of the arena might not always be clean but might have some imperfection, this functions computes the four vertexes taking all the points and computing the four that are the closest to the corner of the image.

Parameters

in	<i>_rect</i>	The vector of cv::Point to work on.
in	<i>width</i>	The width of the image.
in	<i>height</i>	The height of the image.

7.20.1.2 loadCoefficients()

```
void loadCoefficients (
    const string filename,
    Mat & camera_matrix,
    Mat & dist_coeffs )
```

Load coefficients from a file.

Load two matrix 'camera_matrix' and 'distortion_coefficients' from the xml file passed.

Parameters

in	<i>filename</i>	The string that identify the location of the xml file.
out	<i>camera_matrix</i>	Where the 'camera_matrix' matrix is saved.
out	<i>dist_coeffs</i>	Where the 'distortion_coefficients' matrix is saved.

7.20.1.3 unwrapping()

```
int unwrapping ( )
```

Take some images according to a xml and unwrap the black rectangle inside the image after applying undistortion transformation.

Macros

- `#define NAME(x) #x`
Returns the name of the variable.
- `#define COUT(x)`
Print a messag to stderr.
- `#define INFO(msg)`
Print the name of a variable and its content. Only if DEBUG is defined.

Typedefs

- `typedef chrono::high_resolution_clock Clock`

Enumerations

- `enum CHRONO::TIME_TYPE { CHRONO::SEC, CHRONO::MSEC, CHRONO::MUSEC, CHRONO::NSEC }`
- `enum EXCEPTION_TYPE { EXISTS, SIZE }`

Functions

- `string CHRONO::getType (TIME_TYPE type, string ret="")`
- `double CHRONO::getElapsed (Clock::time_point start, Clock::time_point stop, TIME_TYPE type=MUSEC)`
- `string CHRONO::getElapsed (Clock::time_point start, Clock::time_point stop, string ret, TIME_TYPE type=MUSEC)`
- `void my_imshow (const char *win_name, Mat img, bool reset=false)`
Function to show images in an order grill.
- `void mywaitkey ()`
Function to use after my_imshow() for keeping the image opened until a key is pressed.
- `void mywaitkey (string windowName)`
Function to use after my_imshow() for keeping the image opened until a key is pressed. When a key is pressed a specific window is closed.
- `int64_t timeutils::timespecDiff (struct timespec *timeA_p, struct timespec *timeB_p)`
- `double timeutils::getTimeS ()`

7.21.1 Macro Definition Documentation

7.21.1.1 COUT

```
#define COUT(  
    x )
```

Print a messag to stderr.

7.21.1.2 INFO

```
#define INFO(  
    msg )
```

Print the name of a variable and its content. Only if DEBUG is defined.

7.21.1.3 NAME

```
#define NAME(  
    x ) #x
```

Returns the name of the variable.

7.21.2 Typedef Documentation

7.21.2.1 Clock

```
typedef chrono::high_resolution_clock Clock
```

7.21.3 Enumeration Type Documentation

7.21.3.1 EXCEPTION_TYPE

```
enum EXCEPTION\_TYPE
```

Enumerator

EXISTS	
SIZE	

7.21.4 Function Documentation

7.21.4.1 my_imshow()

```
void my_imshow (  
    const char * win_name,
```

```
Mat img,  
bool reset = false )
```

Function to show images in an order grill.

Parameters

<i>win_name</i>	The name of the window to use.
<i>img</i>	The Mat containing the image.
<i>reset</i>	If true the image is going to be placed in 0,0 i.e. the top left corner of the screen.

7.21.4.2 mywaitkey() [1/2]

```
void mywaitkey ( )
```

Function to use after `my_imshow()` for keeping the image opened until a key is pressed.

7.21.4.3 mywaitkey() [2/2]

```
void mywaitkey (
    string windowName )
```

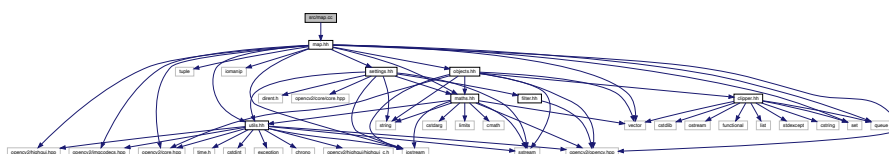
Function to use after `my_imshow()` for keeping the image opened until a key is pressed. When a key is pressed a specific window is closed.

Parameters

<i>windowName</i>	The window to close after pressing a key.
-------------------	---

7.22 src/map.cc File Reference

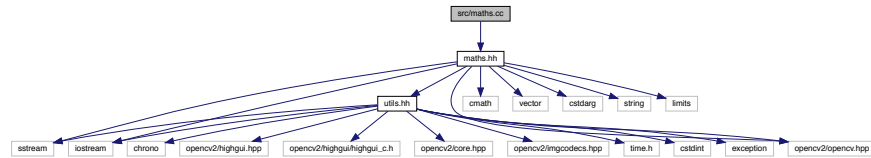
```
#include <map.hh>
```



7.23 src/maths.cc File Reference

```
#include "maths.hh"
```

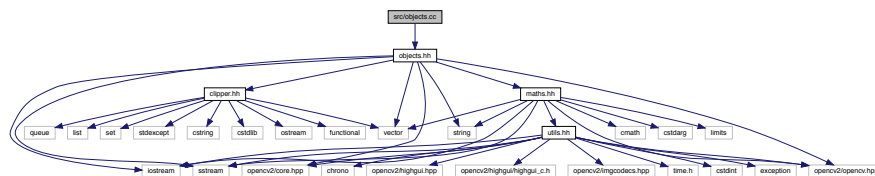
Include dependency graph for maths.cc:



7.24 src/objects.cc File Reference

```
#include "objects.hh"
```

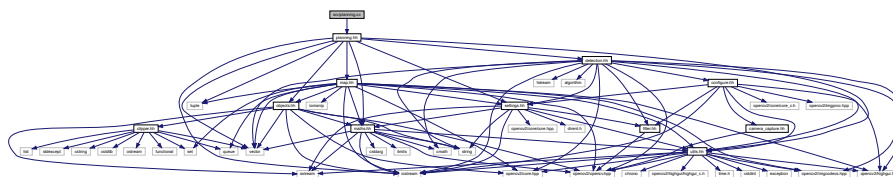
Include dependency graph for objects.cc:



7.25 src/planning.cc File Reference

```
#include "planning.hh"
```

Include dependency graph for planning.cc:



Functions

- `pair< vector< Point2< int > >, Mapp * > planning ()`
The function plan a route from the actual position of the robot up to the final gate through all the victims.
- `Mapp * createMapp ()`
The goal is to load, all the necessary data, from files and create a `Mapp` that store everything.
- `void loadVVP (vector< vector< Point2< int > > > &vvp, FileNode fn)`
The function load from the given fileNode a vector of vectors of `Point2<int>`.
- `void loadVP (vector< Point2< int > > &vp, FileNode fn)`
The function load from the given fileNode a vector of `Point2<int>`.

7.25.1 Function Documentation

7.25.1.1 createMapp()

```
Mapp* createMapp ( )
```

The goal is to load, all the necessary data, from files and create a [Mapp](#) that store everything.

Returns

The created mapp.

7.25.1.2 loadVP()

```
void loadVP (
    vector< Point2< int > > & vp,
    FileNode fn )
```

The function load from the given fileNode a vector of [Point2<int>](#).

Parameters

out	<i>vp</i>	The location where to save the loaded vector.
in	<i>fn</i>	The fileNode from which to load the vector.

7.25.1.3 loadVVP()

```
void loadVVP (
    vector< vector< Point2< int > > > & vvp,
    FileNode fn )
```

The function load from the given fileNode a vector of vectors of [Point2<int>](#).

Parameters

out	<i>vvp</i>	The location where to save the loaded vector of vectors.
in	<i>fn</i>	The fileNode from which to load the vector of vectors.

Functions

- [int main\(\)](#)

7.27.1 Function Documentation

7.27.1.1 main()

```
int main ( )
```

7.28 src/run/main.cc File Reference

```
#include <utils.hh>
#include <detection.hh>
#include <unwrapping.hh>
#include <calibration.hh>
#include <planning.hh>
#include <configure.hh>
#include <settings.hh>
#include <iostream>
```

Include dependency graph for main.cc:



Functions

- [int main\(\)](#)

Variables

- [Settings * sett = new Settings\(\)](#)
Global variable defined in [main.cc](#).

7.28.1 Function Documentation

7.28.1.1 main()

```
int main ( )
```

7.28.2 Variable Documentation

7.28.2.1 sett

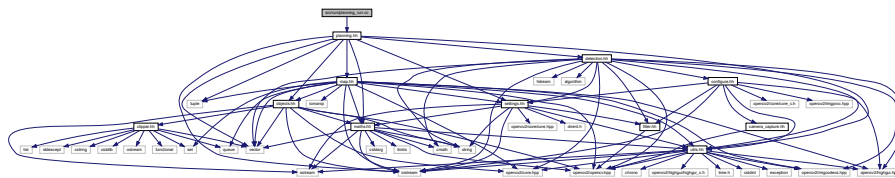
```
Settings* sett =new Settings()
```

Global variable defined in [main.cc](#).

7.29 src/run/planning_run.cc File Reference

```
#include <planning.hh>
```

Include dependency graph for planning_run.cc:



Functions

- [int main\(\)](#)

7.29.1 Function Documentation

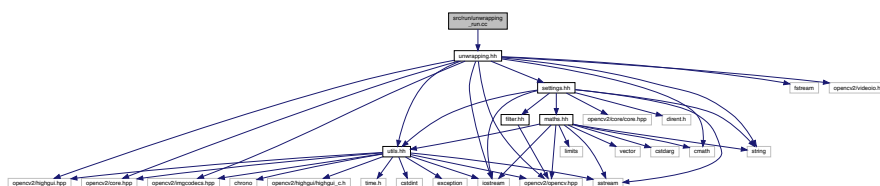
7.29.1.1 main()

```
int main ( )
```

7.30 src/run/unwrapping_run.cc File Reference

```
#include <unwrapping.hh>
```

Include dependency graph for unwrapping_run.cc:



Functions

- `int main ()`

7.30.1 Function Documentation

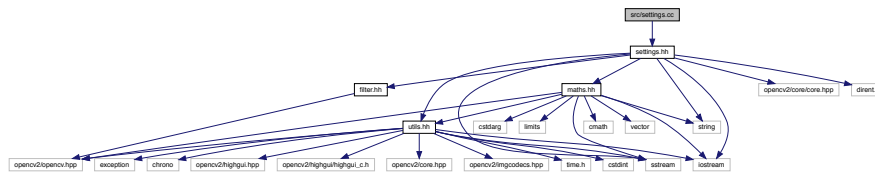
7.30.1.1 main()

```
int main ( )
```

7.31 src/settings.cc File Reference

```
#include "settings.hh"
```

Include dependency graph for settings.cc:



Macros

- `#define NPOS string::npos`
Shortcut for `string::npos`.

Functions

- `vector< string > getFiles (const string &path)`
Function to get all files in directory. From <https://stackoverflow.com/questions/612097/how-can-i-get-the-list-of-files-in-a-directory>
- `void vecToFile (FileStorage &fs, vector< int > x)`

7.31.1 Macro Definition Documentation

7.31.1.1 NPOS

```
#define NPOS string::npos
```

Shortcut for `string::npos`.

Macros

- `#define AREA_RATIO 0.7`
- `#define AREA_MIN 500`

Functions

- static float `distance` (Point c1, Point c2)
Compute the euclidean distance.
- int `unwrapping` ()
Take some images according to a xml and unwrap the black rectangle inside the image after applying undistortion trasformation.
- void `find_rect` (vector< Point > &_rect, const int &width, const int &height)
Since the border of the arena might not always be clean but might have some imperfection, this functions computes the four vertexes taking all the points and computing the four that are the closest to the corner of the image.
- void `loadCoefficients` (const string filename, Mat &camera_matrix, Mat &dist_coeffs)
Load coefficients from a file.

7.32.1 Macro Definition Documentation

7.32.1.1 AREA_MIN

```
#define AREA_MIN 500
```

7.32.1.2 AREA_RATIO

```
#define AREA_RATIO 0.7
```

7.32.2 Function Documentation

7.32.2.1 distance()

```
static float distance (  
    Point c1,  
    Point c2 ) [static]
```

Compute the euclidean distance.

Parameters

in, out	<i>c1</i>	The first point.
in, out	<i>c2</i>	The second point.

Returns

The euclidean distance.

7.32.2.2 find_rect()

```
void find_rect (
    vector< Point > & _rect,
    const int & width,
    const int & height )
```

Since the border of the arena might not always be clean but might have some imperfection, this functions computes the four vertexes taking all the points and computing the four that are the closest to the corner of the image.

Parameters

in	<i>_rect</i>	The vector of cv::Point to work on.
in	<i>width</i>	The width of the image.
in	<i>height</i>	The height of the image.

7.32.2.3 loadCoefficients()

```
void loadCoefficients (
    const string filename,
    Mat & camera_matrix,
    Mat & dist_coeffs )
```

Load coefficients from a file.

Load two matrix 'camera_matrix' and 'distortion_coefficients' from the xml file passed.

Parameters

in	<i>filename</i>	The string that identify the location of the xml file.
out	<i>camera_matrix</i>	Where the 'camera_matrix' matrix is saved.
out	<i>dist_coeffs</i>	Where the 'distortion_coefficients' matrix is saved.

7.32.2.4 unwrapping()

```
int unwrapping ( )
```

Take some images according to a xml and unwrap the black rectangle inside the image after applying undistortion transformation.

Load from the xml file 'data/settings.xml' the name of some images, load the images from the file, apply the calibration (undistortion transformation) thanks to the matrices load with the 'loadCoefficients' function. Then, with the use of a filter for the black region of interest (a rectangle) is identified and all the perspective is rotated for reach a top view of the rectangle. Finally, the images are saved on some files.

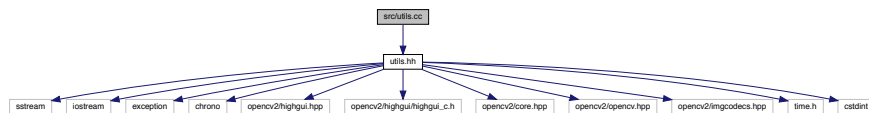
Returns

A 0 is return if the function reach the end.

7.33 src/utils.cc File Reference

```
#include "utils.hh"
```

Include dependency graph for utils.cc:



Namespaces

- [timeutils](#)

Functions

- void [my_imshow](#) (const char *win_name, cv::Mat img, bool reset)
Function to show images in an order grill.
- void [mywaitkey](#) ()
Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed.
- void [mywaitkey](#) (string windowName)
Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed. When a key is pressed a specific window is closed.
- int64_t [timeutils::timespecDiff](#) (struct timespec *timeA_p, struct timespec *timeB_p)
- double [timeutils::getTimeS](#) ()

7.33.1 Function Documentation

7.33.1.1 my_imshow()

```
void my_imshow (
    const char * win_name,
    cv::Mat img,
    bool reset )
```

Function to show images in an order grill.

Parameters

<i>win_name</i>	The name of the window to use.
<i>img</i>	The Mat containing the image.
<i>reset</i>	If true the image is going to be placed in 0,0 i.e. the top left corner of the screen.

7.33.1.2 mywaitkey() [1/2]

```
void mywaitkey ( )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed.

7.33.1.3 mywaitkey() [2/2]

```
void mywaitkey (
    string windowName )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed. When a key is pressed a specific window is closed.

Parameters

<i>windowName</i>	The window to close after pressing a key.
-------------------	---

Index

- `_compare`
 - `detection.cc`, 218
 - `~CameraCapture`
 - `CameraCapture`, 61
 - `~ClipperBase`
 - `ClipperLib::ClipperBase`, 68
 - `~ClipperOffset`
 - `ClipperLib::ClipperOffset`, 74
 - `~PolyNode`
 - `ClipperLib::PolyNode`, 170
 - `~PolyTree`
 - `ClipperLib::PolyTree`, 172
 - `~Settings`
 - `Settings`, 177
 - `~clipperException`
 - `ClipperLib::clipperException`, 73
- `a`
 - `MyException< T >`, 148
- `A_180`
 - `maths.hh`, 246
- `A_2PI`
 - `maths.hh`, 246
- `A_360`
 - `maths.hh`, 246
- `A_90`
 - `maths.hh`, 246
- `A_DEG_NULL`
 - `maths.hh`, 246
- `A_PI`
 - `maths.hh`, 247
- `A_PI2`
 - `maths.hh`, 247
- `A_RAD_NULL`
 - `maths.hh`, 247
- `Abs`
 - `ClipperLib`, 17
- `acquireImage`
 - `configure.cc`, 214
 - `configure.hh`, 229
- `add`
 - `Angle`, 38
 - `Tuple< T >`, 194
- `AddBoundsToLML`
 - `ClipperLib::ClipperBase`, 68
- `addIfNot`
 - `Tuple< T >`, 194
- `addObject`
 - `Mapp`, 136
- `addObjects`
 - `Mapp`, 136, 137
- `AddPath`
 - `ClipperLib::ClipperBase`, 68
 - `ClipperLib::ClipperOffset`, 74
- `AddPaths`
 - `ClipperLib::ClipperBase`, 68
 - `ClipperLib::ClipperOffset`, 74
- `AddPolyNodeToPaths`
 - `ClipperLib`, 17
- `addUnMap`
 - `Settings`, 177
- `ahead`
 - `Tuple< T >`, 194
- `Angle`, 35
 - `add`, 38
 - `Angle`, 37
 - `ANGLE_TYPE`, 37
 - `checkValue`, 38
 - `copy`, 38
 - `cos`, 39
 - `DEG`, 37
 - `degToRad`, 39
 - `div`, 39
 - `equal`, 39
 - `get`, 40
 - `getType`, 40
 - `getTypeName`, 40
 - `greater`, 40
 - `INVALID`, 37
 - `less`, 41
 - `mul`, 41
 - `normalize`, 41
 - `operator *`, 42
 - `operator *=`, 42
 - `operator double`, 42
 - `operator float`, 43
 - `operator int`, 43
 - `operator long`, 43
 - `operator !=`, 43
 - `operator <`, 46
 - `operator <<`, 50
 - `operator <=`, 46
 - `operator >`, 47
 - `operator >=`, 47
 - `operator +`, 44
 - `operator +=`, 44
 - `operator -`, 45
 - `operator -=`, 45
 - `operator /`, 45

- operator/=, [46](#)
- operator=, [46](#), [47](#)
- operator==, [47](#)
- RAD, [37](#)
- radToDeg, [47](#)
- set, [48](#)
- setType, [48](#)
- sin, [48](#)
- sub, [49](#)
- tan, [49](#)
- to_string, [49](#)
- toDeg, [50](#)
- toRad, [50](#)
- angle
 - Configuration2< T1 >, [78](#)
- ANGLE_TYPE
 - Angle, [37](#)
- ArcTolerance
 - ClipperLib::ClipperOffset, [75](#)
- Area
 - ClipperLib, [17](#)
- AREA_MIN
 - unwrapping.cc, [267](#)
- AREA_RATIO
 - unwrapping.cc, [267](#)
- aspectRatio
 - CalSettings, [56](#)
- atImageList
 - CalSettings, [56](#)
- b
 - MyException< T >, [148](#)
- baseDistance
 - Mapp, [145](#)
- begin
 - Curve< T >, [90](#)
 - Tuple< T >, [195](#)
- BLACK
 - detection.hh, [231](#)
 - Settings, [176](#)
- blackMask
 - Settings, [186](#)
- BLUE
 - detection.hh, [231](#)
 - Settings, [176](#)
- blueMask
 - Settings, [186](#)
- boardSize
 - CalSettings, [56](#)
- BODA
 - map.hh, [244](#)
- borderSize
 - Mapp, [145](#)
- borderSizeDefault
 - Mapp, [145](#)
- Bot
 - ClipperLib::TEdge, [189](#)
- bottom
 - ClipperLib::IntRect, [129](#)
- BottomPt
 - ClipperLib::OutRec, [157](#)
- calcBoardCornerPositions
 - calibration.cc, [206](#)
- calibFixPrincipalPoint
 - CalSettings, [56](#)
- CALIBRATED
 - calibration.hh, [223](#)
- calibration
 - calibration.cc, [206](#)
 - calibration.hh, [223](#)
- calibration.cc
 - calcBoardCornerPositions, [206](#)
 - calibration, [206](#)
 - computeReprojectionErrors, [206](#)
 - read, [207](#)
 - runCalibration, [207](#)
 - runCalibrationAndSave, [208](#)
 - saveCameraParams, [208](#)
- calibration.hh
 - CALIBRATED, [223](#)
 - calibration, [223](#)
 - CAPTURING, [223](#)
 - DETECTION, [223](#)
 - runCalibrationAndSave, [223](#)
 - sett, [224](#)
- calibration_run.cc
 - main, [262](#)
- calibrationFile
 - Settings, [186](#)
- calibrationPattern
 - CalSettings, [56](#)
- calibZeroTangentDist
 - CalSettings, [56](#)
- CalSettings, [51](#)
 - aspectRatio, [56](#)
 - atImageList, [56](#)
 - boardSize, [56](#)
 - calibFixPrincipalPoint, [56](#)
 - calibrationPattern, [56](#)
 - calibZeroTangentDist, [56](#)
 - CalSettings, [53](#)
 - cameraID, [56](#)
 - CHESSBOARD, [53](#)
 - delay, [57](#)
 - fixK1, [57](#)
 - fixK2, [57](#)
 - fixK3, [57](#)
 - fixK4, [57](#)
 - fixK5, [57](#)
 - flag, [58](#)
 - flipVertical, [58](#)
 - goodInput, [58](#)
 - IMAGE_LIST, [53](#)
 - imageList, [58](#)
 - input, [58](#)
 - inputCapture, [58](#)
 - InputType, [53](#)

- inputType, [58](#)
- INVALID, [53](#)
- isListOfImages, [53](#)
- nextImage, [54](#)
- NOT_EXISTING, [53](#)
- nrFrames, [59](#)
- outputFileName, [59](#)
- Pattern, [53](#)
- read, [54](#)
- readStringList, [54](#)
- showUndistorted, [59](#)
- squareSize, [59](#)
- useFisheye, [59](#)
- validate, [55](#)
- write, [55](#)
- writeExtrinsics, [59](#)
- writePoints, [60](#)
- camera_capture.cc
 - SDEBUG, [210](#)
- CameraCapture, [60](#)
 - ~CameraCapture, [61](#)
 - CameraCapture, [60](#)
 - grab, [61](#)
 - isAlive, [61](#)
 - isOpened, [61](#)
 - loadCoefficients, [62](#)
 - startCamera, [62](#)
- CameraCapture::input_options_t, [120](#)
 - cameraFPS, [121](#)
 - frameHeight_px, [121](#)
 - frameWidth_px, [121](#)
 - input_options_t, [121](#)
 - nameCamera, [121](#)
- cameraFPS
 - CameraCapture::input_options_t, [121](#)
- cameraID
 - CalSettings, [56](#)
- CAPTURING
 - calibration.hh, [223](#)
- cellsFromSegment
 - Mapp, [137](#)
- cellSize
 - Mapp, [145](#)
- center
 - Object, [153](#)
- changeBuffer
 - DW, [32](#)
- changeMask
 - Settings, [177](#), [178](#)
- check
 - Dubins< T >, [96](#)
- checkSegment
 - Mapp, [138](#)
- checkSegmentCollisionWithType
 - Mapp, [138](#)
- checkValue
 - Angle, [38](#)
- CHESSBOARD
 - CalSettings, [53](#)
- ChildCount
 - ClipperLib::PolyNode, [170](#)
- Childs
 - ClipperLib::PolyNode, [171](#)
- CHRONO, [9](#)
 - getElapsed, [9](#), [10](#)
 - getType, [10](#)
 - MSEC, [9](#)
 - MUSEC, [9](#)
 - NSEC, [9](#)
 - SEC, [9](#)
 - TIME_TYPE, [9](#)
- clnt
 - ClipperLib, [13](#)
- circline
 - dubins.hh, [238](#)
- clean
 - Settings, [178](#)
- cleanAndRead
 - Settings, [178](#)
- CleanPolygon
 - ClipperLib, [18](#)
- CleanPolygons
 - ClipperLib, [18](#)
- Clear
 - ClipperLib::ClipperBase, [68](#)
 - ClipperLib::ClipperOffset, [74](#)
 - ClipperLib::PolyTree, [172](#)
- Clipper
 - ClipperLib::Clipper, [63](#)
 - ClipperLib::PolyNode, [170](#)
 - ClipperLib::PolyTree, [173](#)
- clipper.cc
 - HORIZONTAL, [213](#)
 - NEAR_ZERO, [213](#)
 - TOLERANCE, [213](#)
- clipper.hh
 - CLIPPER_VERSION, [227](#)
 - use_lines, [228](#)
- CLIPPER_VERSION
 - clipper.hh, [227](#)
- ClipperBase
 - ClipperLib::ClipperBase, [68](#)
- clipperException
 - ClipperLib::clipperException, [73](#)
- ClipperLib, [10](#)
 - Abs, [17](#)
 - AddPolyNodeToPaths, [17](#)
 - Area, [17](#)
 - clnt, [13](#)
 - CleanPolygon, [18](#)
 - CleanPolygons, [18](#)
 - ClipType, [14](#)
 - ClosedPathsFromPolyTree, [18](#)
 - ctDifference, [15](#)
 - ctIntersection, [15](#)
 - ctUnion, [15](#)

- ctXor, 15
- def_arc_tolerance, 31
- Direction, 15
- DisposeOutPts, 18
- DistanceFromLineSqrd, 19
- DistanceSqrd, 19
- dLeftToRight, 15
- dRightToLeft, 15
- DupOutPt, 19
- E2InsertsBeforeE1, 19
- EdgeList, 13
- EdgesAdjacent, 19
- EdgeSide, 15
- EndType, 15
- esLeft, 15
- esRight, 15
- etClosedLine, 15
- etClosedPolygon, 15
- etOpenButt, 15
- etOpenRound, 15
- etOpenSquare, 15
- ExcludeOp, 19
- FindNextLocMin, 20
- FirstIsBottomPt, 20
- GetBottomPt, 20
- GetDx, 20
- GetHorzDirection, 20
- GetLowermostRec, 20
- GetMaximaPair, 21
- GetMaximaPairEx, 21
- GetNextInAEL, 21
- GetOverlap, 21
- GetOverlapSegment, 21
- GetUnitNormal, 21
- hiRange, 31
- HorzSegmentsOverlap, 22
- InitEdge, 22
- InitEdge2, 22
- InitOptions, 15
- Int128Mul, 22
- IntersectList, 13
- IntersectListSort, 22
- IntersectPoint, 22
- ioPreserveCollinear, 16
- ioReverseSolution, 16
- ioStrictlySimple, 16
- IsHorizontal, 23
- IsIntermediate, 23
- IsMaxima, 23
- IsMinima, 23
- JoinHorz, 23
- JoinList, 13
- JoinType, 16
- jtMiter, 16
- jtRound, 16
- jtSquare, 16
- long64, 13
- loRange, 31
- Minkowski, 23
- MinkowskiDiff, 24
- MinkowskiSum, 24
- NodeType, 16
- ntAny, 16
- ntClosed, 16
- ntOpen, 16
- OpenPathsFromPolyTree, 24
- operator<<, 24, 25
- Orientation, 25
- OutRec1RightOfOutRec2, 25
- ParseFirstLeft, 26
- Path, 14
- Paths, 14
- pftEvenOdd, 16
- pftNegative, 16
- pftNonZero, 16
- pftPositive, 16
- pi, 31
- PointCount, 26
- PointInPolygon, 26
- PointIsVertex, 26
- PointsAreClose, 26
- Poly2ContainsPoly1, 27
- PolyFillType, 16
- PolyNodes, 14
- PolyOutList, 14
- PolyTreeToPaths, 27
- PolyType, 17
- Pt2IsBetweenPt1AndPt3, 27
- ptClip, 17
- ptSubject, 17
- RangeTest, 27
- RemoveEdge, 27
- ReverseHorizontal, 27
- ReversePath, 28
- ReversePaths, 28
- ReversePolyPtLinks, 28
- Round, 28
- SetDx, 28
- SimplifyPolygon, 28
- SimplifyPolygons, 28, 29
- Skip, 31
- SlopesEqual, 29
- SlopesNearCollinear, 29
- SwapIntersectNodes, 30
- SwapPoints, 30
- SwapPolyIndexes, 30
- SwapSides, 30
- TopX, 30
- TranslatePath, 30
- two_pi, 31
- ulong64, 14
- Unassigned, 32
- UpdateOutPtIdxs, 31
- ClipperLib::Clipper, 62
- Clipper, 63
- Execute, 64

- ExecuteInternal, 64
- ReverseSolution, 65
- StrictlySimple, 65
- ClipperLib::ClipperBase, 66
 - ~ClipperBase, 68
 - AddBoundsToLML, 68
 - AddPath, 68
 - AddPaths, 68
 - Clear, 68
 - ClipperBase, 68
 - CreateOutRec, 69
 - DeleteFromAEL, 69
 - DisposeAllOutRecs, 69
 - DisposeLocalMinimaList, 69
 - DisposeOutRec, 69
 - GetBounds, 69
 - InsertScanbeam, 69
 - LocalMinimaPending, 70
 - m_ActiveEdges, 71
 - m_CurrentLM, 71
 - m_edges, 71
 - m_HasOpenPaths, 71
 - m_MinimaList, 71
 - m_PolyOuts, 72
 - m_PreserveCollinear, 72
 - m_Scanbeam, 72
 - m_UseFullRange, 72
 - MinimaList, 67
 - PopLocalMinima, 70
 - PopScanbeam, 70
 - PreserveCollinear, 70
 - ProcessBound, 70
 - Reset, 70
 - ScanbeamList, 67
 - SwapPositionsInAEL, 71
 - UpdateEdgeIntoAEL, 71
- ClipperLib::clipperException, 72
 - ~clipperException, 73
 - clipperException, 73
 - what, 73
- ClipperLib::ClipperOffset, 73
 - ~ClipperOffset, 74
 - AddPath, 74
 - AddPaths, 74
 - ArcTolerance, 75
 - Clear, 74
 - ClipperOffset, 74
 - Execute, 74, 75
 - MiterLimit, 75
- ClipperLib::DoublePoint, 92
 - DoublePoint, 92
 - X, 92
 - Y, 93
- ClipperLib::Int128, 122
 - hi, 125
 - Int128, 122, 123
 - lo, 125
 - operator !=, 123
 - operator >, 123
 - operator >=, 123
 - operator -, 123
 - operator -=, 123
 - operator double, 124
 - operator <, 124
 - operator <=, 124
 - operator +, 124
 - operator +=, 124
 - operator -, 124
 - operator =, 124
 - operator ==, 125
- ClipperLib::IntersectNode, 126
 - Edge1, 126
 - Edge2, 126
 - Pt, 126
- ClipperLib::IntPoint, 127
 - IntPoint, 127
 - operator !=, 127
 - operator ==, 128
 - X, 128
 - Y, 128
- ClipperLib::IntRect, 128
 - bottom, 129
 - left, 129
 - right, 129
 - top, 129
- ClipperLib::Join, 129
 - OffPt, 130
 - OutPt1, 130
 - OutPt2, 130
- ClipperLib::LocalMinimum, 131
 - LeftBound, 131
 - RightBound, 131
 - Y, 131
- ClipperLib::LocMinSorter, 132
 - operator(), 132
- ClipperLib::OutPt, 156
 - Idx, 156
 - Next, 156
 - Prev, 156
 - Pt, 156
- ClipperLib::OutRec, 157
 - BottomPt, 157
 - FirstLeft, 157
 - Idx, 158
 - IsHole, 158
 - IsOpen, 158
 - PolyNd, 158
 - Pts, 158
- ClipperLib::PolyNode, 168
 - ~PolyNode, 170
 - ChildCount, 170
 - Childs, 171
 - Clipper, 170
 - ClipperOffset, 170
 - Contour, 171
 - GetNext, 170

- IsHole, [170](#)
- IsOpen, [170](#)
- Parent, [171](#)
- PolyNode, [169](#)
- ClipperLib::PolyTree, [171](#)
 - ~PolyTree, [172](#)
 - Clear, [172](#)
 - Clipper, [173](#)
 - GetFirst, [172](#)
 - Total, [173](#)
- ClipperLib::TEdge, [189](#)
 - Bot, [189](#)
 - Curr, [190](#)
 - Dx, [190](#)
 - Next, [190](#)
 - NextInAEL, [190](#)
 - NextInLML, [190](#)
 - NextInSEL, [190](#)
 - OutIdx, [190](#)
 - PolyTyp, [190](#)
 - Prev, [191](#)
 - PrevInAEL, [191](#)
 - PrevInSEL, [191](#)
 - Side, [191](#)
 - Top, [191](#)
 - WindCnt, [191](#)
 - WindCnt2, [191](#)
 - WindDelta, [191](#)
- ClipperOffset
 - ClipperLib::ClipperOffset, [74](#)
 - ClipperLib::PolyNode, [170](#)
- ClipType
 - ClipperLib, [14](#)
- Clock
 - utils.hh, [258](#)
- ClosedPathsFromPolyTree
 - ClipperLib, [18](#)
- COLOR
 - Settings, [176](#)
- COLOR_TYPE
 - detection.hh, [231](#)
- computeCenter
 - Object, [150](#)
- computeConversionParameters
 - detection.cc, [218](#)
 - detection.hh, [231](#)
- computeRadius
 - Object, [150](#)
- computeReprojectionErrors
 - calibration.cc, [206](#)
- Configuration2
 - Configuration2< T1 >, [77](#), [78](#)
- Configuration2< T1 >, [75](#)
 - angle, [78](#)
 - Configuration2, [77](#), [78](#)
 - copy, [79](#)
 - distance, [79](#)
 - equal, [79](#)
 - EuDistance, [80](#)
 - MaDistance, [80](#)
 - offset, [81](#)
 - offset_angle, [82](#)
 - offset_x, [82](#)
 - offset_y, [83](#)
 - operator Configuration2< T2 >, [83](#)
 - operator Point2< T2 >, [83](#)
 - operator<<, [86](#)
 - operator=, [83](#)
 - operator==, [84](#)
 - point, [84](#)
 - to_string, [84](#)
 - x, [85](#)
 - y, [85](#), [86](#)
- configure
 - configure.cc, [214](#)
 - configure.hh, [229](#)
- configure.cc
 - acquireImage, [214](#)
 - configure, [214](#)
 - filter, [216](#)
 - on_high_h_thresh_trackbar, [214](#)
 - on_high_s_thresh_trackbar, [214](#)
 - on_high_v_thresh_trackbar, [215](#)
 - on_low_h_thresh_trackbar, [215](#)
 - on_low_s_thresh_trackbar, [215](#)
 - on_low_v_thresh_trackbar, [215](#)
 - show_all_conditions, [215](#)
 - update_trackers, [216](#)
- configure.hh
 - acquireImage, [229](#)
 - configure, [229](#)
 - show_all_conditions, [229](#)
- Contour
 - ClipperLib::PolyNode, [171](#)
- convexHullFile
 - Settings, [186](#)
- copy
 - Angle, [38](#)
 - Configuration2< T1 >, [79](#)
 - Filter, [115](#)
 - Point2< T >, [161](#)
- cos
 - Angle, [39](#)
- countTries
 - dubins.hh, [240](#)
- COUT
 - utils.hh, [257](#)
- createMapp
 - planning.cc, [261](#)
 - planning.hh, [251](#)
- createMapRepresentation
 - Mapp, [139](#)
- CreateOutRec
 - ClipperLib::ClipperBase, [69](#)
- crop_number_section
 - detection.cc, [218](#)

- detection.hh, [232](#)
- ctDifference
 - ClipperLib, [15](#)
- ctIntersection
 - ClipperLib, [15](#)
- ctUnion
 - ClipperLib, [15](#)
- ctXor
 - ClipperLib, [15](#)
- Curr
 - ClipperLib::TEdge, [190](#)
- Curve
 - Curve< T >, [88](#), [89](#)
- Curve< T >, [87](#)
 - begin, [90](#)
 - Curve, [88](#), [89](#)
 - end, [90](#)
 - operator<<, [91](#)
 - P0, [91](#)
 - P1, [91](#)
 - to_string, [91](#)
- CYAN
 - detection.hh, [231](#)
- def_arc_tolerance
 - ClipperLib, [31](#)
- DEG
 - Angle, [37](#)
- DEGTORAD
 - maths.hh, [249](#)
- degToRad
 - Angle, [39](#)
- delay
 - CalSettings, [57](#)
- DeleteFromAEL
 - ClipperLib::ClipperBase, [69](#)
- DELTA
 - dubins.hh, [237](#)
- DETECTION
 - calibration.hh, [223](#)
- detection
 - detection.cc, [218](#)
 - detection.hh, [232](#)
- detection.cc
 - _compare, [218](#)
 - computeConversionParameters, [218](#)
 - crop_number_section, [218](#)
 - detection, [218](#)
 - EPS_CURVE, [217](#)
 - erode_dilation, [218](#)
 - find_contours, [219](#)
 - load_number_template, [219](#)
 - localize, [219](#)
 - MIN_AREA_SIZE, [217](#)
 - number_recognition, [220](#)
 - robotShape, [221](#)
 - save_convex_hull, [220](#)
 - shape_detection, [220](#)
 - templates, [221](#)
- detection.hh
 - BLACK, [231](#)
 - BLUE, [231](#)
 - COLOR_TYPE, [231](#)
 - computeConversionParameters, [231](#)
 - crop_number_section, [232](#)
 - CYAN, [231](#)
 - detection, [232](#)
 - erode_dilation, [232](#)
 - find_contours, [233](#)
 - GREEN, [231](#)
 - load_number_template, [233](#)
 - localize, [233](#)
 - number_recognition, [233](#)
 - RED, [231](#)
 - save_convex_hull, [234](#)
 - shape_detection, [234](#)
- detection_run.cc
 - main, [263](#)
- dimX
 - Mapp, [145](#)
- dimY
 - Mapp, [145](#)
- DInf
 - maths.hh, [247](#)
- Direction
 - ClipperLib, [15](#)
- disp
 - dubins.hh, [238](#)
- DisposeAllOutRecs
 - ClipperLib::ClipperBase, [69](#)
- DisposeLocalMinimalList
 - ClipperLib::ClipperBase, [69](#)
- DisposeOutPts
 - ClipperLib, [18](#)
- DisposeOutRec
 - ClipperLib::ClipperBase, [69](#)
- distance
 - Configuration2< T1 >, [79](#)
 - Point2< T >, [161](#)
 - Tuple< T >, [195](#)
 - unwrapping.cc, [267](#)
- DISTANCE_TYPE
 - maths.hh, [248](#)
- DistanceFromLineSqr
 - ClipperLib, [19](#)
- DistanceSqr
 - ClipperLib, [19](#)
- div
 - Angle, [39](#)
- dLeftToRight
 - ClipperLib, [15](#)
- DoublePoint
 - ClipperLib::DoublePoint, [92](#)
- draw
 - Dubins< T >, [97](#)
 - DubinsArc< T1, T2 >, [106](#)
- draw.hh

- int, 235
- dRightToLeft
 - ClipperLib, 15
- Dubins
 - Dubins< T >, 95, 96
- Dubins< T >, 93
 - check, 96
 - draw, 97
 - Dubins, 95, 96
 - getA1, 97
 - getA2, 98
 - getA3, 98
 - getId, 98
 - getKMax, 98
 - length, 98
 - LRL, 98
 - LSL, 99
 - LSR, 99
 - operator<<, 103
 - rangeSymm, 100
 - RLR, 100
 - RSL, 100
 - RSR, 101
 - scaleFromStandard, 101
 - scaleToStandard, 101
 - shortest_path, 102
 - splitIt, 102
 - to_string, 102
- dubins.hh
 - circline, 238
 - countTries, 240
 - DELTA, 237
 - disp, 238
 - elapsedArcs, 240
 - elapsedBest, 240
 - elapsedCheck, 240
 - elapsedCirc, 240
 - elapsedLRL, 241
 - elapsedLSL, 241
 - elapsedLSR, 241
 - elapsedPrimitives, 241
 - elapsedRLR, 241
 - elapsedRSL, 241
 - elapsedRSR, 241
 - elapsedScale, 241
 - elapsedSet, 242
 - elapsedVar, 242
 - KMAX, 237
 - PIECE_LENGTH, 237
 - PREC, 238
 - reduce_points, 239
 - sinc, 239
 - toBase, 239
- DubinsArc
 - DubinsArc< T1, T2 >, 105
- DubinsArc< T1, T2 >, 103
 - draw, 106
 - DubinsArc, 105
 - getK, 106
 - length, 106
 - operator<<, 107
 - splitIt, 106
 - to_string, 107
- DubinsSet
 - DubinsSet< T >, 109, 110
- DubinsSet< T >, 108
 - DubinsSet, 109, 110
 - find_best, 110
 - getDubins, 111
 - getDubinses, 111
 - getKmax, 111
 - getLength, 111
 - getSize, 111
 - operator<<, 112
 - to_string, 112
- DupOutPt
 - ClipperLib, 19
- DW, 32
 - changeBuffer, 32
 - init, 32
 - map_buffer, 32
 - window, 33
- Dx
 - ClipperLib::TEdge, 190
- E2InsertsBeforeE1
 - ClipperLib, 19
- Edge1
 - ClipperLib::IntersectNode, 126
- Edge2
 - ClipperLib::IntersectNode, 126
- EdgeList
 - ClipperLib, 13
- EdgesAdjacent
 - ClipperLib, 19
- EdgeSide
 - ClipperLib, 15
- elapsedArcs
 - dubins.hh, 240
- elapsedBest
 - dubins.hh, 240
- elapsedCheck
 - dubins.hh, 240
- elapsedCirc
 - dubins.hh, 240
- elapsedLRL
 - dubins.hh, 241
- elapsedLSL
 - dubins.hh, 241
- elapsedLSR
 - dubins.hh, 241
- elapsedPrimitives
 - dubins.hh, 241
- elapsedRLR
 - dubins.hh, 241
- elapsedRSL
 - dubins.hh, 241

- elapsedRSR
 - dubins.hh, [241](#)
- elapsedScale
 - dubins.hh, [241](#)
- elapsedSet
 - dubins.hh, [242](#)
- elapsedVar
 - dubins.hh, [242](#)
- end
 - Curve< T >, [90](#)
 - Tuple< T >, [196](#)
- EndType
 - ClipperLib, [15](#)
- EPS_CURVE
 - detection.cc, [217](#)
- Epsi
 - maths.hh, [247](#)
- equal
 - Angle, [39](#)
 - Configuration2< T1 >, [79](#)
 - maths.hh, [249](#)
 - Point2< T >, [162](#)
 - Tuple< T >, [196](#)
- eraseAll
 - Tuple< T >, [196](#)
- erode_dilation
 - detection.cc, [218](#)
 - detection.hh, [232](#)
- esLeft
 - ClipperLib, [15](#)
- esRight
 - ClipperLib, [15](#)
- etClosedLine
 - ClipperLib, [15](#)
- etClosedPolygon
 - ClipperLib, [15](#)
- etOpenButt
 - ClipperLib, [15](#)
- etOpenRound
 - ClipperLib, [15](#)
- etOpenSquare
 - ClipperLib, [15](#)
- EUCLIDEAN
 - maths.hh, [249](#)
- EuDistance
 - Configuration2< T1 >, [80](#)
 - Point2< T >, [162](#)
 - Tuple< T >, [196](#)
- EXCEPTION_TYPE
 - utils.hh, [258](#)
- ExcludeOp
 - ClipperLib, [19](#)
- Execute
 - ClipperLib::Clipper, [64](#)
 - ClipperLib::ClipperOffset, [74](#), [75](#)
- ExecuteInternal
 - ClipperLib::Clipper, [64](#)
- EXISTS
 - utils.hh, [258](#)
- Filter, [113](#)
 - copy, [115](#)
 - Filter, [114](#)
 - High, [115](#)
 - high_h, [117](#)
 - high_s, [117](#)
 - high_v, [117](#)
 - Low, [115](#)
 - low_h, [117](#)
 - low_s, [117](#)
 - low_v, [117](#)
 - operator vector< int >, [115](#)
 - operator<<, [116](#)
 - operator=, [115](#)
 - to_string, [116](#)
- filter
 - configure.cc, [216](#)
- find
 - Tuple< T >, [197](#)
- find_best
 - DubinsSet< T >, [110](#)
- find_contours
 - detection.cc, [219](#)
 - detection.hh, [233](#)
- find_rect
 - unwrapping.cc, [268](#)
 - unwrapping.hh, [255](#)
- FindNextLocMin
 - ClipperLib, [20](#)
- FirstIsBottomPt
 - ClipperLib, [20](#)
- FirstLeft
 - ClipperLib::OutRec, [157](#)
- fixK1
 - CalSettings, [57](#)
- fixK2
 - CalSettings, [57](#)
- fixK3
 - CalSettings, [57](#)
- fixK4
 - CalSettings, [57](#)
- fixK5
 - CalSettings, [57](#)
- flag
 - CalSettings, [58](#)
- flipVertical
 - CalSettings, [58](#)
- foundLimit
 - Mapp, [146](#)
- frameHeight_px
 - CameraCapture::input_options_t, [121](#)
- frameWidth_px
 - CameraCapture::input_options_t, [121](#)
- FREE
 - map.hh, [244](#)
- GATE

- map.hh, 244
- Gate, 118
 - Gate, 119
 - print, 119
 - toString, 119
- get
 - Angle, 40
 - Tuple< T >, 197
- getA1
 - Dubins< T >, 97
- getA2
 - Dubins< T >, 98
- getA3
 - Dubins< T >, 98
- GetBottomPt
 - ClipperLib, 20
- GetBounds
 - ClipperLib::ClipperBase, 69
- getCenter
 - Object, 150
- getDubins
 - DubinsSet< T >, 111
- getDubinses
 - DubinsSet< T >, 111
- GetDx
 - ClipperLib, 20
- getElapsed
 - CHRONO, 9, 10
- getFiles
 - settings.cc, 266
- GetFirst
 - ClipperLib::PolyTree, 172
- getGateCenter
 - Mapp, 139
- GetHorzDirection
 - ClipperLib, 20
- getId
 - Dubins< T >, 98
- getK
 - DubinsArc< T1, T2 >, 106
- getKMax
 - Dubins< T >, 98
- getKmax
 - DubinsSet< T >, 111
- getLength
 - DubinsSet< T >, 111
- GetLowermostRec
 - ClipperLib, 20
- GetMaximaPair
 - ClipperLib, 21
- GetMaximaPairEx
 - ClipperLib, 21
- GetNext
 - ClipperLib::PolyNode, 170
- GetNextInAEL
 - ClipperLib, 21
- GetOverlap
 - ClipperLib, 21
- GetOverlapSegment
 - ClipperLib, 21
- getPoints
 - Object, 151
- getPointType
 - Mapp, 139
- getRadius
 - Object, 151
- getSize
 - DubinsSet< T >, 111
- getTemplates
 - Settings, 178, 179
- getTimeS
 - timeutils, 33
- getType
 - Angle, 40
 - CHRONO, 10
- getTypeName
 - Angle, 40
- GetUnitNormal
 - ClipperLib, 21
- getValue
 - Victim, 203
- getVictimCenters
 - Mapp, 140
- goodInput
 - CalSettings, 58
- grab
 - CameraCapture, 61
- greater
 - Angle, 40
- GREEN
 - detection.hh, 231
 - Settings, 176
- greenMask
 - Settings, 186
- hi
 - ClipperLib::Int128, 125
- High
 - Filter, 115
- high_h
 - Filter, 117
- high_s
 - Filter, 117
- high_v
 - Filter, 117
- hiRange
 - ClipperLib, 31
- HORIZONTAL
 - clipper.cc, 213
- HorzSegmentsOverlap
 - ClipperLib, 22
- Idx
 - ClipperLib::OutPt, 156
 - ClipperLib::OutRec, 158
- IMAGE_LIST
 - CalSettings, 53

- imageAddPoint
 - Mapp, [140](#)
- imageAddPoints
 - Mapp, [140](#)
- imageAddSegment
 - Mapp, [141](#)
- imageAddSegments
 - Mapp, [141](#)
- imageList
 - CalSettings, [58](#)
- INFO
 - utils.hh, [257](#)
- init
 - DW, [32](#)
- InitEdge
 - ClipperLib, [22](#)
- InitEdge2
 - ClipperLib, [22](#)
- InitOptions
 - ClipperLib, [15](#)
- input
 - CalSettings, [58](#)
- input_options_t
 - CameraCapture::input_options_t, [121](#)
- inputCapture
 - CalSettings, [58](#)
- InputType
 - CalSettings, [53](#)
- inputType
 - CalSettings, [58](#)
- InsertScanbeam
 - ClipperLib::ClipperBase, [69](#)
- insidePoly
 - Object, [151](#)
- insidePolyApprox
 - Object, [152](#)
- int
 - draw.hh, [235](#)
- Int128
 - ClipperLib::Int128, [122](#), [123](#)
- Int128Mul
 - ClipperLib, [22](#)
- IntersectList
 - ClipperLib, [13](#)
- IntersectListSort
 - ClipperLib, [22](#)
- IntersectPoint
 - ClipperLib, [22](#)
- IntPoint
 - ClipperLib::IntPoint, [127](#)
- intrinsicCalibrationFile
 - Settings, [187](#)
- INVALID
 - Angle, [37](#)
 - CalSettings, [53](#)
- ioPreserveCollinear
 - ClipperLib, [16](#)
- ioReverseSolution
 - ClipperLib, [16](#)
- ioStrictlySimple
 - ClipperLib, [16](#)
- isAlive
 - CameraCapture, [61](#)
- IsHole
 - ClipperLib::OutRec, [158](#)
 - ClipperLib::PolyNode, [170](#)
- IsHorizontal
 - ClipperLib, [23](#)
- IsIntermediate
 - ClipperLib, [23](#)
- isListOfImages
 - CalSettings, [53](#)
- IsMaxima
 - ClipperLib, [23](#)
- IsMinima
 - ClipperLib, [23](#)
- IsOpen
 - ClipperLib::OutRec, [158](#)
 - ClipperLib::PolyNode, [170](#)
- isOpened
 - CameraCapture, [61](#)
- JoinHorz
 - ClipperLib, [23](#)
- JoinList
 - ClipperLib, [13](#)
- JoinType
 - ClipperLib, [16](#)
- jtMiter
 - ClipperLib, [16](#)
- jtRound
 - ClipperLib, [16](#)
- jtSquare
 - ClipperLib, [16](#)
- kernelSide
 - Settings, [187](#)
- KMAX
 - dubins.hh, [237](#)
- left
 - ClipperLib::IntRect, [129](#)
- LeftBound
 - ClipperLib::LocalMinimum, [131](#)
- length
 - Dubins< T >, [98](#)
 - DubinsArc< T1, T2 >, [106](#)
- lengthX
 - Mapp, [146](#)
- lengthY
 - Mapp, [146](#)
- less
 - Angle, [41](#)
- lo
 - ClipperLib::Int128, [125](#)
- load_number_template
 - detection.cc, [219](#)

- detection.hh, [233](#)
- loadCoefficients
 - CameraCapture, [62](#)
 - unwrapping.cc, [268](#)
 - unwrapping.hh, [255](#)
- loadVP
 - planning.cc, [261](#)
 - planning.hh, [252](#)
- loadVVP
 - planning.cc, [261](#)
 - planning.hh, [252](#)
- localize
 - detection.cc, [219](#)
 - detection.hh, [233](#)
- LocalMinimaPending
 - ClipperLib::ClipperBase, [70](#)
- long64
 - ClipperLib, [13](#)
- loRange
 - ClipperLib, [31](#)
- Low
 - Filter, [115](#)
- low_h
 - Filter, [117](#)
- low_s
 - Filter, [117](#)
- low_v
 - Filter, [117](#)
- LRL
 - Dubins< T >, [98](#)
- LSL
 - Dubins< T >, [99](#)
- LSR
 - Dubins< T >, [99](#)
- m_ActiveEdges
 - ClipperLib::ClipperBase, [71](#)
- m_CurrentLM
 - ClipperLib::ClipperBase, [71](#)
- m_edges
 - ClipperLib::ClipperBase, [71](#)
- m_HasOpenPaths
 - ClipperLib::ClipperBase, [71](#)
- m_MinimaList
 - ClipperLib::ClipperBase, [71](#)
- m_PolyOuts
 - ClipperLib::ClipperBase, [72](#)
- m_PreserveCollinear
 - ClipperLib::ClipperBase, [72](#)
- m_Scanbeam
 - ClipperLib::ClipperBase, [72](#)
- m_UseFullRange
 - ClipperLib::ClipperBase, [72](#)
- MaDistance
 - Configuration2< T1 >, [80](#)
 - Point2< T >, [162](#)
 - Tuple< T >, [197](#)
- main
 - calibration_run.cc, [262](#)
 - detection_run.cc, [263](#)
 - main.cc, [263](#)
 - planning_run.cc, [264](#)
 - unwrapping_run.cc, [265](#)
- main.cc
 - main, [263](#)
 - sett, [264](#)
- MANHATTAN
 - maths.hh, [249](#)
- map
 - Mapp, [146](#)
- map.hh
 - BODA, [244](#)
 - FREE, [244](#)
 - GATE, [244](#)
 - OBJ_TYPE, [244](#)
 - OBST, [244](#)
 - VICT, [244](#)
- map_buffer
 - DW, [32](#)
- Mapp, [132](#)
 - addObject, [136](#)
 - addObjects, [136](#), [137](#)
 - baseDistance, [145](#)
 - borderSize, [145](#)
 - borderSizeDefault, [145](#)
 - cellsFromSegment, [137](#)
 - cellSize, [145](#)
 - checkSegment, [138](#)
 - checkSegmentCollisionWithType, [138](#)
 - createMapRepresentation, [139](#)
 - dimX, [145](#)
 - dimY, [145](#)
 - foundLimit, [146](#)
 - getGateCenter, [139](#)
 - getPointType, [139](#)
 - getVictimCenters, [140](#)
 - imageAddPoint, [140](#)
 - imageAddPoints, [140](#)
 - imageAddSegment, [141](#)
 - imageAddSegments, [141](#)
 - lengthX, [146](#)
 - lengthY, [146](#)
 - map, [146](#)
 - Mapp, [134](#)
 - matrixToString, [141](#)
 - minPathNPoints, [141](#)
 - minPathTwoPoints, [142](#)
 - minPathTwoPointsInternal, [142](#)
 - nPoints, [146](#)
 - offsetValue, [146](#)
 - pixX, [146](#)
 - pixY, [146](#)
 - printDimensions, [143](#)
 - printMap, [143](#)
 - range, [147](#)
 - resetDistanceMap, [143](#)
 - sampleNPoints, [144](#)

- samplePointsEachNCells, 144
- vGates, 147
- vObstacles, 147
- vVictims, 147
- maps
 - Settings, 179, 181, 182
- mapsFolder
 - Settings, 187
- mapsNames
 - Settings, 187
- mapsUnNames
 - Settings, 187
- maths.hh
 - A_180, 246
 - A_2PI, 246
 - A_360, 246
 - A_90, 246
 - A_DEG_NULL, 246
 - A_PI, 247
 - A_PI2, 247
 - A_RAD_NULL, 247
 - DEGTORAD, 249
 - DInf, 247
 - DISTANCE_TYPE, 248
 - Epsi, 247
 - equal, 249
 - EUCLIDEAN, 249
 - MANHATTAN, 249
 - pow2, 249
 - RADTODEG, 249
 - tupleConstIter, 247
 - tupleIter, 247
- matrixToString
 - Mapp, 141
- MIN_AREA_SIZE
 - detection.cc, 217
- MinimalList
 - ClipperLib::ClipperBase, 67
- Minkowski
 - ClipperLib, 23
- MinkowskiDiff
 - ClipperLib, 24
- MinkowskiSum
 - ClipperLib, 24
- minPathNPoints
 - Mapp, 141
- minPathTwoPoints
 - Mapp, 142
- minPathTwoPointsInternal
 - Mapp, 142
- MiterLimit
 - ClipperLib::ClipperOffset, 75
- MSEC
 - CHRONO, 9
- mul
 - Angle, 41
 - Tuple< T >, 198
- MUSEC
 - CHRONO, 9
- my_imshow
 - utils.cc, 269
 - utils.hh, 258
- MyException
 - MyException< T >, 148
- MyException< T >, 147
 - a, 148
 - b, 148
 - MyException, 148
 - type, 148
 - what, 148
- mywaitkey
 - utils.cc, 270
 - utils.hh, 259
- NAME
 - utils.hh, 258
- nameCamera
 - CameraCapture::input_options_t, 121
- NEAR_ZERO
 - clipper.cc, 213
- Next
 - ClipperLib::OutPt, 156
 - ClipperLib::TEdge, 190
- nextImage
 - CalSettings, 54
- NextInAEL
 - ClipperLib::TEdge, 190
- NextInLML
 - ClipperLib::TEdge, 190
- NextInSEL
 - ClipperLib::TEdge, 190
- NodeType
 - ClipperLib, 16
- normalize
 - Angle, 41
- NOT_EXISTING
 - CalSettings, 53
- nPoints
 - Mapp, 146
 - Object, 152
- NPOS
 - settings.cc, 265
- nrFrames
 - CalSettings, 59
- NSEC
 - CHRONO, 9
- ntAny
 - ClipperLib, 16
- ntClosed
 - ClipperLib, 16
- ntOpen
 - ClipperLib, 16
- number_recognition
 - detection.cc, 220
 - detection.hh, 233
- OBJ_TYPE

- map.hh, 244
- Object, 149
 - center, 153
 - computeCenter, 150
 - computeRadius, 150
 - getCenter, 150
 - getPoints, 151
 - getRadius, 151
 - insidePoly, 151
 - insidePolyApprox, 152
 - nPoints, 152
 - offsetting, 152
 - points, 153
 - radius, 153
 - size, 153
 - toString, 153
- OBST
 - map.hh, 244
- Obstacle, 154
 - Obstacle, 155
 - print, 155
 - toString, 155
- OffPt
 - ClipperLib::Join, 130
- offset
 - Configuration2< T1 >, 81
 - Point2< T >, 163, 164
- offset_angle
 - Configuration2< T1 >, 82
- offset_x
 - Configuration2< T1 >, 82
 - Point2< T >, 164
- offset_y
 - Configuration2< T1 >, 83
 - Point2< T >, 164
- offsetting
 - Object, 152
- offsetValue
 - Mapp, 146
- on_high_h_thresh_trackbar
 - configure.cc, 214
- on_high_s_thresh_trackbar
 - configure.cc, 214
- on_high_v_thresh_trackbar
 - configure.cc, 215
- on_low_h_thresh_trackbar
 - configure.cc, 215
- on_low_s_thresh_trackbar
 - configure.cc, 215
- on_low_v_thresh_trackbar
 - configure.cc, 215
- OpenPathsFromPolyTree
 - ClipperLib, 24
- operator !=
 - ClipperLib::Int128, 123
- operator >
 - ClipperLib::Int128, 123
- operator >=
 - ClipperLib::Int128, 123
- operator *
 - Angle, 42
 - Tuple< T >, 198
- operator *=
 - Angle, 42
 - Tuple< T >, 198
- operator -
 - ClipperLib::Int128, 123
- operator -=
 - ClipperLib::Int128, 123
- operator Configuration2< T2 >
 - Configuration2< T1 >, 83
- operator cv::Point
 - Point2< T >, 165
- operator double
 - Angle, 42
 - ClipperLib::Int128, 124
- operator float
 - Angle, 43
- operator int
 - Angle, 43
- operator long
 - Angle, 43
- operator Point2< T2 >
 - Configuration2< T1 >, 83
- operator std::string
 - Tuple< T >, 198
- operator vector< int >
 - Filter, 115
- operator vector< T1 >
 - Tuple< T >, 198
- operator !=
 - Angle, 43
 - ClipperLib::IntPoint, 127
 - Point2< T >, 165
- operator<
 - Angle, 46
 - ClipperLib::Int128, 124
 - Point2< T >, 165
- operator<<
 - Angle, 50
 - ClipperLib, 24, 25
 - Configuration2< T1 >, 86
 - Curve< T >, 91
 - Dubins< T >, 103
 - DubinsArc< T1, T2 >, 107
 - DubinsSet< T >, 112
 - Filter, 116
 - Point2< T >, 168
 - Settings, 185
 - Tuple< T >, 201
- operator<=
 - Angle, 46
 - ClipperLib::Int128, 124
- operator>
 - Angle, 47
- operator>=

- Angle, [47](#)
- operator()
 - ClipperLib::LocMinSorter, [132](#)
- operator+
 - Angle, [44](#)
 - ClipperLib::Int128, [124](#)
 - Tuple< T >, [199](#)
- operator+=
 - Angle, [44](#)
 - ClipperLib::Int128, [124](#)
 - Tuple< T >, [199](#)
- operator-
 - Angle, [45](#)
 - ClipperLib::Int128, [124](#)
- operator-=
 - Angle, [45](#)
- operator/
 - Angle, [45](#)
- operator/=
 - Angle, [46](#)
- operator=
 - Angle, [46](#), [47](#)
 - ClipperLib::Int128, [124](#)
 - Configuration2< T1 >, [83](#)
 - Filter, [115](#)
 - Point2< T >, [165](#)
- operator==
 - Angle, [47](#)
 - ClipperLib::Int128, [125](#)
 - ClipperLib::IntPoint, [128](#)
 - Configuration2< T1 >, [84](#)
 - Point2< T >, [166](#)
 - Tuple< T >, [199](#)
- operator[]
 - Tuple< T >, [199](#)
- Orientation
 - ClipperLib, [25](#)
- OutIdx
 - ClipperLib::TEdge, [190](#)
- OutPt1
 - ClipperLib::Join, [130](#)
- OutPt2
 - ClipperLib::Join, [130](#)
- outputFileName
 - CalSettings, [59](#)
- OutRec1RightOfOutRec2
 - ClipperLib, [25](#)
- P0
 - Curve< T >, [91](#)
- P1
 - Curve< T >, [91](#)
- Parent
 - ClipperLib::PolyNode, [171](#)
- ParseFirstLeft
 - ClipperLib, [26](#)
- Path
 - ClipperLib, [14](#)
- Paths
 - ClipperLib, [14](#)
- Pattern
 - CalSettings, [53](#)
- pftEvenOdd
 - ClipperLib, [16](#)
- pftNegative
 - ClipperLib, [16](#)
- pftNonZero
 - ClipperLib, [16](#)
- pftPositive
 - ClipperLib, [16](#)
- pi
 - ClipperLib, [31](#)
- PIECE_LENGTH
 - dubins.hh, [237](#)
- pixX
 - Mapp, [146](#)
- pixY
 - Mapp, [146](#)
- planning
 - planning.cc, [261](#)
 - planning.hh, [252](#)
- planning.cc
 - createMapp, [261](#)
 - loadVP, [261](#)
 - loadVVP, [261](#)
 - planning, [261](#)
- planning.hh
 - createMapp, [251](#)
 - loadVP, [252](#)
 - loadVVP, [252](#)
 - planning, [252](#)
- planning_run.cc
 - main, [264](#)
- point
 - Configuration2< T1 >, [84](#)
- Point2
 - Point2< T >, [160](#), [161](#)
- Point2< T >, [158](#)
 - copy, [161](#)
 - distance, [161](#)
 - equal, [162](#)
 - EuDistance, [162](#)
 - MaDistance, [162](#)
 - offset, [163](#), [164](#)
 - offset_x, [164](#)
 - offset_y, [164](#)
 - operator cv::Point, [165](#)
 - operator!=, [165](#)
 - operator<, [165](#)
 - operator<=, [168](#)
 - operator=, [165](#)
 - operator==, [166](#)
 - Point2, [160](#), [161](#)
 - th, [166](#)
 - to_string, [166](#)
 - x, [166](#), [167](#)
 - y, [167](#)

- PointCount
 - ClipperLib, 26
- PointInPolygon
 - ClipperLib, 26
- PointIsVertex
 - ClipperLib, 26
- points
 - Object, 153
- PointsAreClose
 - ClipperLib, 26
- Poly2ContainsPoly1
 - ClipperLib, 27
- PolyFillType
 - ClipperLib, 16
- PolyNd
 - ClipperLib::OutRec, 158
- PolyNode
 - ClipperLib::PolyNode, 169
- PolyNodes
 - ClipperLib, 14
- PolyOutList
 - ClipperLib, 14
- PolyTreeToPaths
 - ClipperLib, 27
- PolyType
 - ClipperLib::TEdge, 190
- PolyType
 - ClipperLib, 17
- PopLocalMinima
 - ClipperLib::ClipperBase, 70
- PopScanbeam
 - ClipperLib::ClipperBase, 70
- pow2
 - maths.hh, 249
- PREC
 - dubins.hh, 238
- PreserveCollinear
 - ClipperLib::ClipperBase, 70
- Prev
 - ClipperLib::OutPt, 156
 - ClipperLib::TEdge, 191
- PrevInAEL
 - ClipperLib::TEdge, 191
- PrevInSEL
 - ClipperLib::TEdge, 191
- print
 - Gate, 119
 - Obstacle, 155
 - Victim, 203
- printDimensions
 - Mapp, 143
- printMap
 - Mapp, 143
- ProcessBound
 - ClipperLib::ClipperBase, 70
- Pt
 - ClipperLib::IntersectNode, 126
 - ClipperLib::OutPt, 156
- Pt2IsBetweenPt1AndPt3
 - ClipperLib, 27
- ptClip
 - ClipperLib, 17
- Pts
 - ClipperLib::OutRec, 158
- ptSubject
 - ClipperLib, 17
- RAD
 - Angle, 37
- radius
 - Object, 153
- RADTODEG
 - maths.hh, 249
- radToDeg
 - Angle, 47
- range
 - Mapp, 147
- rangeSymm
 - Dubins< T >, 100
- RangeTest
 - ClipperLib, 27
- read
 - calibration.cc, 207
 - CalSettings, 54
- readFromFile
 - Settings, 182
- readStringList
 - CalSettings, 54
- RED
 - detection.hh, 231
 - Settings, 176
- redMask
 - Settings, 187
- reduce_points
 - dubins.hh, 239
- remove
 - Tuple< T >, 200
- RemoveEdge
 - ClipperLib, 27
- Reset
 - ClipperLib::ClipperBase, 70
- resetDistanceMap
 - Mapp, 143
- ReverseHorizontal
 - ClipperLib, 27
- ReversePath
 - ClipperLib, 28
- ReversePaths
 - ClipperLib, 28
- ReversePolyPtLinks
 - ClipperLib, 28
- ReverseSolution
 - ClipperLib::Clipper, 65
- right
 - ClipperLib::IntRect, 129
- RightBound
 - ClipperLib::LocalMinimum, 131

- RLR
 - Dubins< T >, 100
- ROBOT
 - Settings, 176
- robotMask
 - Settings, 188
- robotShape
 - detection.cc, 221
- Round
 - ClipperLib, 28
- RSL
 - Dubins< T >, 100
- RSR
 - Dubins< T >, 101
- runCalibration
 - calibration.cc, 207
- runCalibrationAndSave
 - calibration.cc, 208
 - calibration.hh, 223
- sampleNPoints
 - Mapp, 144
- samplePointsEachNCells
 - Mapp, 144
- save
 - Settings, 182
- save_convex_hull
 - detection.cc, 220
 - detection.hh, 234
- saveCameraParams
 - calibration.cc, 208
- scaleFromStandard
 - Dubins< T >, 101
- scaleToStandard
 - Dubins< T >, 101
- ScanbeamList
 - ClipperLib::ClipperBase, 67
- SDEBUG
 - camera_capture.cc, 210
- SEC
 - CHRONO, 9
- set
 - Angle, 48
 - Tuple< T >, 200
- SetDx
 - ClipperLib, 28
- sett
 - calibration.hh, 224
 - main.cc, 264
 - settings.hh, 253
- Settings, 173
 - ~Settings, 177
 - addUnMap, 177
 - BLACK, 176
 - blackMask, 186
 - BLUE, 176
 - blueMask, 186
 - calibrationFile, 186
 - changeMask, 177, 178
 - clean, 178
 - cleanAndRead, 178
 - COLOR, 176
 - convexHullFile, 186
 - getTemplates, 178, 179
 - GREEN, 176
 - greenMask, 186
 - intrinsicCalibrationFile, 187
 - kernelSide, 187
 - maps, 179, 181, 182
 - mapsFolder, 187
 - mapsNames, 187
 - mapsUnNames, 187
 - operator<<, 185
 - readFromFile, 182
 - RED, 176
 - redMask, 187
 - ROBOT, 176
 - robotMask, 188
 - save, 182
 - Settings, 176
 - templates, 188
 - templatesFolder, 188
 - to_string, 183
 - unMaps, 183–185
 - victimMask, 188
 - VICTIMS, 176
 - WHITE, 176
 - whiteMask, 188
 - writeToFile, 185
- settings.cc
 - getFiles, 266
 - NPOS, 265
 - vecToFile, 266
- settings.hh
 - sett, 253
- setType
 - Angle, 48
- setValue
 - Victim, 204
- shape_detection
 - detection.cc, 220
 - detection.hh, 234
- shortest_path
 - Dubins< T >, 102
- show_all_conditions
 - configure.cc, 215
 - configure.hh, 229
- showUndistorted
 - CalSettings, 59
- Side
 - ClipperLib::TEdge, 191
- SimplifyPolygon
 - ClipperLib, 28
- SimplifyPolygons
 - ClipperLib, 28, 29
- sin
 - Angle, 48

- sinc
 - dubins.hh, [239](#)
- SIZE
 - utils.hh, [258](#)
- size
 - Object, [153](#)
 - Tuple< T >, [200](#)
- Skip
 - ClipperLib, [31](#)
- SlopesEqual
 - ClipperLib, [29](#)
- SlopesNearCollinear
 - ClipperLib, [29](#)
- splitIt
 - Dubins< T >, [102](#)
 - DubinsArc< T1, T2 >, [106](#)
- squareSize
 - CalSettings, [59](#)
- src/calibration.cc, [205](#)
- src/camera_capture.cc, [209](#)
- src/clipper.cc, [210](#)
- src/configure.cc, [213](#)
- src/detection.cc, [216](#)
- src/dubins.cc, [221](#)
- src/include/calibration.hh, [221](#)
- src/include/camera_capture.hh, [224](#)
- src/include/clipper.hh, [225](#)
- src/include/configure.hh, [228](#)
- src/include/detection.hh, [230](#)
- src/include/draw.hh, [234](#)
- src/include/dubins.hh, [236](#)
- src/include/filter.hh, [242](#)
- src/include/map.hh, [243](#)
- src/include/maths.hh, [244](#)
- src/include/objects.hh, [250](#)
- src/include/planning.hh, [251](#)
- src/include/settings.hh, [253](#)
- src/include/unwrapping.hh, [254](#)
- src/include/utils.hh, [256](#)
- src/map.cc, [259](#)
- src/maths.cc, [260](#)
- src/objects.cc, [260](#)
- src/planning.cc, [260](#)
- src/run/calibration_run.cc, [262](#)
- src/run/detection_run.cc, [262](#)
- src/run/main.cc, [263](#)
- src/run/planning_run.cc, [264](#)
- src/run/unwrapping_run.cc, [264](#)
- src/settings.cc, [265](#)
- src/unwrapping.cc, [266](#)
- src/utils.cc, [269](#)
- startCamera
 - CameraCapture, [62](#)
- StrictlySimple
 - ClipperLib::Clipper, [65](#)
- sub
 - Angle, [49](#)
- sum
 - Tuple< T >, [201](#)
- SwapIntersectNodes
 - ClipperLib, [30](#)
- SwapPoints
 - ClipperLib, [30](#)
- SwapPolyIndexes
 - ClipperLib, [30](#)
- SwapPositionsInAEL
 - ClipperLib::ClipperBase, [71](#)
- SwapSides
 - ClipperLib, [30](#)
- tan
 - Angle, [49](#)
- templates
 - detection.cc, [221](#)
 - Settings, [188](#)
- templatesFolder
 - Settings, [188](#)
- th
 - Point2< T >, [166](#)
- TIME_TYPE
 - CHRONO, [9](#)
- timespecDiff
 - timeutils, [33](#)
- timeutils, [33](#)
 - getTimeS, [33](#)
 - timespecDiff, [33](#)
- to_std_string
 - Tuple< T >, [201](#)
- to_string
 - Angle, [49](#)
 - Configuration2< T1 >, [84](#)
 - Curve< T >, [91](#)
 - Dubins< T >, [102](#)
 - DubinsArc< T1, T2 >, [107](#)
 - DubinsSet< T >, [112](#)
 - Filter, [116](#)
 - Point2< T >, [166](#)
 - Settings, [183](#)
 - Tuple< T >, [201](#)
- toBase
 - dubins.hh, [239](#)
- toDeg
 - Angle, [50](#)
- TOLERANCE
 - clipper.cc, [213](#)
- Top
 - ClipperLib::TEdge, [191](#)
- top
 - ClipperLib::IntRect, [129](#)
- TopX
 - ClipperLib, [30](#)
- toRad
 - Angle, [50](#)
- toString
 - Gate, [119](#)
 - Object, [153](#)
 - Obstacle, [155](#)

- Victim, [204](#)
- Total
 - ClipperLib::PolyTree, [173](#)
- TranslatePath
 - ClipperLib, [30](#)
- Tuple
 - Tuple< T >, [193](#), [194](#)
- Tuple< T >, [192](#)
 - add, [194](#)
 - addIfNot, [194](#)
 - ahead, [194](#)
 - begin, [195](#)
 - distance, [195](#)
 - end, [196](#)
 - equal, [196](#)
 - eraseAll, [196](#)
 - EuDistance, [196](#)
 - find, [197](#)
 - get, [197](#)
 - MaDistance, [197](#)
 - mul, [198](#)
 - operator *, [198](#)
 - operator ==, [198](#)
 - operator std::string, [198](#)
 - operator vector< T1 >, [198](#)
 - operator<<, [201](#)
 - operator+, [199](#)
 - operator+=, [199](#)
 - operator==, [199](#)
 - operator[], [199](#)
 - remove, [200](#)
 - set, [200](#)
 - size, [200](#)
 - sum, [201](#)
 - to_std_string, [201](#)
 - to_string, [201](#)
 - Tuple, [193](#), [194](#)
- tupleConstIter
 - maths.hh, [247](#)
- tupleIter
 - maths.hh, [247](#)
- two_pi
 - ClipperLib, [31](#)
- type
 - MyException< T >, [148](#)
- ulong64
 - ClipperLib, [14](#)
- Unassigned
 - ClipperLib, [32](#)
- unMaps
 - Settings, [183](#)–[185](#)
- unwrapping
 - unwrapping.cc, [268](#)
 - unwrapping.hh, [255](#)
- unwrapping.cc
 - AREA_MIN, [267](#)
 - AREA_RATIO, [267](#)
 - distance, [267](#)
 - find_rect, [268](#)
 - loadCoefficients, [268](#)
 - unwrapping, [268](#)
- unwrapping.hh
 - find_rect, [255](#)
 - loadCoefficients, [255](#)
 - unwrapping, [255](#)
- unwrapping_run.cc
 - main, [265](#)
- update_trackers
 - configure.cc, [216](#)
- UpdateEdgeIntoAEL
 - ClipperLib::ClipperBase, [71](#)
- UpdateOutPtIdxs
 - ClipperLib, [31](#)
- use_lines
 - clipper.hh, [228](#)
- useFisheye
 - CalSettings, [59](#)
- utils.cc
 - my_imshow, [269](#)
 - mywaitkey, [270](#)
- utils.hh
 - Clock, [258](#)
 - COUT, [257](#)
 - EXCEPTION_TYPE, [258](#)
 - EXISTS, [258](#)
 - INFO, [257](#)
 - my_imshow, [258](#)
 - mywaitkey, [259](#)
 - NAME, [258](#)
 - SIZE, [258](#)
- validate
 - CalSettings, [55](#)
- value
 - Victim, [204](#)
- vecToFile
 - settings.cc, [266](#)
- vGates
 - Mapp, [147](#)
- VICT
 - map.hh, [244](#)
- Victim, [202](#)
 - getValue, [203](#)
 - print, [203](#)
 - setValue, [204](#)
 - toString, [204](#)
 - value, [204](#)
 - Victim, [203](#)
- victimMask
 - Settings, [188](#)
- VICTIMS
 - Settings, [176](#)
- vObstacles
 - Mapp, [147](#)
- vVictims
 - Mapp, [147](#)

- what
 - ClipperLib::clipperException, [73](#)
 - MyException< T >, [148](#)
- WHITE
 - Settings, [176](#)
- whiteMask
 - Settings, [188](#)
- WindCnt
 - ClipperLib::TEdge, [191](#)
- WindCnt2
 - ClipperLib::TEdge, [191](#)
- WindDelta
 - ClipperLib::TEdge, [191](#)
- window
 - DW, [33](#)
- write
 - CalSettings, [55](#)
- writeExtrinsics
 - CalSettings, [59](#)
- writePoints
 - CalSettings, [60](#)
- writeToFile
 - Settings, [185](#)
- X
 - ClipperLib::DoublePoint, [92](#)
 - ClipperLib::IntPoint, [128](#)
- x
 - Configuration2< T1 >, [85](#)
 - Point2< T >, [166](#), [167](#)
- Y
 - ClipperLib::DoublePoint, [93](#)
 - ClipperLib::IntPoint, [128](#)
 - ClipperLib::LocalMinimum, [131](#)
- y
 - Configuration2< T1 >, [85](#), [86](#)
 - Point2< T >, [167](#)