

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 ClipperLib Namespace Reference	9
5.1.1 Typedef Documentation	12
5.1.1.1 clnt	12
5.1.1.2 EdgeList	12
5.1.1.3 IntersectList	12
5.1.1.4 JoinList	12
5.1.1.5 long64	12
5.1.1.6 Path	12
5.1.1.7 Paths	12
5.1.1.8 PolyNodes	13
5.1.1.9 PolyOutList	13
5.1.1.10 ulong64	13
5.1.2 Enumeration Type Documentation	13
5.1.2.1 ClipType	13
5.1.2.2 Direction	13
5.1.2.3 EdgeSide	14
5.1.2.4 EndType	14
5.1.2.5 InitOptions	14
5.1.2.6 JoinType	14
5.1.2.7 NodeType	15
5.1.2.8 PolyFillType	15
5.1.2.9 PolyType	15
5.1.3 Function Documentation	15
5.1.3.1 Abs()	15
5.1.3.2 AddPolyNodeToPaths()	16
5.1.3.3 Area() [1/3]	16
5.1.3.4 Area() [2/3]	16
5.1.3.5 Area() [3/3]	16
5.1.3.6 CleanPolygon() [1/2]	16
5.1.3.7 CleanPolygon() [2/2]	16
5.1.3.8 CleanPolygons() [1/2]	17

5.1.3.9 CleanPolygons() [2/2]	17
5.1.3.10 ClosedPathsFromPolyTree()	17
5.1.3.11 DisposeOutPts()	17
5.1.3.12 DistanceFromLineSqrd()	17
5.1.3.13 DistanceSqrd()	17
5.1.3.14 DupOutPt()	18
5.1.3.15 E2InsertsBeforeE1()	18
5.1.3.16 EdgesAdjacent()	18
5.1.3.17 ExcludeOp()	18
5.1.3.18 FindNextLocMin()	18
5.1.3.19 FirstIsBottomPt()	18
5.1.3.20 GetBottomPt()	19
5.1.3.21 GetDx()	19
5.1.3.22 GetHorzDirection()	19
5.1.3.23 GetLowermostRec()	19
5.1.3.24 GetMaximaPair()	19
5.1.3.25 GetMaximaPairEx()	19
5.1.3.26 GetNextInAEL()	20
5.1.3.27 GetOverlap()	20
5.1.3.28 GetOverlapSegment()	20
5.1.3.29 GetUnitNormal()	20
5.1.3.30 HorzSegmentsOverlap()	20
5.1.3.31 InitEdge()	21
5.1.3.32 InitEdge2()	21
5.1.3.33 Int128Mul()	21
5.1.3.34 IntersectListSort()	21
5.1.3.35 IntersectPoint()	21
5.1.3.36 IsHorizontal()	21
5.1.3.37 IsIntermediate()	22
5.1.3.38 IsMaxima()	22
5.1.3.39 IsMinima()	22
5.1.3.40 JoinHorz()	22
5.1.3.41 Minkowski()	22
5.1.3.42 MinkowskiDiff()	23
5.1.3.43 MinkowskiSum() [1/2]	23
5.1.3.44 MinkowskiSum() [2/2]	23
5.1.3.45 OpenPathsFromPolyTree()	23
5.1.3.46 operator<<() [1/5]	23
5.1.3.47 operator<<() [2/5]	23
5.1.3.48 operator<<() [3/5]	24
5.1.3.49 operator<<() [4/5]	24
5.1.3.50 operator<<() [5/5]	24

5.1.3.51 Orientation()	24
5.1.3.52 OutRec1RightOfOutRec2()	24
5.1.3.53 ParseFirstLeft()	24
5.1.3.54 PointCount()	25
5.1.3.55 PointInPolygon() [1/2]	25
5.1.3.56 PointInPolygon() [2/2]	25
5.1.3.57 PointIsVertex()	25
5.1.3.58 PointsAreClose()	25
5.1.3.59 Poly2ContainsPoly1()	25
5.1.3.60 PolyTreeToPaths()	26
5.1.3.61 Pt2IsBetweenPt1AndPt3()	26
5.1.3.62 RangeTest()	26
5.1.3.63 RemoveEdge()	26
5.1.3.64 ReverseHorizontal()	26
5.1.3.65 ReversePath()	26
5.1.3.66 ReversePaths()	27
5.1.3.67 ReversePolyPtLinks()	27
5.1.3.68 Round()	27
5.1.3.69 SetDx()	27
5.1.3.70 SimplifyPolygon()	27
5.1.3.71 SimplifyPolygons() [1/2]	27
5.1.3.72 SimplifyPolygons() [2/2]	28
5.1.3.73 SlopesEqual() [1/3]	28
5.1.3.74 SlopesEqual() [2/3]	28
5.1.3.75 SlopesEqual() [3/3]	28
5.1.3.76 SlopesNearCollinear()	28
5.1.3.77 SwapIntersectNodes()	29
5.1.3.78 SwapPoints()	29
5.1.3.79 SwapPolyIndexes()	29
5.1.3.80 SwapSides()	29
5.1.3.81 TopX()	29
5.1.3.82 TranslatePath()	29
5.1.3.83 UpdateOutPtIdxs()	30
5.1.4 Variable Documentation	30
5.1.4.1 def_arc_tolerance	30
5.1.4.2 hiRange	30
5.1.4.3 loRange	30
5.1.4.4 pi	30
5.1.4.5 Skip	30
5.1.4.6 two_pi	30
5.1.4.7 Unassigned	31
5.2 DW Namespace Reference	31

5.2.1 Function Documentation	31
5.2.1.1 changeBuffer()	31
5.2.1.2 init()	31
5.2.2 Variable Documentation	31
5.2.2.1 map_buffer	31
5.2.2.2 window	32
5.3 timeutils Namespace Reference	32
5.3.1 Function Documentation	32
5.3.1.1 getTimeS()	32
5.3.1.2 timespecDiff()	32
6 Class Documentation	33
6.1 Angle Class Reference	33
6.1.1 Detailed Description	35
6.1.2 Member Enumeration Documentation	35
6.1.2.1 ANGLE_TYPE	35
6.1.3 Constructor & Destructor Documentation	35
6.1.3.1 Angle() [1/2]	35
6.1.3.2 Angle() [2/2]	35
6.1.4 Member Function Documentation	36
6.1.4.1 add()	36
6.1.4.2 checkValue()	36
6.1.4.3 copy()	36
6.1.4.4 cos()	37
6.1.4.5 degToRad()	37
6.1.4.6 div()	37
6.1.4.7 equal()	38
6.1.4.8 get()	38
6.1.4.9 getType()	38
6.1.4.10 getTypeName()	38
6.1.4.11 mul()	38
6.1.4.12 normalize()	39
6.1.4.13 operator *()	39
6.1.4.14 operator *=()	40
6.1.4.15 operator double()	40
6.1.4.16 operator float()	40
6.1.4.17 operator int()	40
6.1.4.18 operator long()	41
6.1.4.19 operator !=()	41
6.1.4.20 operator +()	41
6.1.4.21 operator +=()	42
6.1.4.22 operator -()	42

6.1.4.23 operator-=()	42
6.1.4.24 operator/()	43
6.1.4.25 operator/=()	43
6.1.4.26 operator=() [1/2]	43
6.1.4.27 operator=() [2/2]	44
6.1.4.28 operator==()	44
6.1.4.29 radToDeg()	44
6.1.4.30 set()	44
6.1.4.31 setType()	45
6.1.4.32 sin()	45
6.1.4.33 sub()	45
6.1.4.34 tan()	46
6.1.4.35 to_string()	46
6.1.4.36 toDeg()	46
6.1.4.37 toRad()	46
6.1.5 Friends And Related Function Documentation	46
6.1.5.1 operator<<	46
6.2 CalSettings Class Reference	47
6.2.1 Member Enumeration Documentation	49
6.2.1.1 InputType	49
6.2.1.2 Pattern	49
6.2.2 Constructor & Destructor Documentation	49
6.2.2.1 CalSettings()	49
6.2.3 Member Function Documentation	49
6.2.3.1 isListOfImages()	49
6.2.3.2 nextImage()	50
6.2.3.3 read()	50
6.2.3.4 readStringList()	50
6.2.3.5 validate()	51
6.2.3.6 write()	51
6.2.4 Member Data Documentation	52
6.2.4.1 aspectRatio	52
6.2.4.2 atImageList	52
6.2.4.3 boardSize	52
6.2.4.4 calibFixPrincipalPoint	52
6.2.4.5 calibrationPattern	52
6.2.4.6 calibZeroTangentDist	52
6.2.4.7 cameraID	53
6.2.4.8 delay	53
6.2.4.9 fixK1	53
6.2.4.10 fixK2	53
6.2.4.11 fixK3	53

6.2.4.12 fixK4	53
6.2.4.13 fixK5	54
6.2.4.14 flag	54
6.2.4.15 flipVertical	54
6.2.4.16 goodInput	54
6.2.4.17 imageList	54
6.2.4.18 input	54
6.2.4.19 inputCapture	54
6.2.4.20 inputType	55
6.2.4.21 nrFrames	55
6.2.4.22 outputFileName	55
6.2.4.23 showUndistorted	55
6.2.4.24 squareSize	55
6.2.4.25 useFisheye	55
6.2.4.26 writeExtrinsics	56
6.2.4.27 writePoints	56
6.3 CameraCapture Class Reference	56
6.3.1 Constructor & Destructor Documentation	56
6.3.1.1 CameraCapture()	56
6.3.1.2 ~CameraCapture()	57
6.3.2 Member Function Documentation	57
6.3.2.1 grab()	57
6.3.2.2 isAlive()	57
6.3.2.3 isOpened()	58
6.3.2.4 loadCoefficients()	58
6.3.2.5 startCamera()	58
6.4 ClipperLib::Clipper Class Reference	58
6.4.1 Constructor & Destructor Documentation	59
6.4.1.1 Clipper()	60
6.4.2 Member Function Documentation	60
6.4.2.1 Execute() [1/4]	60
6.4.2.2 Execute() [2/4]	60
6.4.2.3 Execute() [3/4]	60
6.4.2.4 Execute() [4/4]	60
6.4.2.5 ExecuteInternal()	61
6.4.2.6 ReverseSolution() [1/2]	61
6.4.2.7 ReverseSolution() [2/2]	61
6.4.2.8 StrictlySimple() [1/2]	61
6.4.2.9 StrictlySimple() [2/2]	61
6.5 ClipperLib::ClipperBase Class Reference	62
6.5.1 Member Typedef Documentation	63
6.5.1.1 MinimaList	63

6.5.1.2 ScanbeamList	64
6.5.2 Constructor & Destructor Documentation	64
6.5.2.1 ClipperBase()	64
6.5.2.2 ~ClipperBase()	64
6.5.3 Member Function Documentation	64
6.5.3.1 AddBoundsToLML()	64
6.5.3.2 AddPath()	64
6.5.3.3 AddPaths()	64
6.5.3.4 Clear()	65
6.5.3.5 CreateOutRec()	65
6.5.3.6 DeleteFromAEL()	65
6.5.3.7 DisposeAllOutRecs()	65
6.5.3.8 DisposeLocalMinimaList()	65
6.5.3.9 DisposeOutRec()	65
6.5.3.10 GetBounds()	65
6.5.3.11 InsertScanbeam()	66
6.5.3.12 LocalMinimaPending()	66
6.5.3.13 PopLocalMinima()	66
6.5.3.14 PopScanbeam()	66
6.5.3.15 PreserveCollinear() [1/2]	66
6.5.3.16 PreserveCollinear() [2/2]	66
6.5.3.17 ProcessBound()	66
6.5.3.18 Reset()	67
6.5.3.19 SwapPositionsInAEL()	67
6.5.3.20 UpdateEdgeIntoAEL()	67
6.5.4 Member Data Documentation	67
6.5.4.1 m_ActiveEdges	67
6.5.4.2 m_CurrentLM	67
6.5.4.3 m_edges	67
6.5.4.4 m_HasOpenPaths	67
6.5.4.5 m_MinimaList	68
6.5.4.6 m_PolyOuts	68
6.5.4.7 m_PreserveCollinear	68
6.5.4.8 m_Scanbeam	68
6.5.4.9 m_UseFullRange	68
6.6 ClipperLib::clipperException Class Reference	68
6.6.1 Constructor & Destructor Documentation	69
6.6.1.1 clipperException()	69
6.6.1.2 ~clipperException()	69
6.6.2 Member Function Documentation	69
6.6.2.1 what()	69
6.7 ClipperLib::ClipperOffset Class Reference	69

6.7.1 Constructor & Destructor Documentation	70
6.7.1.1 ClipperOffset()	70
6.7.1.2 ~ClipperOffset()	70
6.7.2 Member Function Documentation	70
6.7.2.1 AddPath()	70
6.7.2.2 AddPaths()	70
6.7.2.3 Clear()	70
6.7.2.4 Execute() [1/2]	71
6.7.2.5 Execute() [2/2]	71
6.7.3 Member Data Documentation	71
6.7.3.1 ArcTolerance	71
6.7.3.2 MiterLimit	71
6.8 Configuration2< T1 > Class Template Reference	71
6.8.1 Detailed Description	73
6.8.2 Constructor & Destructor Documentation	73
6.8.2.1 Configuration2() [1/3]	73
6.8.2.2 Configuration2() [2/3]	73
6.8.2.3 Configuration2() [3/3]	74
6.8.3 Member Function Documentation	74
6.8.3.1 angle() [1/2]	74
6.8.3.2 angle() [2/2]	74
6.8.3.3 copy()	75
6.8.3.4 distance()	75
6.8.3.5 equal()	76
6.8.3.6 EuDistance()	76
6.8.3.7 MaDistance()	76
6.8.3.8 offset() [1/3]	77
6.8.3.9 offset() [2/3]	77
6.8.3.10 offset() [3/3]	77
6.8.3.11 offset_angle()	78
6.8.3.12 offset_x()	78
6.8.3.13 offset_y()	79
6.8.3.14 operator Point2< T2 >()	79
6.8.3.15 operator=()	79
6.8.3.16 operator==()	80
6.8.3.17 point()	80
6.8.3.18 to_string()	80
6.8.3.19 x() [1/2]	81
6.8.3.20 x() [2/2]	81
6.8.3.21 y() [1/2]	81
6.8.3.22 y() [2/2]	81
6.8.4 Friends And Related Function Documentation	82

6.8.4.1 operator<<	82
6.9 Curve< T > Class Template Reference	82
6.9.1 Constructor & Destructor Documentation	84
6.9.1.1 Curve() [1/4]	84
6.9.1.2 Curve() [2/4]	84
6.9.1.3 Curve() [3/4]	84
6.9.1.4 Curve() [4/4]	84
6.9.2 Member Function Documentation	84
6.9.2.1 begin() [1/2]	85
6.9.2.2 begin() [2/2]	85
6.9.2.3 end() [1/2]	85
6.9.2.4 end() [2/2]	85
6.9.2.5 to_string()	85
6.9.3 Friends And Related Function Documentation	85
6.9.3.1 operator<<	85
6.9.4 Member Data Documentation	86
6.9.4.1 P0	86
6.9.4.2 P1	86
6.10 ClipperLib::DoublePoint Struct Reference	86
6.10.1 Constructor & Destructor Documentation	86
6.10.1.1 DoublePoint() [1/2]	86
6.10.1.2 DoublePoint() [2/2]	87
6.10.2 Member Data Documentation	87
6.10.2.1 X	87
6.10.2.2 Y	87
6.11 Dubins< T > Class Template Reference	87
6.11.1 Constructor & Destructor Documentation	89
6.11.1.1 Dubins() [1/4]	89
6.11.1.2 Dubins() [2/4]	89
6.11.1.3 Dubins() [3/4]	89
6.11.1.4 Dubins() [4/4]	89
6.11.2 Member Function Documentation	90
6.11.2.1 check()	90
6.11.2.2 getA1()	90
6.11.2.3 getA2()	90
6.11.2.4 getA3()	90
6.11.2.5 getId()	90
6.11.2.6 getKMax()	91
6.11.2.7 length()	91
6.11.2.8 LRL()	91
6.11.2.9 LSL()	91
6.11.2.10 LSR()	91

6.11.2.11 rangeSymm()	91
6.11.2.12 RLR()	92
6.11.2.13 RSL()	92
6.11.2.14 RSR()	92
6.11.2.15 scaleFromStandard()	92
6.11.2.16 scaleToStandard()	92
6.11.2.17 shortest_path()	93
6.11.2.18 splitIt()	93
6.11.2.19 to_string()	93
6.11.3 Friends And Related Function Documentation	93
6.11.3.1 operator<<	93
6.12 DubinsArc< T1, T2 > Class Template Reference	94
6.12.1 Constructor & Destructor Documentation	95
6.12.1.1 DubinsArc() [1/2]	95
6.12.1.2 DubinsArc() [2/2]	95
6.12.2 Member Function Documentation	95
6.12.2.1 getK()	95
6.12.2.2 length()	95
6.12.2.3 splitIt()	96
6.12.2.4 to_string()	96
6.12.3 Friends And Related Function Documentation	96
6.12.3.1 operator<<	96
6.13 Filter Class Reference	96
6.13.1 Detailed Description	97
6.13.2 Constructor & Destructor Documentation	97
6.13.2.1 Filter() [1/3]	97
6.13.2.2 Filter() [2/3]	97
6.13.2.3 Filter() [3/3]	98
6.13.3 Member Function Documentation	98
6.13.3.1 copy()	98
6.13.3.2 High()	99
6.13.3.3 Low()	99
6.13.3.4 operator vector< int >()	99
6.13.3.5 operator=()	99
6.13.3.6 to_string()	99
6.13.4 Friends And Related Function Documentation	100
6.13.4.1 operator<<	100
6.13.5 Member Data Documentation	100
6.13.5.1 high_h	100
6.13.5.2 high_s	100
6.13.5.3 high_v	101
6.13.5.4 low_h	101

6.13.5.5 low_s	101
6.13.5.6 low_v	101
6.14 CameraCapture::input_options_t Struct Reference	101
6.14.1 Detailed Description	102
6.14.2 Constructor & Destructor Documentation	102
6.14.2.1 input_options_t() [1/3]	102
6.14.2.2 input_options_t() [2/3]	102
6.14.2.3 input_options_t() [3/3]	102
6.14.3 Member Data Documentation	102
6.14.3.1 cameraFPS	103
6.14.3.2 frameHeight_px	103
6.14.3.3 frameWidth_px	103
6.14.3.4 nameCamera	103
6.15 ClipperLib::Int128 Class Reference	103
6.15.1 Constructor & Destructor Documentation	104
6.15.1.1 Int128() [1/3]	104
6.15.1.2 Int128() [2/3]	104
6.15.1.3 Int128() [3/3]	104
6.15.2 Member Function Documentation	104
6.15.2.1 operator "!="()	104
6.15.2.2 operator -()	104
6.15.2.3 operator -=()	105
6.15.2.4 operator >()	105
6.15.2.5 operator >=()	105
6.15.2.6 operator double()	105
6.15.2.7 operator+()	105
6.15.2.8 operator+=()	105
6.15.2.9 operator-()	105
6.15.2.10 operator<()	106
6.15.2.11 operator<=()	106
6.15.2.12 operator=()	106
6.15.2.13 operator==()	106
6.15.3 Member Data Documentation	106
6.15.3.1 hi	106
6.15.3.2 lo	106
6.16 ClipperLib::IntersectNode Struct Reference	107
6.16.1 Member Data Documentation	107
6.16.1.1 Edge1	107
6.16.1.2 Edge2	107
6.16.1.3 Pt	108
6.17 ClipperLib::IntPoint Struct Reference	108
6.17.1 Constructor & Destructor Documentation	108

6.17.1.1 IntPoint()	108
6.17.2 Friends And Related Function Documentation	108
6.17.2.1 operator"!="	109
6.17.2.2 operator=="	109
6.17.3 Member Data Documentation	109
6.17.3.1 X	109
6.17.3.2 Y	109
6.18 ClipperLib::IntRect Struct Reference	109
6.18.1 Member Data Documentation	110
6.18.1.1 bottom	110
6.18.1.2 left	110
6.18.1.3 right	110
6.18.1.4 top	110
6.19 ClipperLib::Join Struct Reference	110
6.19.1 Member Data Documentation	111
6.19.1.1 OffPt	111
6.19.1.2 OutPt1	111
6.19.1.3 OutPt2	111
6.20 ClipperLib::LocalMinimum Struct Reference	112
6.20.1 Member Data Documentation	112
6.20.1.1 LeftBound	112
6.20.1.2 RightBound	112
6.20.1.3 Y	113
6.21 ClipperLib::LocMinSorter Struct Reference	113
6.21.1 Member Function Documentation	113
6.21.1.1 operator>()	113
6.22 Mapp Class Reference	113
6.22.1 Constructor & Destructor Documentation	114
6.22.1.1 Mapp()	114
6.22.2 Member Function Documentation	114
6.22.2.1 addObject()	114
6.22.2.2 cellsFromSegment()	115
6.22.2.3 checkSegment()	115
6.22.2.4 checkSegmentCollisionWithType()	116
6.22.2.5 getPointType()	116
6.22.2.6 matrixToString()	116
6.22.2.7 printDimensions()	117
6.22.2.8 printMap()	117
6.22.3 Member Data Documentation	117
6.22.3.1 dimX	117
6.22.3.2 dimY	117
6.22.3.3 lengthX	117

6.22.3.4 lengthY	117
6.22.3.5 map	117
6.22.3.6 pixX	118
6.22.3.7 pixY	118
6.23 Object Class Reference	118
6.23.1 Member Function Documentation	119
6.23.1.1 computeCenter()	119
6.23.1.2 computeRadius()	119
6.23.1.3 insidePoly()	119
6.23.1.4 insidePolyApprox()	120
6.23.1.5 nPoints()	120
6.23.1.6 offsetting()	120
6.23.1.7 size()	121
6.23.1.8 toString()	121
6.23.2 Member Data Documentation	121
6.23.2.1 center	121
6.23.2.2 points	121
6.23.2.3 radius	122
6.24 Obstacle Class Reference	122
6.24.1 Constructor & Destructor Documentation	123
6.24.1.1 Obstacle()	123
6.24.2 Member Function Documentation	123
6.24.2.1 print()	123
6.24.2.2 toString()	124
6.25 ClipperLib::OutPt Struct Reference	124
6.25.1 Member Data Documentation	124
6.25.1.1 Idx	124
6.25.1.2 Next	125
6.25.1.3 Prev	125
6.25.1.4 Pt	125
6.26 ClipperLib::OutRec Struct Reference	125
6.26.1 Member Data Documentation	126
6.26.1.1 BottomPt	126
6.26.1.2 FirstLeft	126
6.26.1.3 Idx	126
6.26.1.4 IsHole	126
6.26.1.5 IsOpen	126
6.26.1.6 PolyNd	126
6.26.1.7 Pts	126
6.27 Point2< T > Class Template Reference	127
6.27.1 Detailed Description	128
6.27.2 Constructor & Destructor Documentation	128

6.27.2.1 Point2() [1/3]	128
6.27.2.2 Point2() [2/3]	128
6.27.2.3 Point2() [3/3]	129
6.27.3 Member Function Documentation	129
6.27.3.1 copy()	129
6.27.3.2 distance()	130
6.27.3.3 equal()	130
6.27.3.4 EuDistance()	130
6.27.3.5 MaDistance()	131
6.27.3.6 offset() [1/3]	131
6.27.3.7 offset() [2/3]	132
6.27.3.8 offset() [3/3]	132
6.27.3.9 offset_x()	132
6.27.3.10 offset_y()	133
6.27.3.11 operator cv::Point()	133
6.27.3.12 operator"!=()	133
6.27.3.13 operator<()	134
6.27.3.14 operator=()	134
6.27.3.15 operator==(())	134
6.27.3.16 to_string()	135
6.27.3.17 x() [1/2]	135
6.27.3.18 x() [2/2]	135
6.27.3.19 y() [1/2]	135
6.27.3.20 y() [2/2]	136
6.27.4 Friends And Related Function Documentation	136
6.27.4.1 operator<<	136
6.28 ClipperLib::PolyNode Class Reference	136
6.28.1 Constructor & Destructor Documentation	137
6.28.1.1 PolyNode()	138
6.28.1.2 ~PolyNode()	138
6.28.2 Member Function Documentation	138
6.28.2.1 ChildCount()	138
6.28.2.2 GetNext()	138
6.28.2.3 IsHole()	138
6.28.2.4 IsOpen()	138
6.28.3 Friends And Related Function Documentation	138
6.28.3.1 Clipper	138
6.28.3.2 ClipperOffset	139
6.28.4 Member Data Documentation	139
6.28.4.1 Childs	139
6.28.4.2 Contour	139
6.28.4.3 Parent	139

6.29 ClipperLib::PolyTree Class Reference	139
6.29.1 Constructor & Destructor Documentation	140
6.29.1.1 ~PolyTree()	140
6.29.2 Member Function Documentation	140
6.29.2.1 Clear()	140
6.29.2.2 GetFirst()	141
6.29.2.3 Total()	141
6.29.3 Friends And Related Function Documentation	141
6.29.3.1 Clipper	141
6.30 Settings Class Reference	141
6.30.1 Detailed Description	144
6.30.2 Member Enumeration Documentation	144
6.30.2.1 COLOR	144
6.30.3 Constructor & Destructor Documentation	144
6.30.3.1 Settings()	144
6.30.3.2 ~Settings()	145
6.30.4 Member Function Documentation	145
6.30.4.1 addUnMap()	145
6.30.4.2 changeMask() [1/2]	145
6.30.4.3 changeMask() [2/2]	146
6.30.4.4 clean()	146
6.30.4.5 cleanAndRead()	146
6.30.4.6 getTemplates() [1/3]	146
6.30.4.7 getTemplates() [2/3]	147
6.30.4.8 getTemplates() [3/3]	147
6.30.4.9 maps() [1/4]	147
6.30.4.10 maps() [2/4]	149
6.30.4.11 maps() [3/4]	149
6.30.4.12 maps() [4/4]	150
6.30.4.13 readFromFile()	150
6.30.4.14 save()	150
6.30.4.15 to_string()	151
6.30.4.16 unMaps() [1/4]	151
6.30.4.17 unMaps() [2/4]	152
6.30.4.18 unMaps() [3/4]	152
6.30.4.19 unMaps() [4/4]	153
6.30.4.20 writeToFile()	153
6.30.5 Friends And Related Function Documentation	153
6.30.5.1 operator<<	153
6.30.6 Member Data Documentation	154
6.30.6.1 blackMask	154
6.30.6.2 blueMask	154

6.30.6.3 calibrationFile	154
6.30.6.4 convexHullFile	154
6.30.6.5 greenMask	155
6.30.6.6 intrinsicCalibrationFile	155
6.30.6.7 kernelSide	155
6.30.6.8 mapsFolder	155
6.30.6.9 mapsNames	155
6.30.6.10 mapsUnNames	155
6.30.6.11 redMask	156
6.30.6.12 robotMask	156
6.30.6.13 templates	156
6.30.6.14 templatesFolder	156
6.30.6.15 victimMask	156
6.30.6.16 whiteMask	156
6.31 ClipperLib::TEdge Struct Reference	157
6.31.1 Member Data Documentation	157
6.31.1.1 Bot	158
6.31.1.2 Curr	158
6.31.1.3 Dx	158
6.31.1.4 Next	158
6.31.1.5 NextInAEL	158
6.31.1.6 NextInLML	158
6.31.1.7 NextInSEL	158
6.31.1.8 OutIdx	158
6.31.1.9 PolyTyp	159
6.31.1.10 Prev	159
6.31.1.11 PrevInAEL	159
6.31.1.12 PrevInSEL	159
6.31.1.13 Side	159
6.31.1.14 Top	159
6.31.1.15 WindCnt	159
6.31.1.16 WindCnt2	159
6.31.1.17 WindDelta	160
6.32 Tuple< T > Class Template Reference	160
6.32.1 Detailed Description	161
6.32.2 Constructor & Destructor Documentation	161
6.32.2.1 Tuple() [1/2]	161
6.32.2.2 Tuple() [2/2]	161
6.32.3 Member Function Documentation	162
6.32.3.1 add()	162
6.32.3.2 begin() [1/2]	162
6.32.3.3 begin() [2/2]	162

6.32.3.4 distance()	162
6.32.3.5 end() [1/2]	163
6.32.3.6 end() [2/2]	163
6.32.3.7 EuDistance()	163
6.32.3.8 get()	164
6.32.3.9 MaDistance()	164
6.32.3.10 operator vector< T1 >()	165
6.32.3.11 remove()	165
6.32.3.12 set()	165
6.32.3.13 size()	166
6.32.3.14 to_string()	166
6.32.4 Friends And Related Function Documentation	166
6.32.4.1 operator<<	166
6.33 Victim Class Reference	167
6.33.1 Constructor & Destructor Documentation	168
6.33.1.1 Victim()	168
6.33.2 Member Function Documentation	168
6.33.2.1 getValue()	168
6.33.2.2 print()	168
6.33.2.3 setValue()	168
6.33.2.4 toString()	169
6.33.3 Member Data Documentation	169
6.33.3.1 value	169
7 File Documentation	171
7.1 src/calibration.cc File Reference	171
7.1.1 Function Documentation	172
7.1.1.1 calcBoardCornerPositions()	172
7.1.1.2 calibration()	172
7.1.1.3 computeReprojectionErrors()	172
7.1.1.4 read()	173
7.1.1.5 runCalibration()	173
7.1.1.6 runCalibrationAndSave()	174
7.1.1.7 saveCameraParams()	175
7.2 src/camera_capture.cc File Reference	175
7.2.1 Macro Definition Documentation	176
7.2.1.1 DEBUG	176
7.2.1.2 SDEBUG	176
7.3 src/clipper.cc File Reference	176
7.3.1 Macro Definition Documentation	179
7.3.1.1 HORIZONTAL	179
7.3.1.2 NEAR_ZERO	179

7.3.1.3 TOLERANCE	179
7.4 src/configure.cc File Reference	180
7.4.1 Function Documentation	180
7.4.1.1 configure()	180
7.4.1.2 on_high_h_thresh_trackbar()	181
7.4.1.3 on_high_s_thresh_trackbar()	181
7.4.1.4 on_high_v_thresh_trackbar()	181
7.4.1.5 on_low_h_thresh_trackbar()	181
7.4.1.6 on_low_s_thresh_trackbar()	181
7.4.1.7 on_low_v_thresh_trackbar()	181
7.4.1.8 show_all_conditions()	181
7.4.1.9 update_trackers()	182
7.4.2 Variable Documentation	182
7.4.2.1 filter	182
7.5 src/detection.cc File Reference	182
7.5.1 Macro Definition Documentation	183
7.5.1.1 EPS_CURVE	183
7.5.2 Function Documentation	184
7.5.2.1 crop_number_section()	184
7.5.2.2 detection()	184
7.5.2.3 erode_dilation()	184
7.5.2.4 find_contours()	185
7.5.2.5 load_number_template()	185
7.5.2.6 number_recognition()	185
7.5.2.7 save_convex_hull()	186
7.5.2.8 shape_detection()	186
7.5.3 Variable Documentation	186
7.5.3.1 s	186
7.5.3.2 templates	186
7.6 src/dubins.cc File Reference	187
7.7 src/include/calibration.hh File Reference	187
7.7.1 Detailed Description	188
7.7.2 Enumeration Type Documentation	188
7.7.2.1 anonymous enum	188
7.7.3 Function Documentation	189
7.7.3.1 calibration()	189
7.7.3.2 runCalibrationAndSave()	189
7.8 src/include/camera_capture.hh File Reference	190
7.9 src/include/clipper.hh File Reference	191
7.9.1 Macro Definition Documentation	193
7.9.1.1 CLIPPER_VERSION	193
7.9.1.2 use_lines	193

7.10 src/include/configure.hh File Reference	193
7.10.1 Function Documentation	194
7.10.1.1 configure()	194
7.10.1.2 show_all_conditions()	195
7.11 src/include/detection.hh File Reference	195
7.11.1 Function Documentation	196
7.11.1.1 crop_number_section()	196
7.11.1.2 detection()	196
7.11.1.3 erode_dilation()	197
7.11.1.4 find_contours()	197
7.11.1.5 load_number_template()	197
7.11.1.6 number_recognition()	198
7.11.1.7 save_convex_hull()	198
7.11.1.8 shape_detection()	198
7.12 src/include/draw.hh File Reference	199
7.12.1 Typedef Documentation	199
7.12.1.1 int	200
7.13 src/include/dubins.hh File Reference	200
7.13.1 Macro Definition Documentation	201
7.13.1.1 KMAX	201
7.13.1.2 MORE_FUNCTIONS	201
7.13.1.3 PIECE_LENGTH	201
7.13.2 Function Documentation	201
7.13.2.1 circline()	201
7.13.2.2 sinc()	201
7.14 src/include/filter.hh File Reference	202
7.15 src/include/map.hh File Reference	202
7.15.1 Enumeration Type Documentation	203
7.15.1.1 OBJ_TYPE	203
7.16 src/include/maths.hh File Reference	203
7.16.1 Macro Definition Documentation	205
7.16.1.1 DEGTORAD	205
7.16.1.2 DInf	205
7.16.1.3 Epsi	205
7.16.1.4 RADTODEG	205
7.16.1.5 tupleConstIter	206
7.16.1.6 tupleIter	206
7.16.2 Enumeration Type Documentation	206
7.16.2.1 DISTANCE_TYPE	206
7.16.3 Function Documentation	206
7.16.3.1 equal()	206
7.16.3.2 pow2()	207

7.16.4 Variable Documentation	207
7.16.4.1 A_180	207
7.16.4.2 A_2PI	207
7.16.4.3 A_360	207
7.16.4.4 A_PI	207
7.17 src/include/objects.hh File Reference	208
7.18 src/include/settings.hh File Reference	208
7.19 src/include/unwrapping.hh File Reference	209
7.19.1 Function Documentation	210
7.19.1.1 find_rect()	210
7.19.1.2 loadCoefficients()	211
7.19.1.3 unwrapping()	211
7.20 src/include/utils.hh File Reference	211
7.20.1 Macro Definition Documentation	213
7.20.1.1 COUT	213
7.20.1.2 INFO	213
7.20.1.3 NAME	213
7.20.2 Function Documentation	213
7.20.2.1 my_imshow()	213
7.20.2.2 mywaitkey() [1/3]	214
7.20.2.3 mywaitkey() [2/3]	214
7.20.2.4 mywaitkey() [3/3]	214
7.21 src/map.cc File Reference	214
7.22 src/math.cc File Reference	215
7.23 src/objects.cc File Reference	215
7.24 src/run/calibration_run.cc File Reference	216
7.24.1 Function Documentation	216
7.24.1.1 main()	216
7.25 src/run/detection_run.cc File Reference	216
7.25.1 Function Documentation	217
7.25.1.1 main()	217
7.26 src/run/main.cc File Reference	217
7.26.1 Function Documentation	217
7.26.1.1 main()	217
7.27 src/run/unwrapping_run.cc File Reference	217
7.27.1 Function Documentation	218
7.27.1.1 main()	218
7.28 src/settings.cc File Reference	218
7.28.1 Macro Definition Documentation	218
7.28.1.1 NPOS	218
7.28.2 Function Documentation	219
7.28.2.1 getFiles()	219

7.28.2.2 vecToFile()	219
7.29 src/unwrapping.cc File Reference	219
7.29.1 Macro Definition Documentation	220
7.29.1.1 AREA_MIN	220
7.29.1.2 AREA_RATIO	220
7.29.2 Function Documentation	220
7.29.2.1 distance()	220
7.29.2.2 find_rect()	221
7.29.2.3 loadCoefficients()	221
7.29.2.4 unwrapping()	222
7.30 src/utils.cc File Reference	222
7.30.1 Function Documentation	223
7.30.1.1 my_imshow()	223
7.30.1.2 mywaitkey() [1/3]	223
7.30.1.3 mywaitkey() [2/3]	223
7.30.1.4 mywaitkey() [3/3]	223
Index	225

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

ClipperLib	9
DW	31
timeutils	32

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Angle	33
CalSettings	47
ClipperLib::ClipperBase	62
ClipperLib::Clipper	58
ClipperLib::ClipperOffset	69
Configuration2< T1 >	71
Configuration2< T >	71
Configuration2< T2 >	71
Curve< T >	82
Dubins< T >	87
Curve< T2 >	82
DubinsArc< T1, T2 >	94
ClipperLib::DoublePoint	86
exception	
ClipperLib::clipperException	68
Filter	96
CameraCapture::input_options_t	101
ClipperLib::Int128	103
ClipperLib::IntersectNode	107
ClipperLib::IntPoint	108
ClipperLib::IntRect	109
ClipperLib::Join	110
ClipperLib::LocalMinimum	112
ClipperLib::LocMinSorter	113
Mapp	113
Object	118
Obstacle	122
Victim	167
ClipperLib::OutPt	124
ClipperLib::OutRec	125
Point2< T >	127
Point2< int >	127
Point2< T1 >	127
Point2< T2 >	127

ClipperLib::PolyNode	136
ClipperLib::PolyTree	139
Settings	141
ClipperLib::TEdge	157
Tuple< T >	160
Tuple< int >	160
Tuple< string >	160
Tuple< T1 >	160
Tuple< T2 >	160
VideoCapture	
CameraCapture	56

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

Angle	33
CalSettings	47
CameraCapture	56
ClipperLib::Clipper	58
ClipperLib::ClipperBase	62
ClipperLib::clipperException	68
ClipperLib::ClipperOffset	69
Configuration2< T1 >	
This class stores a configuration, that is a point and an angle	71
Curve< T >	82
ClipperLib::DoublePoint	86
Dubins< T >	87
DubinsArc< T1, T2 >	94
Filter	96
CameraCapture::input_options_t	
Structure for store the input option for the class CameraCapture	101
ClipperLib::Int128	103
ClipperLib::IntersectNode	107
ClipperLib::IntPoint	108
ClipperLib::IntRect	109
ClipperLib::Join	110
ClipperLib::LocalMinimum	112
ClipperLib::LocMinSorter	113
Mapp	113
Object	118
Obstacle	122
ClipperLib::OutPt	124
ClipperLib::OutRec	125
Point2< T >	
Class that stores two value to construct a point in 2D. The value is saved in a Tuple	127
ClipperLib::PolyNode	136
ClipperLib::PolyTree	139

Settings	141
ClipperLib::TEdge	157
Tuple< T >	160
Victim	167

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

src/calibration.cc	171
src/camera_capture.cc	175
src/clipper.cc	176
src/configure.cc	180
src/detection.cc	182
src/dubins.cc	187
src/map.cc	214
src/math.cc	215
src/objects.cc	215
src/settings.cc	218
src/unwrapping.cc	219
src/utls.cc	222
src/include/calibration.hh	
Library for calibration	187
src/include/camera_capture.hh	190
src/include/clipper.hh	191
src/include/configure.hh	193
src/include/detection.hh	195
src/include/draw.hh	199
src/include/dubins.hh	200
src/include/filter.hh	202
src/include/map.hh	202
src/include/math.cc	203
src/include/objects.hh	208
src/include/settings.hh	208
src/include/unwrapping.hh	209
src/include/utls.hh	211
src/run/calibration_run.cc	216
src/run/detection_run.cc	216
src/run/main.cc	217
src/run/unwrapping_run.cc	217

Chapter 5

Namespace Documentation

5.1 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 [cInt](#)
- 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](#) { [ctlIntersection](#), [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 [DistanceSqrD](#) (const [IntPoint](#) &pt1, const [IntPoint](#) &pt2)
- double [DistanceFromLineSqrD](#) (const [IntPoint](#) &pt, const [IntPoint](#) &ln1, const [IntPoint](#) &ln2)
- bool [SlopesNearCollinear](#) (const [IntPoint](#) &pt1, const [IntPoint](#) &pt2, const [IntPoint](#) &pt3, double distSqrD)
- bool [PointsAreClose](#) ([IntPoint](#) pt1, [IntPoint](#) pt2, double distSqrD)
- [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 [clnt](#) const [loRange](#) = 0x3FFFFFFF
- static [clnt](#) const [hiRange](#) = 0x3FFFFFFFFFFFFFFFLL

5.1.1 Typedef Documentation

5.1.1.1 cInt

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

5.1.1.2 EdgeList

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

5.1.1.3 IntersectList

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

5.1.1.4 JoinList

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

5.1.1.5 long64

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

5.1.1.6 Path

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

5.1.1.7 Paths

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

5.1.1.8 PolyNodes

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

5.1.1.9 PolyOutList

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

5.1.1.10 ulong64

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

5.1.2 Enumeration Type Documentation

5.1.2.1 ClipType

```
enum ClipperLib::ClipType
```

Enumerator

ctIntersection	
ctUnion	
ctDifference	
ctXor	

5.1.2.2 Direction

```
enum ClipperLib::Direction
```

Enumerator

dRightToLeft	
dLeftToRight	

5.1.2.3 EdgeSide

enum `ClipperLib::EdgeSide`

Enumerator

esLeft	
esRight	

5.1.2.4 EndType

enum `ClipperLib::EndType`

Enumerator

etClosedPolygon	
etClosedLine	
etOpenButt	
etOpenSquare	
etOpenRound	

5.1.2.5 InitOptions

enum `ClipperLib::InitOptions`

Enumerator

ioReverseSolution	
ioStrictlySimple	
ioPreserveCollinear	

5.1.2.6 JoinType

enum `ClipperLib::JoinType`

Enumerator

jtSquare	
jtRound	
jtMiter	

5.1.2.7 NodeType

```
enum ClipperLib::NodeType
```

Enumerator

ntAny	
ntOpen	
ntClosed	

5.1.2.8 PolyFillType

```
enum ClipperLib::PolyFillType
```

Enumerator

pftEvenOdd	
pftNonZero	
pftPositive	
pftNegative	

5.1.2.9 PolyType

```
enum ClipperLib::PolyType
```

Enumerator

ptSubject	
ptClip	

5.1.3 Function Documentation

5.1.3.1 Abs()

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

5.1.3.2 AddPolyNodeToPaths()

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

5.1.3.3 Area() [1/3]

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

5.1.3.4 Area() [2/3]

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

5.1.3.5 Area() [3/3]

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

5.1.3.6 CleanPolygon() [1/2]

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

5.1.3.7 CleanPolygon() [2/2]

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


5.1.3.8 CleanPolygons() [1/2]

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

5.1.3.9 CleanPolygons() [2/2]

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

5.1.3.10 ClosedPathsFromPolyTree()

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

5.1.3.11 DisposeOutPts()

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

5.1.3.12 DistanceFromLineSqrd()

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

5.1.3.13 DistanceSqrd()

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

5.1.3.14 DupOutPt()

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

5.1.3.15 E2InsertsBeforeE1()

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

5.1.3.16 EdgesAdjacent()

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

5.1.3.17 ExcludeOp()

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

5.1.3.18 FindNextLocMin()

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

5.1.3.19 FirstIsBottomPt()

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

5.1.3.20 GetBottomPt()

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

5.1.3.21 GetDx()

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

5.1.3.22 GetHorzDirection()

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

5.1.3.23 GetLowermostRec()

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

5.1.3.24 GetMaximaPair()

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

5.1.3.25 GetMaximaPairEx()

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

5.1.3.26 GetNextInAEL()

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

5.1.3.27 GetOverlap()

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

5.1.3.28 GetOverlapSegment()

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

5.1.3.29 GetUnitNormal()

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

5.1.3.30 HorzSegmentsOverlap()

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

5.1.3.31 InitEdge()

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

5.1.3.32 InitEdge2()

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

5.1.3.33 Int128Mul()

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

5.1.3.34 IntersectListSort()

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

5.1.3.35 IntersectPoint()

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

5.1.3.36 IsHorizontal()

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

5.1.3.37 IsIntermediate()

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

5.1.3.38 IsMaxima()

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

5.1.3.39 IsMinima()

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

5.1.3.40 JoinHorz()

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

5.1.3.41 Minkowski()

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

5.1.3.42 MinkowskiDiff()

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

5.1.3.43 MinkowskiSum() [1/2]

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

5.1.3.44 MinkowskiSum() [2/2]

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

5.1.3.45 OpenPathsFromPolyTree()

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

5.1.3.46 operator<<() [1/5]

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

5.1.3.47 operator<<() [2/5]

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

5.1.3.48 operator<<() [3/5]

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

5.1.3.49 operator<<() [4/5]

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

5.1.3.50 operator<<() [5/5]

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

5.1.3.51 Orientation()

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

5.1.3.52 OutRec1RightOfOutRec2()

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

5.1.3.53 ParseFirstLeft()

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


5.1.3.54 PointCount()

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

5.1.3.55 PointInPolygon() [1/2]

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

5.1.3.56 PointInPolygon() [2/2]

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

5.1.3.57 PointIsVertex()

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

5.1.3.58 PointsAreClose()

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

5.1.3.59 Poly2ContainsPoly1()

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

5.1.3.60 PolyTreeToPaths()

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

5.1.3.61 Pt2IsBetweenPt1AndPt3()

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

5.1.3.62 RangeTest()

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

5.1.3.63 RemoveEdge()

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

5.1.3.64 ReverseHorizontal()

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

5.1.3.65 ReversePath()

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

5.1.3.66 ReversePaths()

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

5.1.3.67 ReversePolyPtLinks()

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

5.1.3.68 Round()

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

5.1.3.69 SetDx()

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

5.1.3.70 SimplifyPolygon()

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

5.1.3.71 SimplifyPolygons() [1/2]

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

5.1.3.72 SimplifyPolygons() [2/2]

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

5.1.3.73 SlopesEqual() [1/3]

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

5.1.3.74 SlopesEqual() [2/3]

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

5.1.3.75 SlopesEqual() [3/3]

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

5.1.3.76 SlopesNearCollinear()

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

5.1.3.77 SwapIntersectNodes()

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

5.1.3.78 SwapPoints()

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

5.1.3.79 SwapPolyIndexes()

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

5.1.3.80 SwapSides()

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

5.1.3.81 TopX()

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

5.1.3.82 TranslatePath()

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

5.1.3.83 UpdateOutPtIdxs()

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

5.1.4 Variable Documentation

5.1.4.1 def_arc_tolerance

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

5.1.4.2 hiRange

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

5.1.4.3 loRange

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

5.1.4.4 pi

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

5.1.4.5 Skip

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

5.1.4.6 two_pi

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

5.1.4.7 Unassigned

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

5.2 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](#)

5.2.1 Function Documentation

5.2.1.1 [changeBuffer\(\)](#)

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

5.2.1.2 [init\(\)](#)

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

5.2.2 Variable Documentation

5.2.2.1 [map_buffer](#)

```
GLuint DW::map_buffer
```

5.2.2.2 window

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

5.3 timeutils Namespace Reference

Functions

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

5.3.1 Function Documentation

5.3.1.1 getTimeS()

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

5.3.1.2 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.
- bool `equal (const Angle &th0, const Angle &th1)`
- `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 `operator== (const Angle &phi)`
- bool `operator!= (const Angle &phi)`
- double `cos ()` const
Compute the cosine of the angle. \returns A double that is the cosine of the angle.
- double `sin ()` const
Compute the sine of the angle. \returns A double that is the sine of the angle.
- double `tan ()` const
Compute the tangent of the angle. \returns 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 ()` 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. \returnns 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

in	A	The costant to use to divide.
----	---	-------------------------------

Returns

The angle divided.

6.1.4.7 equal()

```
bool Angle::equal (
    const Angle & th0,
    const Angle & th1 ) [inline]
```

This function takes the value in radians of two angles, and using the equal function for double calculate if they are equal or not.

Parameters

in	<i>th0</i>	The first angle.
in	<i>th1</i>	The second angle.

Returns

true if the two angle 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 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 costant to use to multiply.
----	------------	---------------------------------

Returns

The angle multiplied.

6.1.4.12 `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`.

6.1.4.13 `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.

6.1.4.14 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	A	The coefficient.
----	---	------------------

Returns

this.

6.1.4.15 operator double()

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

Cast to double.

Returns

The value in RAD of the angle casted to double

6.1.4.16 operator float()

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

Cast to float.

Returns

The value in RAD of the angle casted to float

6.1.4.17 operator int()

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

Cast to int.

Returns

The value in RAD of the angle casted to int

6.1.4.18 operator long()

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

Cast to long.

Returns

The value in RAD of the angle casted to long

6.1.4.19 operator!=(())

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

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.

6.1.4.20 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.

6.1.4.21 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.`

6.1.4.22 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.

6.1.4.23 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.`

6.1.4.24 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.

6.1.4.25 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`.

6.1.4.26 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.

6.1.4.27 operator=() [2/2]

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

6.1.4.28 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.

6.1.4.29 radToDeg()

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

Converts and stores the angle from RAD to DEG.

Returns

The value of the angle.

6.1.4.30 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>	

6.1.4.31 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>	

6.1.4.32 sin()

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

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

6.1.4.33 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.

6.1.4.34 tan()

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

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

6.1.4.35 to_string()

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

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

Returns

A string stream.

6.1.4.36 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

6.1.4.37 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

<code>in</code>	<code>out</code>	The out stream.
<code>in</code>	<code>data</code>	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 [isOpened](#) ()
- 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

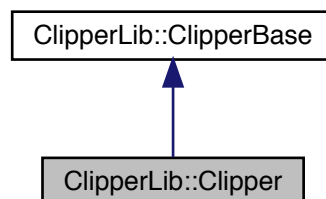
The documentation for this class was generated from the following files:

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

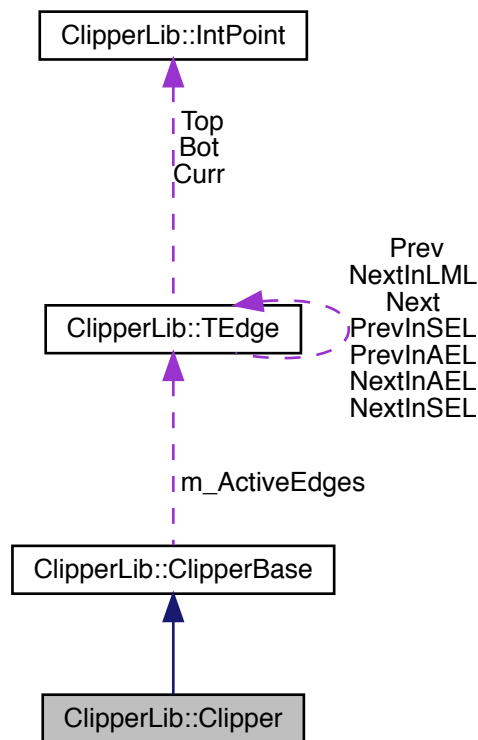
6.4 ClipperLib::Clipper Class Reference

```
#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]
```

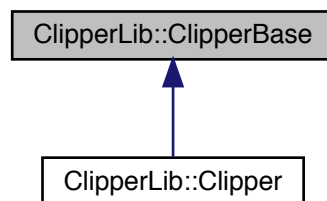
The documentation for this class was generated from the following files:

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

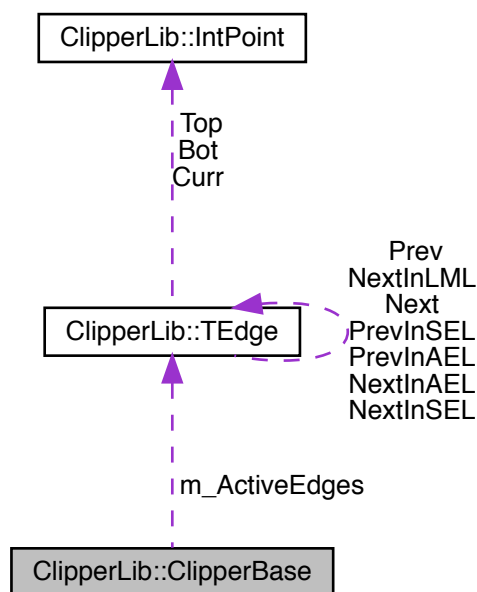
6.5 ClipperLib::ClipperBase Class Reference

```
#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](#) > [MinimalList](#)
- typedef std::priority_queue< [cInt](#) > [ScanbeamList](#)

Protected Member Functions

- void [DisposeLocalMinimalList](#) ()
- [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

- [MinimalList::iterator](#) [m_CurrentLM](#)
- [MinimalList](#) [m_MinimalList](#)
- 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 MinimalList

```
typedef std::vector<LocalMinimum> ClipperLib::ClipperBase::MinimalList [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 = operator. 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`.

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	\leftarrow \overleftarrow{x}	The abscissa coordinate.
in	\leftarrow \overleftarrow{y}	The ordinate coordinate.
in	\leftarrow \overleftarrow{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	\leftarrow \overleftarrow{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	<code>_offset</code>	The offset.
----	----------------------	-------------

Returns

1 if everything went fine, 0 otherwise.

6.8.3.14 operator Point2< T2 >()

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

Cast of Configuration to [Point2](#).

Template Parameters

<code>T2</code>	Type of Point2 to be casted to.
-----------------	---

Returns

A [Point2](#) of type T2.

6.8.3.15 operator=()

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

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

Parameters

in	A	Configuration to be coppied.
----	---	------------------------------

Returns

this.

6.8.3.16 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.

6.8.3.17 point()

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

Returns

A [Point2](#) variable containing the coordinates.

6.8.3.18 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.

6.8.3.19 `x()` [1/2]

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

Returns

The abscissa coordinate.

6.8.3.20 `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.

6.8.3.21 `y()` [1/2]

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

Returns

The ordinate coordinate.

6.8.3.22 `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	\leftrightarrow	The value to be stored.
	\overleftarrow{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.

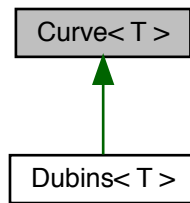
The documentation for this class was generated from the following file:

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

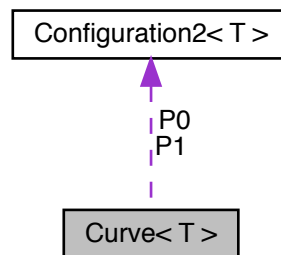
6.9 Curve< T > Class Template Reference

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

Inheritance diagram for Curve< T >:



Collaboration diagram for Curve< T >:



Public Member Functions

- [Curve](#) ()
- [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
- [Configuration2](#)< T > [end](#) () const
- void [begin](#) ([Configuration2](#)< T > _P0)
- void [end](#) ([Configuration2](#)< T > _P1)
- stringstream [to_string](#) () const

Protected Attributes

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

Friends

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

6.9.1 Constructor & Destructor Documentation

6.9.1.1 [Curve\(\)](#) [1/4]

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

6.9.1.2 [Curve\(\)](#) [2/4]

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

6.9.1.3 [Curve\(\)](#) [3/4]

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

6.9.1.4 [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]
```

6.9.2 Member Function Documentation

6.9.2.1 begin() [1/2]

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

6.9.2.2 begin() [2/2]

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

6.9.2.3 end() [1/2]

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

6.9.2.4 end() [2/2]

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

6.9.2.5 to_string()

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

6.9.3 Friends And Related Function Documentation

6.9.3.1 operator<<

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

6.9.4 Member Data Documentation

6.9.4.1 P0

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

6.9.4.2 P1

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

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
```

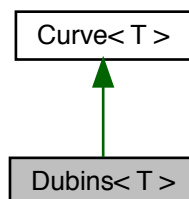
The documentation for this struct was generated from the following file:

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

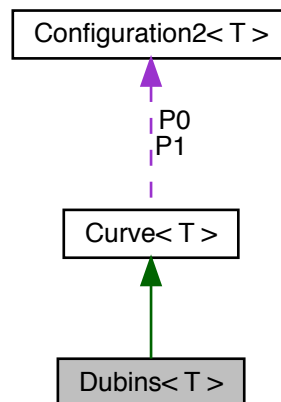
6.11 Dubins< T > Class Template Reference

```
#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`
- `double length () const`
- `double getId ()`
- `DubinsArc getA1 () const`
- `DubinsArc getA2 () const`
- `DubinsArc getA3 () const`
- `Tuple< double > LSL (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > RSR (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > LSR (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > RSL (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > RLR (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > LRL (Angle th0, Angle th1, double _kmax)`
- `Tuple< double > scaleToStandard ()`
- `Tuple< double > scaleFromStandard (double lambda, double sc_s1, double sc_s2, double sc_s3)`
- `int shortest_path ()`
- `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`

Static Public Member Functions

- `static double rangeSymm (double ang)`

Friends

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

Additional Inherited Members

6.11.1 Constructor & Destructor Documentation

6.11.1.1 Dubins() [1/4]

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

6.11.1.2 Dubins() [2/4]

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

6.11.1.3 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]
```

6.11.1.4 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]
```

6.11.2 Member Function Documentation

6.11.2.1 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]
```

6.11.2.2 getA1()

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

6.11.2.3 getA2()

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

6.11.2.4 getA3()

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

6.11.2.5 getId()

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

6.11.2.6 getKMax()

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

6.11.2.7 length()

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

6.11.2.8 LRL()

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

6.11.2.9 LSL()

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

6.11.2.10 LSR()

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

6.11.2.11 rangeSymm()

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

6.11.2.12 RLR()

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

6.11.2.13 RSL()

```
template<class T >
Tuple<double> Dubins< T >::RSL (
    Angle th0,
    Angle th1,
    double _kmax ) [inline]
```

6.11.2.14 RSR()

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

6.11.2.15 scaleFromStandard()

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

6.11.2.16 scaleToStandard()

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

6.11.2.17 shortest_path()

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

6.11.2.18 splittt()

```
template<class T >
Tuple<Tuple<Point2<double> > > > Dubins< T >::splittt (
    int _arch = 0,
    double _L = PIECE_LENGTH ) [inline]
```

6.11.2.19 to_string()

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

6.11.3 Friends And Related Function Documentation

6.11.3.1 operator<<

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

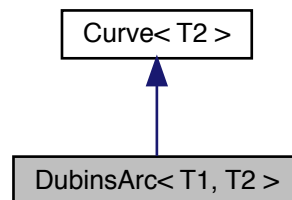
The documentation for this class was generated from the following file:

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

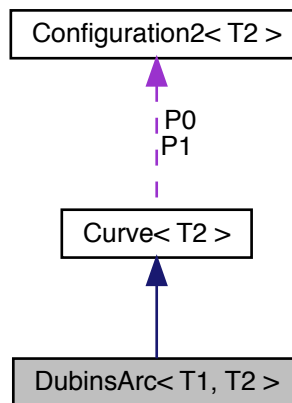
6.12 DubinsArc< T1, T2 > Class Template Reference

```
#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
- `T1 length` () const
- `Tuple< Point2< T2 > > splitIt` (double `_L=PIECE_LENGTH`)
- `stringstream to_string` () const

Friends

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

Additional Inherited Members

6.12.1 Constructor & Destructor Documentation

6.12.1.1 DubinsArc() [1/2]

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

6.12.1.2 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]
```

6.12.2 Member Function Documentation

6.12.2.1 getK()

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

6.12.2.2 length()

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

6.12.2.3 splittt()

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

6.12.2.4 to_string()

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

6.12.3 Friends And Related Function Documentation

6.12.3.1 operator<<

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

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

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

6.13 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)`

6.13.1 Detailed Description

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

6.13.2 Constructor & Destructor Documentation

6.13.2.1 Filter() [1/3]

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

Default constructor: it set all values to 0.

6.13.2.2 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

6.13.2.3 Filter() [3/3]

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

Constructor from a vector.

Parameters

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

6.13.3 Member Function Documentation**6.13.3.1 copy()**

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

A function to copy a filter to this.

Parameters

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

Returns

this filter with the new values copied.

6.13.3.2 High()

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

Returns a Scalar containing the lower boudary.

6.13.3.3 Low()

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

Returns a Scalar containing the lower boudary.

6.13.3.4 operator vector< int >()

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

Overload of operator cast to vector<int>.

Returns

A vector containing the 6 values.

6.13.3.5 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.

6.13.3.6 to_string()

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

Save value in a stringstream.

Returns

A stringstream containing the values of both boundaries.

6.13.4 Friends And Related Function Documentation**6.13.4.1 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.

6.13.5 Member Data Documentation**6.13.5.1 high_h**

```
int Filter::high_h
```

Higher value for hue.

6.13.5.2 high_s

```
int Filter::high_s
```

Higher value for saturation.

6.13.5.3 high_v

`int` Filter::high_v

Higher value for value.

6.13.5.4 low_h

`int` Filter::low_h

Lower value for hue.

6.13.5.5 low_s

`int` Filter::low_s

Lower value for saturation.

6.13.5.6 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](#)

6.14 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]

6.14.1 Detailed Description

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

[frameHeight_px](#) desired height of the camera

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

[cameraFPS](#) desired FPS of the camera

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

6.14.2 Constructor & Destructor Documentation

6.14.2.1 [input_options_t\(\)](#) [1/3]

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

6.14.2.2 [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_ )
```

6.14.2.3 [input_options_t\(\)](#) [3/3]

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

6.14.3 Member Data Documentation

6.14.3.1 cameraFPS

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

6.14.3.2 frameHeight_px

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

6.14.3.3 frameWidth_px

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

6.14.3.4 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](#)

6.15 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](#)

6.15.1 Constructor & Destructor Documentation

6.15.1.1 Int128() [1/3]

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

6.15.1.2 Int128() [2/3]

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

6.15.1.3 Int128() [3/3]

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

6.15.2 Member Function Documentation

6.15.2.1 operator "!="()

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

6.15.2.2 operator -()

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


6.15.2.3 operator -=()

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

6.15.2.4 operator >()

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

6.15.2.5 operator >=()

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

6.15.2.6 operator double()

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

6.15.2.7 operator+()

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

6.15.2.8 operator+=()

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

6.15.2.9 operator-()

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

6.15.2.10 operator<()

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

6.15.2.11 operator<=()

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

6.15.2.12 operator=()

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

6.15.2.13 operator==()

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

6.15.3 Member Data Documentation

6.15.3.1 hi

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

6.15.3.2 lo

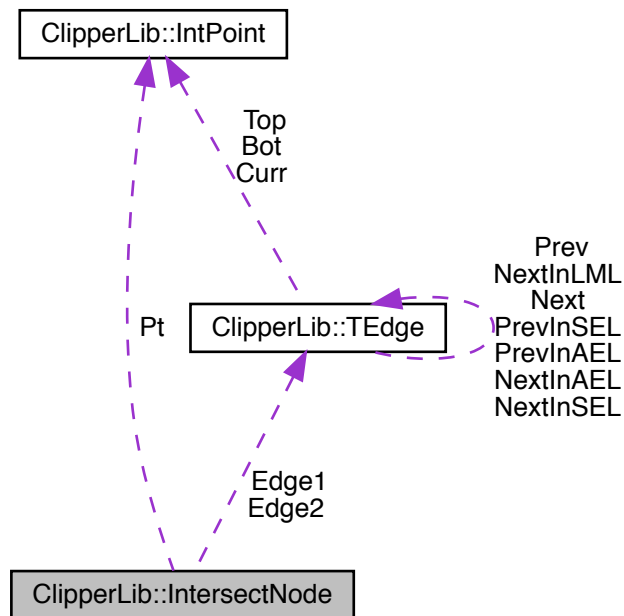
```
ulong64 ClipperLib::Int128::lo
```

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

- [src/clipper.cc](#)

6.16 ClipperLib::IntersectNode Struct Reference

Collaboration diagram for ClipperLib::IntersectNode:



Public Attributes

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

6.16.1 Member Data Documentation

6.16.1.1 Edge1

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

6.16.1.2 Edge2

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

6.16.1.3 Pt

`IntPoint` ClipperLib::IntersectNode::Pt

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

- [src/clipper.cc](#)

6.17 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)

6.17.1 Constructor & Destructor Documentation

6.17.1.1 IntPoint()

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

6.17.2 Friends And Related Function Documentation

6.17.2.1 operator!=

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

6.17.2.2 operator==

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

6.17.3 Member Data Documentation

6.17.3.1 X

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

6.17.3.2 Y

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

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

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

6.18 ClipperLib::IntRect Struct Reference

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

Public Attributes

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

6.18.1 Member Data Documentation

6.18.1.1 bottom

`cInt` ClipperLib::IntRect::bottom

6.18.1.2 left

`cInt` ClipperLib::IntRect::left

6.18.1.3 right

`cInt` ClipperLib::IntRect::right

6.18.1.4 top

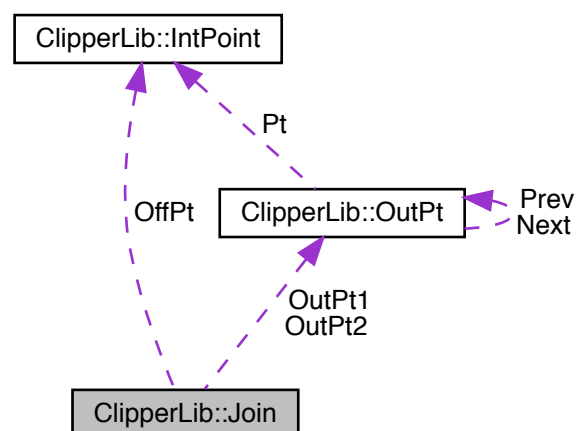
`cInt` ClipperLib::IntRect::top

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

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

6.19 ClipperLib::Join Struct Reference

Collaboration diagram for ClipperLib::Join:



Public Attributes

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

6.19.1 Member Data Documentation

6.19.1.1 OffPt

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

6.19.1.2 OutPt1

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

6.19.1.3 OutPt2

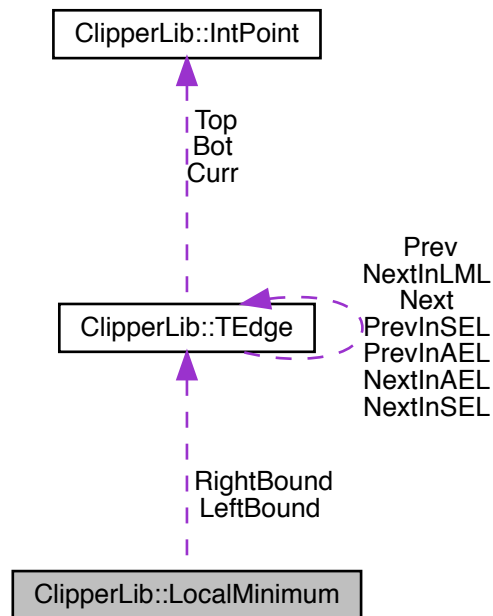
[OutPt](#)* [ClipperLib::Join::OutPt2](#)

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

- [src/clipper.cc](#)

6.20 ClipperLib::LocalMinimum Struct Reference

Collaboration diagram for ClipperLib::LocalMinimum:



Public Attributes

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

6.20.1 Member Data Documentation

6.20.1.1 LeftBound

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

6.20.1.2 RightBound

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

6.20.1.3 Y

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

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

- [src/clipper.cc](#)

6.21 ClipperLib::LocMinSorter Struct Reference

Public Member Functions

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

6.21.1 Member Function Documentation

6.21.1.1 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](#)

6.22 Mapp Class Reference

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

Public Member Functions

- [Mapp](#) (int _lengthX=1000, int _lengthY=1500, int _pixX=5, int _pixY=5, vector< vector< [Point2](#)< int > > > vvp=vector< vector< [Point2](#)< int > > >())
Constructor of the class.
- void [addObject](#) (vector< [Point2](#)< int > > vp, const [OBJ_TYPE](#) type)
Given an obstacle it is added to the map.
- 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.
- [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 > p1, const [Point2](#)< int > p2)
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.

Protected Member Functions

- `set< pair< int, int > > cellsFromSegment (Point2< int > p0, 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.

Protected Attributes

- `OBJ_TYPE ** map`
- `int lengthX`
- `int lengthY`
- `int dimX`
- `int dimY`
- `int pixX`
- `int pixY`

6.22.1 Constructor & Destructor Documentation

6.22.1.1 Mapp()

```
Mapp::Mapp (
    int _lengthX = 1000,
    int _lengthY = 1500,
    int _pixX = 5,
    int _pixY = 5,
    vector< vector< Point2< int > > > vvp = vector< vector<Point2<int> > >() )
```

Constructor of the class.

Parameters

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

6.22.2 Member Function Documentation

6.22.2.1 addObject()

```
void Mapp::addObject (
    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 id the type of the given object. Defined as a OBJ_TYPE.

6.22.2.2 cellsFromSegment()

```
set< pair< int, int > > Mapp::cellsFromSegment (
    Point2< int > p0,
    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).

6.22.2.3 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.

6.22.2.4 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.

6.22.2.5 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.

6.22.2.6 matrixToString()

```
string Mapp::matrixToString ( )
```

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

Returns

The generated string.

6.22.2.7 printDimensions()

```
void Mapp::printDimensions ( )
```

Print to the terminal the main informations of the Map.

6.22.2.8 printMap()

```
void Mapp::printMap ( )
```

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

6.22.3 Member Data Documentation

6.22.3.1 dimX

```
int Mapp::dimX [protected]
```

6.22.3.2 dimY

```
int Mapp::dimY [protected]
```

6.22.3.3 lengthX

```
int Mapp::lengthX [protected]
```

6.22.3.4 lengthY

```
int Mapp::lengthY [protected]
```

6.22.3.5 map

```
OBJ_TYPE** Mapp::map [protected]
```

6.22.3.6 pixX

```
int Mapp::pixX [protected]
```

6.22.3.7 pixY

```
int Mapp::pixY [protected]
```

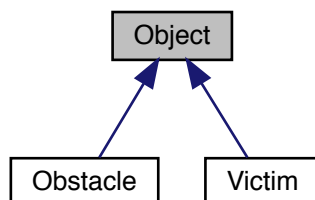
The documentation for this class was generated from the following files:

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

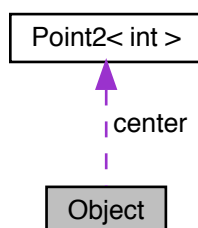
6.23 Object Class Reference

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

Inheritance diagram for Object:



Collaboration diagram for Object:



Public Member Functions

- string `toString` ()
Generate a string that describe the object.
- unsigned `size` ()
Return the number of points of the object.
- unsigned `nPoints` ()
Return the number of points 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)
Enlarge the object of the given offset.
- 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`
- float `radius`

6.23.1 Member Function Documentation

6.23.1.1 `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.

6.23.1.2 `computeRadius()`

```
void Object::computeRadius ( )
```

Compute the radius of the object.

This function assume that the center of the object is already computed and consistent.

6.23.1.3 `insidePoly()`

```
bool Object::insidePoly (
    Point2< int > pt )
```

Exact check if a point is inside the object (no approximation).

Parameters

<code>in</code>	<code>pt</code>	The point to be checked.
-----------------	-----------------	--------------------------

Returns

True if the point is inside the object, false otherwise.

6.23.1.4 insidePolyApprox()

```
bool Object::insidePolyApprox (
    Point2< int > pt )
```

Check if the given point is inside the approximation shape of the object (a circle).

Parameters

<code>in</code>	<code>pt</code>	The point to be checked.
-----------------	-----------------	--------------------------

Returns

True if the point is inside the object, false otherwise.

6.23.1.5 nPoints()

```
unsigned Object::nPoints ( )
```

Return the number of points of the object.

Returns

The number of points.

6.23.1.6 offsetting()

```
void Object::offsetting (
    const int offset )
```

Enlarge the object of the given offset.

The function automatically update even the center and the radius.

Parameters

<code>in</code>	<code>offset</code>	The size of the offset.
-----------------	---------------------	-------------------------

6.23.1.7 `size()`

```
unsigned Object::size ( )
```

Return the number of points of the object.

Returns

The number of points.

6.23.1.8 `toString()`

```
string Object::toString ( )
```

Generate a string that describe the object.

Returns

The generated string.

6.23.2 Member Data Documentation

6.23.2.1 `center`

```
Point2<int> Object::center [protected]
```

6.23.2.2 `points`

```
vector<Point2<int> > Object::points [protected]
```

6.23.2.3 radius

```
float Object::radius [protected]
```

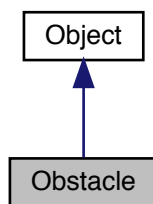
The documentation for this class was generated from the following files:

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

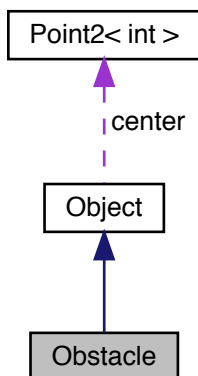
6.24 Obstacle Class Reference

```
#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

6.24.1 Constructor & Destructor Documentation

6.24.1.1 Obstacle()

```
Obstacle::Obstacle (  
    vector< Point2< int > > vp )
```

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

Parameters

<code>in</code>	<code>vp</code>	Vector of points that is the convex hull of the obstacle.
-----------------	-----------------	---

Returns

Return the created obstacle.

6.24.2 Member Function Documentation

6.24.2.1 print()

```
void Obstacle::print ( )
```

Print the describing string of the obstacle.

6.24.2.2 toString()

```
string Obstacle::toString ( )
```

Generate a string that describe the obstacle.

Returns

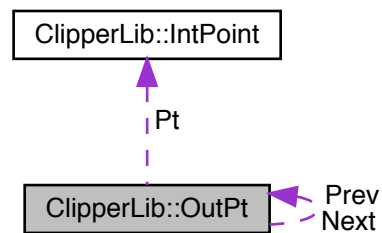
The generated string.

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

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

6.25 ClipperLib::OutPt Struct Reference

Collaboration diagram for ClipperLib::OutPt:



Public Attributes

- [int Idx](#)
- [IntPoint Pt](#)
- [OutPt * Next](#)
- [OutPt * Prev](#)

6.25.1 Member Data Documentation

6.25.1.1 Idx

```
int ClipperLib::OutPt::Idx
```

6.25.1.2 Next

```
OutPt* ClipperLib::OutPt::Next
```

6.25.1.3 Prev

```
OutPt* ClipperLib::OutPt::Prev
```

6.25.1.4 Pt

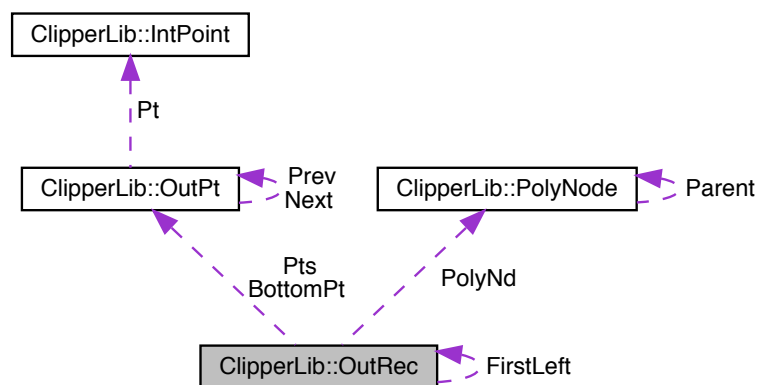
```
IntPoint ClipperLib::OutPt::Pt
```

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

- [src/clipper.cc](#)

6.26 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`

6.26.1 Member Data Documentation

6.26.1.1 BottomPt

`OutPt*` ClipperLib::OutRec::BottomPt

6.26.1.2 FirstLeft

`OutRec*` ClipperLib::OutRec::FirstLeft

6.26.1.3 Idx

`int` ClipperLib::OutRec::Idx

6.26.1.4 IsHole

`bool` ClipperLib::OutRec::IsHole

6.26.1.5 IsOpen

`bool` ClipperLib::OutRec::IsOpen

6.26.1.6 PolyNd

`PolyNode*` ClipperLib::OutRec::PolyNd

6.26.1.7 Pts

`OutPt*` ClipperLib::OutRec::Pts

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

- [src/clipper.cc](#)

6.27 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 taked to 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
- int x (const T _x)
Set the abscissa value.
- int y (const T _y)
Set the ordinate value.
- template<class T1 >
 [int offset](#) (const T1 _offset, 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.
- [int offset](#) (const [Point2](#)< T > p)
This function compute an offset given another point made of the abscissa offset and the ordinate offset.
- [int offset](#) (const [Tuple](#)< T > p)
This function compute an offset given a [Tuple](#) made of the abscissa offset and the ordinate offset.
- [int offset_x](#) (const T _offset)
This function compute an offset for the abscissa.
- [int 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. \tparan 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. \tparan 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. \tparan 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 = operatore. 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)`

Friends

- `ostream & operator<< (ostream &out, const Point2< T > &data)`

Overload of operator << to output the content of a `Point2`.

6.27.1 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

<code>T</code>	The type of the coordinates to be stored.
----------------	---

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Point2() [1/3]

```
template<class T>
Point2< T >::Point2 ( ) [inline]
```

Default constructor to build an empty [Tuple](#).

6.27.2.2 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	\leftrightarrow $\xleftarrow{-}$ x	The abscissa coordinate.
in	\leftrightarrow $\xleftarrow{-}$ y	The ordinate coordinate.

6.27.2.3 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	p	The cv::Point to be copied.
----	-----	-----------------------------

6.27.3 Member Function Documentation

6.27.3.1 copy()

```
template<class T>
Point2<T> Point2< T >::copy (
    const Point2< T > & A ) [inline]
```

Copy a point into another one.

Parameters

in	A	point to be copied.
----	-----	---------------------

Returns

this.

6.27.3.2 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.

6.27.3.3 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.

6.27.3.4 EuDistance()

```
template<class T>
template<class T1 >
double Point2< T >::EuDistance (
    Point2< T1 > B ) [inline]
```

Function that compute the Euclidean Distance between two points. \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.
----	----------	--

Returns

The Euclidean distance between the two points.

6.27.3.5 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	<i>B</i>	the second Point2 to use for computing the distance.
----	----------	--

Returns

The Manhattan distance between the two points.

6.27.3.6 offset() [1/3]

```
template<class T>
template<class T1 >
int 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

--	--

6.27.3.7 offset() [2/3]

```
template<class T>
int 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.

6.27.3.8 offset() [3/3]

```
template<class T>
int 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	p	The Tuple with the offsets. Its dimension must be 2.
----	-----	--

Returns

1 if everything went fine, 0 otherwise.

6.27.3.9 offset_x()

```
template<class T>
int Point2< T >::offset_x (
    const T _offset ) [inline]
```

This function compute an offset for the abscissa.

Parameters

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

Returns

1 if everything went fine, 0 otherwise.

6.27.3.10 offset_y()

```
template<class T>
int Point2< T >::offset_y (
    const T _offset ) [inline]
```

This function compute an offset for the ordinate.

Parameters

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

Returns

1 if everything went fine, 0 otherwise.

6.27.3.11 operator cv::Point()

```
template<class T>
Point2< T >::operator cv::Point ( ) const [inline]
```

Cast to cv::Point.

Returns

The value casted to point

6.27.3.12 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	<i>A</i>	point to be compared to.
----	----------	--------------------------

Returns

true if the two configurations are different.

6.27.3.13 operator<()

```
template<class T>
bool Point2< T >::operator< (
    const Point2< T > & A ) [inline]
```

6.27.3.14 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.

6.27.3.15 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.

6.27.3.16 to_string()

```
template<class T>
stringstream Point2< T >::to_string ( ) const [inline]
```

6.27.3.17 x() [1/2]

```
template<class T>
T Point2< T >::x ( ) const [inline]
```

Returns

The abscissa coordinate

6.27.3.18 x() [2/2]

```
template<class T>
int Point2< T >::x (
    const T _x ) [inline]
```

Set the abscissa value.

Parameters

in	↔	The new abscissa value
	↔	
	x	

Returns

1 if it was successful, 0 otherwise.

6.27.3.19 y() [1/2]

```
template<class T>
T Point2< T >::y ( ) const [inline]
```

Returns

The ordinate coordinate

6.27.3.20 `y()` [2/2]

```
template<class T>
int 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.

6.27.4 Friends And Related Function Documentation

6.27.4.1 `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 <code>Point2</code> to print.

Returns

An output stream to be printed.

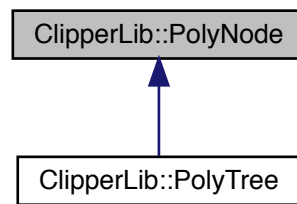
The documentation for this class was generated from the following file:

- `src/include/maths.hh`

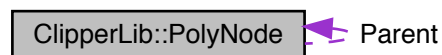
6.28 ClipperLib::PolyNode Class Reference

```
#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](#)

6.28.1 Constructor & Destructor Documentation

6.28.1.1 PolyNode()

```
ClipperLib::PolyNode::PolyNode ( )
```

6.28.1.2 ~PolyNode()

```
virtual ClipperLib::PolyNode::~~PolyNode ( ) [inline], [virtual]
```

6.28.2 Member Function Documentation

6.28.2.1 ChildCount()

```
int ClipperLib::PolyNode::ChildCount ( ) const
```

6.28.2.2 GetNext()

```
PolyNode * ClipperLib::PolyNode::GetNext ( ) const
```

6.28.2.3 IsHole()

```
bool ClipperLib::PolyNode::IsHole ( ) const
```

6.28.2.4 IsOpen()

```
bool ClipperLib::PolyNode::IsOpen ( ) const
```

6.28.3 Friends And Related Function Documentation

6.28.3.1 Clipper

```
friend class Clipper [friend]
```

6.28.3.2 ClipperOffset

```
friend class ClipperOffset [friend]
```

6.28.4 Member Data Documentation

6.28.4.1 Childs

[PolyNodes](#) ClipperLib::PolyNode::Childs

6.28.4.2 Contour

[Path](#) ClipperLib::PolyNode::Contour

6.28.4.3 Parent

[PolyNode*](#) ClipperLib::PolyNode::Parent

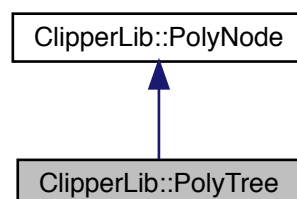
The documentation for this class was generated from the following files:

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

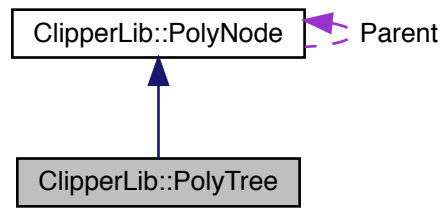
6.29 ClipperLib::PolyTree Class Reference

```
#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

6.29.1 Constructor & Destructor Documentation

6.29.1.1 ~PolyTree()

```
ClipperLib::PolyTree::~~PolyTree ( ) [inline]
```

6.29.2 Member Function Documentation

6.29.2.1 Clear()

```
void ClipperLib::PolyTree::Clear ( )
```

6.29.2.2 GetFirst()

```
PolyNode * ClipperLib::PolyTree::GetFirst ( ) const
```

6.29.2.3 Total()

```
int ClipperLib::PolyTree::Total ( ) const
```

6.29.3 Friends And Related Function Documentation

6.29.3.1 Clipper

```
friend class Clipper [friend]
```

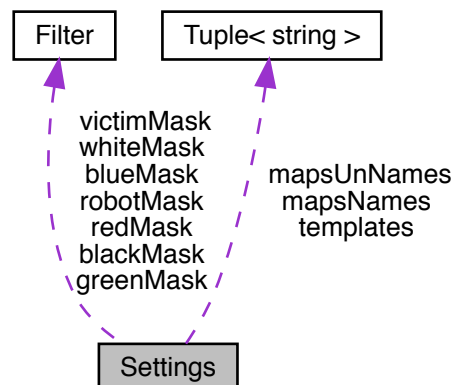
The documentation for this class was generated from the following files:

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

6.30 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)`

6.30.1 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.

6.30.2 Member Enumeration Documentation

6.30.2.1 COLOR

```
enum Settings::COLOR
```

Enumerator

BLACK	
RED	
GREEN	
VICTIMS	
BLUE	
WHITE	
ROBOT	

6.30.3 Constructor & Destructor Documentation

6.30.3.1 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.

6.30.3.2 ~Settings()

```
Settings::~~Settings ( )
```

Destructor.

6.30.4 Member Function Documentation

6.30.4.1 addUnMap()

```
bool Settings::addUnMap (
    string unMap )
```

6.30.4.2 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.

6.30.4.3 `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.

6.30.4.4 `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.

6.30.4.5 `cleanAndRead()`

```
void Settings::cleanAndRead (
    string _path = "data/settings.xml" )
```

Function to clean all settings and then read from file. Default is data/settings.xml.

6.30.4.6 `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.

6.30.4.7 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.

6.30.4.8 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.

6.30.4.9 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.

6.30.4.10 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.

6.30.4.11 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.

6.30.4.12 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.

6.30.4.13 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.
--------------------	--------------------------------

6.30.4.14 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.

6.30.4.15 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.

6.30.4.16 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 this.mapsUnNames) of the undistorted maps to be retrieved.
------------	---

Returns

A [Tuple](#) containing the paths of the undistorted maps.

6.30.4.17 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 this.mapsUnNames 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 this.mapsUnNames of the undistorted map to be retrieved
-----------	--

Returns

A [Tuple](#) containing the paths of the undistorted maps.

6.30.4.18 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.

6.30.4.19 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.

6.30.4.20 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.
--------------------	-----------------------------------

6.30.5 Friends And Related Function Documentation**6.30.5.1 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.

6.30.6 Member Data Documentation**6.30.6.1 blackMask**

`Filter Settings::blackMask`

`Filter` for black.

6.30.6.2 blueMask

`Filter Settings::blueMask`

`Filter` for blue.

6.30.6.3 calibrationFile

`string Settings::calibrationFile`

A string containing the path to the file containing the data for the calibration.

6.30.6.4 convexHullFile

`string Settings::convexHullFile`

AString containing the path to file containing the points of the elements in the arena.

6.30.6.5 greenMask

`Filter Settings::greenMask`

`Filter` for green.

6.30.6.6 intrinsicCalibrationFile

`string Settings::intrinsicCalibrationFile`

A string containing the path to the file containing the values of the matrix for the calibration.

6.30.6.7 kernelSide

`int Settings::kernelSide`

6.30.6.8 mapsFolder

`string Settings::mapsFolder`

A string containing the path for mapsFolder. No certainty is given about the form of this string.

6.30.6.9 mapsNames

`Tuple<string> Settings::mapsNames`

A `Tuple` containing the names of the maps. These are not paths but just names.

6.30.6.10 mapsUnNames

`Tuple<string> Settings::mapsUnNames`

A `Tuple` containing the names of the undistorted maps. These are not paths but just names.

6.30.6.11 redMask

`Filter Settings::redMask`

`Filter` for red.

6.30.6.12 robotMask

`Filter Settings::robotMask`

`Filter` for the triangle above the robot.

6.30.6.13 templates

`Tuple<string> Settings::templates`

A `Tuple` containing the names of the templates. These are not paths but just names.

6.30.6.14 templatesFolder

`string Settings::templatesFolder`

A String containing the path of the folder containing the number templates.

6.30.6.15 victimMask

`Filter Settings::victimMask`

`Filter` for the victims.

6.30.6.16 whiteMask

`Filter Settings::whiteMask`

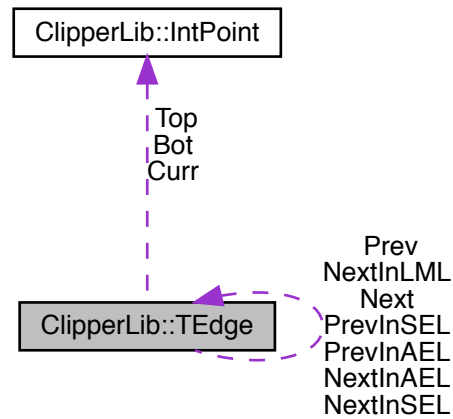
`Filter` for white.

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

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

6.31 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](#)

6.31.1 Member Data Documentation

6.31.1.1 Bot

`IntPoint` ClipperLib::TEdge::Bot

6.31.1.2 Curr

`IntPoint` ClipperLib::TEdge::Curr

6.31.1.3 Dx

`double` ClipperLib::TEdge::Dx

6.31.1.4 Next

`TEdge*` ClipperLib::TEdge::Next

6.31.1.5 NextInAEL

`TEdge*` ClipperLib::TEdge::NextInAEL

6.31.1.6 NextInLML

`TEdge*` ClipperLib::TEdge::NextInLML

6.31.1.7 NextInSEL

`TEdge*` ClipperLib::TEdge::NextInSEL

6.31.1.8 OutIdx

`int` ClipperLib::TEdge::OutIdx

6.31.1.9 PolyTyp

`PolyType` ClipperLib::TEdge::PolyTyp

6.31.1.10 Prev

`TEdge*` ClipperLib::TEdge::Prev

6.31.1.11 PrevInAEL

`TEdge*` ClipperLib::TEdge::PrevInAEL

6.31.1.12 PrevInSEL

`TEdge*` ClipperLib::TEdge::PrevInSEL

6.31.1.13 Side

`EdgeSide` ClipperLib::TEdge::Side

6.31.1.14 Top

`IntPoint` ClipperLib::TEdge::Top

6.31.1.15 WindCnt

`int` ClipperLib::TEdge::WindCnt

6.31.1.16 WindCnt2

`int` ClipperLib::TEdge::WindCnt2

6.31.1.17 WindDelta

```
int ClipperLib::TEdge::WindDelta
```

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

- [src/clipper.cc](#)

6.32 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.
- [int](#) size () const
- [T](#) [get](#) (const [int](#) _n) const
Gets the n-th element.
- void [add](#) (const T _new)
Adds a value at the end of the list.
- [int](#) [remove](#) (const T pos)
Removes a value from the list.
- [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.
- 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. \tparam 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. \tparam T1 The type of the elements in the second [Tuple](#).
- stringstream [to_string](#) (string _prefix="") const
- template<class T1 >
[operator](#) [vector](#)< T1 > () const
Overload of cast to vector.
- [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.

6.32.1 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.
----------	------------------------------------

6.32.2 Constructor & Destructor Documentation

6.32.2.1 Tuple() [1/2]

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

Default constructor.

6.32.2.2 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.

Parameters

in	\leftrightarrow $_ \leftarrow$ <i>n</i>	Number of objects to store.
in	...	Objects to store.

6.32.3 Member Function Documentation

6.32.3.1 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.
----	-------------------	----------------------------

6.32.3.2 begin() [1/2]

```
template<class T>
tupleIter Tuple< T >::begin ( ) [inline]
```

Iterator.

Returns

the elements.begin() iterator.

6.32.3.3 begin() [2/2]

```
template<class T>
tupleConstIter Tuple< T >::begin ( ) const [inline]
```

Const iterator.

Returns

the elements.begin() iterator.

6.32.3.4 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](#).

6.32.3.5 end() [1/2]

```
template<class T>
tupleIter Tuple< T >::end ( ) [inline]
```

Iterator.

Returns

the elements.end() iterator.

6.32.3.6 end() [2/2]

```
template<class T>
tupleConstIter Tuple< T >::end ( ) const [inline]
```

Const iterator.

Returns

the elements.begin() iterator.

6.32.3.7 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](#).

6.32.3.8 get()

```
template<class T>
T Tuple< T >::get (
    const int _n ) const [inline]
```

Gets the n-th element.

Parameters

in	\leftrightarrow \leftarrow <i>n</i>	The position of the element to retrieve.
----	---	--

Returns

The element in the n-th position or -1 if _n is greater than n or less than 0.

6.32.3.9 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

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

Returns

The Manhattan distance between the two [Tuple](#).

6.32.3.10 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.

6.32.3.11 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.

6.32.3.12 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.

6.32.3.13 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.

6.32.3.14 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.

6.32.4 Friends And Related Function Documentation

6.32.4.1 operator<<

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

Overload of operator << to output the content of the tuple.

Parameters

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

Returns

An output stream to be printed.

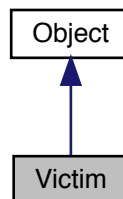
The documentation for this class was generated from the following file:

- `src/include/maths.hh`

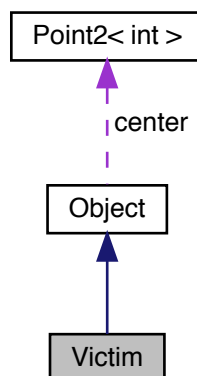
6.33 Victim Class Reference

```
#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](#)

6.33.1 Constructor & Destructor Documentation

6.33.1.1 Victim()

```
Victim::Victim (
    vector< Point2< int > > vp,
    int _value )
```

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

Parameters

in	<i>vp</i>	Vector of points that is the convex hull of the victim.
in	<i>_value</i>	The representative number of the victim.

Returns

Return the created victim.

6.33.2 Member Function Documentation

6.33.2.1 getValue()

```
int Victim::getValue ( ) [inline]
```

6.33.2.2 print()

```
void Victim::print ( )
```

Print the describing string of the victim.

6.33.2.3 setValue()

```
void Victim::setValue (
    int v ) [inline]
```


6.33.2.4 toString()

```
string Victim::toString ( )
```

Generate a string that describe the victim.

Returns

The generated string.

6.33.3 Member Data Documentation

6.33.3.1 value

```
int Victim::value [protected]
```

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

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

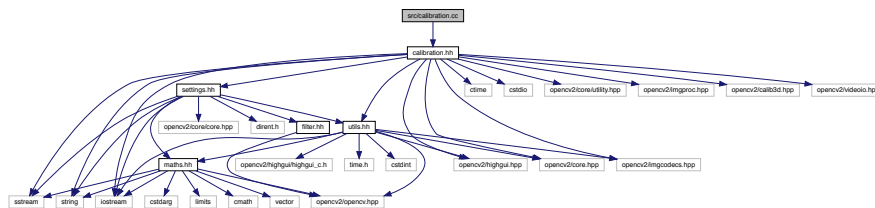
Chapter 7

File Documentation

7.1 src/calibration.cc File Reference

```
#include "calibration.hh"
```

Include dependency graph for calibration.cc:



Functions

- `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 distorsion 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.

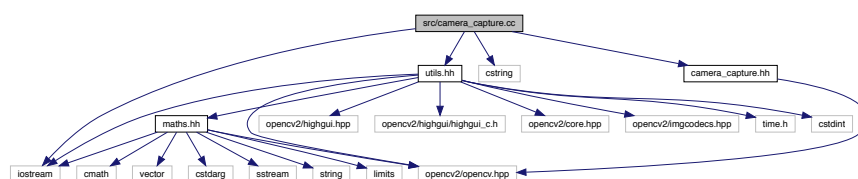
Open file for writing

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` [DEBUG](#)
- `#define` [SDEBUG](#)(X) { std::cout << X << std::endl; }

7.2.1 Macro Definition Documentation

7.2.1.1 DEBUG

```
#define DEBUG
```

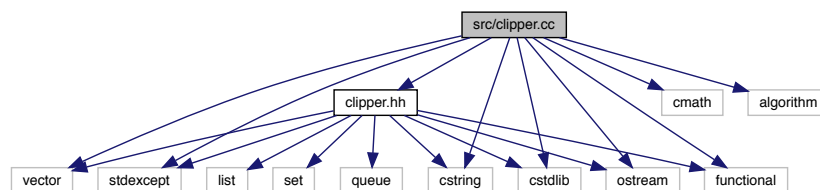
7.2.1.2 SDEBUG

```
#define SDEBUG(  
    X ) { std::cout << X << std::endl; }
```

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

- cInt [ClipperLib::Round](#) (double val)
- cInt [ClipperLib::Abs](#) (cInt 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)
- cInt [ClipperLib::TopX](#) (TEdge &edge, const cInt 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 clnt Y)
- bool [ClipperLib::IsIntermediate](#) (TEdge *e, const clnt 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, clnt &Left, clnt &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 clnt a1, const clnt a2, const clnt b1, const clnt b2, clnt &Left, clnt &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 pathIsClosed)
- void [ClipperLib::TranslatePath](#) (const Path &input, Path &output, const IntPoint delta)

- void [ClipperLib::MinkowskiSum](#) (const Path &pattern, const Paths &paths, Paths &solution, bool pathsClosed)
- 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))
```

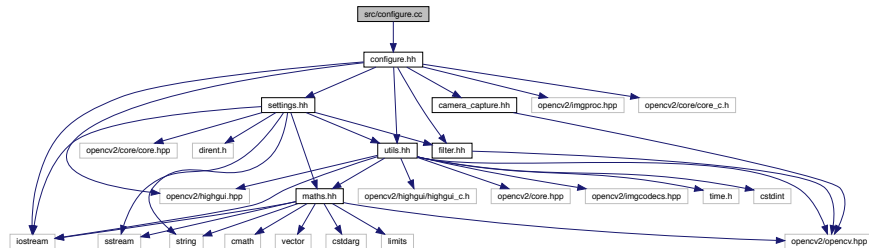
7.3.1.3 TOLERANCE

```
#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](#) ()
- 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, [Settings](#) *s)

Variables

- [Filter filter](#) = [Filter](#)(30, 30, 30, 100, 100, 100)

7.4.1 Function Documentation

7.4.1.1 [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.

7.4.1.2 on_high_h_thresh_trackbar()

```
void on_high_h_thresh_trackbar (
    int ,
    void * )
```

@function on_high_h_thresh_trackbar

7.4.1.3 on_high_s_thresh_trackbar()

```
void on_high_s_thresh_trackbar (
    int ,
    void * )
```

@function on_high_s_thresh_trackbar

7.4.1.4 on_high_v_thresh_trackbar()

```
void on_high_v_thresh_trackbar (
    int ,
    void * )
```

@function on_high_v_thresh_trackbar

7.4.1.5 on_low_h_thresh_trackbar()

```
void on_low_h_thresh_trackbar (
    int ,
    void * )
```

@function on_low_h_thresh_trackbar

7.4.1.6 on_low_s_thresh_trackbar()

```
void on_low_s_thresh_trackbar (
    int ,
    void * )
```

@function on_low_s_thresh_trackbar

7.4.1.7 on_low_v_thresh_trackbar()

```
void on_low_v_thresh_trackbar (
    int ,
    void * )
```

@function on_low_v_thresh_trackbar

7.4.1.8 show_all_conditions()

```
bool show_all_conditions (
    const Mat & frame,
    Settings * s )
```

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.
<i>s</i>	The Settings to use.

Returns

True if the filters are okay, false otherwise.

7.4.1.9 update_trackers()

```
void update_trackers ( )
```

Function to update trackers with filter

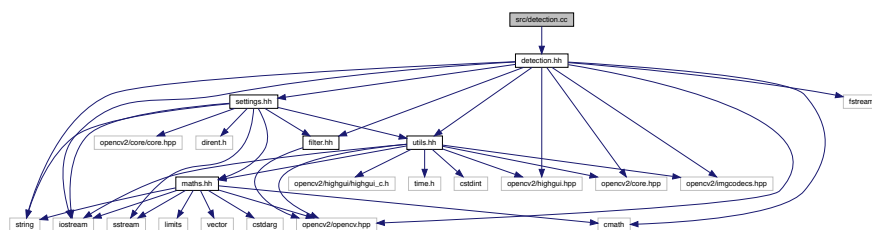
7.4.2 Variable Documentation**7.4.2.1 filter**

```
Filter filter = Filter(30, 30, 30, 100, 100, 100)
```

7.5 src/detection.cc File Reference

```
#include "detection.hh"
```

Include dependency graph for detection.cc:

**Macros**

- `#define EPS_CURVE 3`

Given an image, in black/white format, identify all the borders that delimit the shapes.

Functions

- [int detection](#) ()
Loads some images and detects shapes according to different colors.
- void [load_number_template](#) ()
Load some templates and save them in the global variable 'templates'.
- void [shape_detection](#) (const Mat &img, const [int](#) color, const Mat &un_img)
Detect shapes inside the image according to the variable 'color'.
- void [erode_dilation](#) (Mat &img, const [int](#) color)
It apply some filtering function for isolate the subject and remove the noise.
- void [find_contours](#) (const Mat &img, Mat original, const [int](#) 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 [int](#) color, const vector< [int](#) > &victims)
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](#)
- [Settings](#) * s =new [Settings](#)

7.5.1 Macro Definition Documentation

7.5.1.1 EPS_CURVE

```
#define EPS_CURVE 3
```

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>	Is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	Can has 3 value: 0 -> Red 1 -> Green 2 -> Blue Is used for decid which procedure apply to the image.

7.5.2 Function Documentation

7.5.2.1 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.
----------------	------------	---

7.5.2.2 detection()

```
int detection ( )
```

Loads some images and detects shapes according to different colors.

Returns

Return 0 if the function reach the end.

7.5.2.3 erode_dilation()

```
void erode_dilation (
    Mat & img,
    const int 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>	Can has 4 value: 0 -> Red 1 -> Green 2 -> Blue 3 -> Black According to the color the filtering functions apply can change in the type and in the order.

7.5.2.4 find_contours()

```
void find_contours (
    const Mat & img,
    Mat original,
    const int 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>	Is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	Can has 3 value: 0 -> Red 1 -> Green 2 -> Blue Is used for decid which procedure apply to the image.

7.5.2.5 load_number_template()

```
void load_number_template ( )
```

Load some templates and save them in the global variable 'templates'.

7.5.2.6 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.

7.5.2.7 save_convex_hull()

```
void save_convex_hull (
    const vector< vector< Point >> & contours,
    const int color,
    const vector< int > & victims )
```

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>	Is the parameter according to which the function decide if saved ('color==1') or not ('otherwise') the vector 'victims'.
in	<i>victims</i>	Is a vector that is saved in a xml file.

7.5.2.8 shape_detection()

```
void shape_detection (
    const Mat & img,
    const int color,
    const Mat & un_img )
```

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>	Can has 3 value: 0 -> Red 1 -> Green 2 -> Blue These color identify the possible spectrum that the function search on the image.

7.5.3 Variable Documentation

7.5.3.1 s

```
Settings* s =new Settings
```

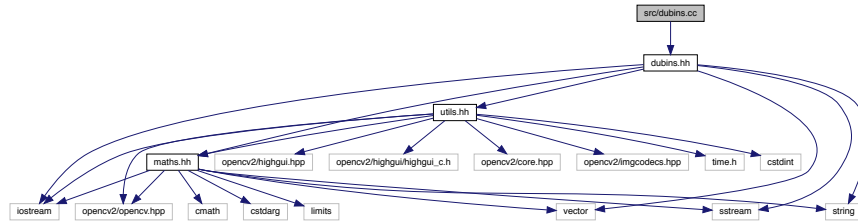
7.5.3.2 templates

```
vector<Mat> templates
```

7.6 src/dubins.cc File Reference

```
#include "dubins.hh"
```

Include dependency graph for dubins.cc:

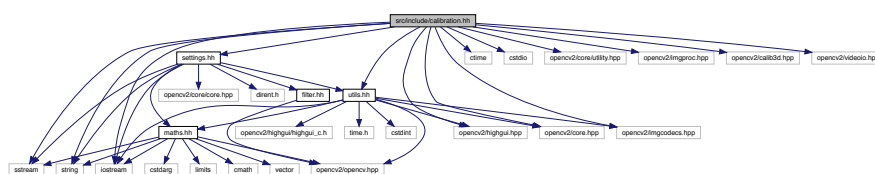


7.7 src/include/calibration.hh File Reference

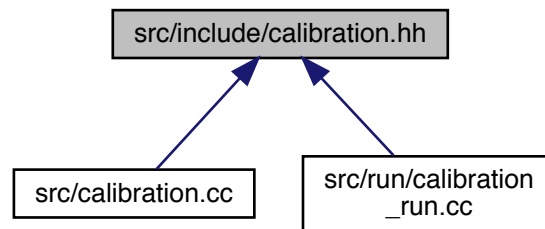
Library for calibration.

```
#include <utils.hh>
#include <settings.hh>
#include <iostream>
#include <sstream>
#include <string>
#include <ctime>
#include <cstdio>
#include <opencv2/core.hpp>
#include <opencv2/core/utility.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/calib3d.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/videoio.hpp>
#include <opencv2/highgui.hpp>
```

Include dependency graph for calibration.hh:



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



Classes

- class [CalSettings](#)

Enumerations

- enum { [DETECTION](#) = 0, [CAPTURING](#) = 1, [CALIBRATED](#) = 2 }

Functions

- [int calibration](#) (string inputFile="")
Function to run the complete calibration.
- [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](#).

7.7.1 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.

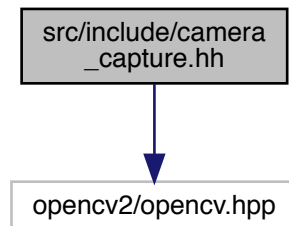
Returns

`true` if the calibration succeeded.
`false` otherwise.

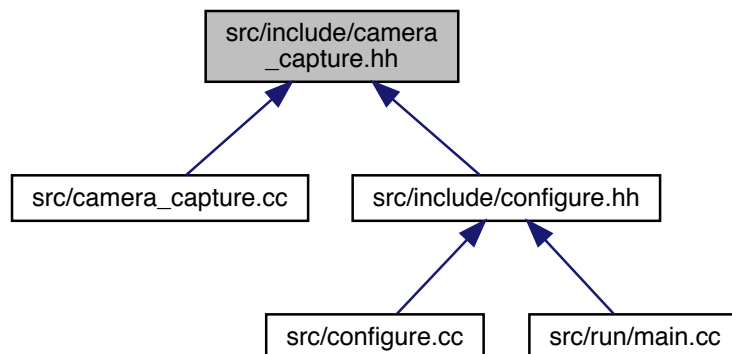
7.8 src/include/camera_capture.hh File Reference

```
#include <opencv2/opencv.hpp>
```

Include dependency graph for camera_capture.hh:



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

**Classes**

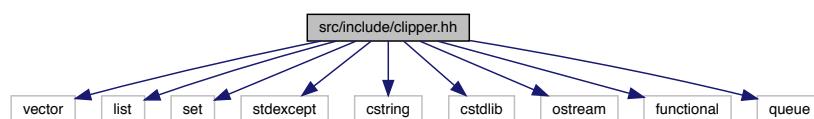
- class [CameraCapture](#)
- struct [CameraCapture::input_options_t](#)

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

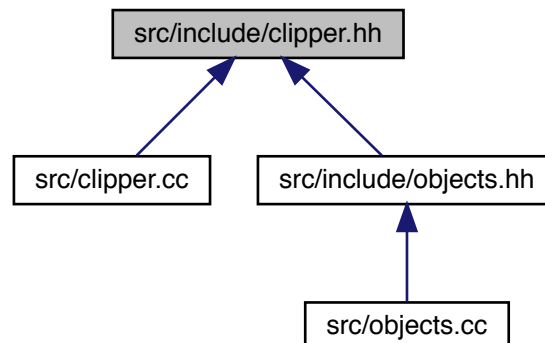
7.9 src/include/clipper.hh File Reference

```
#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

7.9.1.1 CLIPPER_VERSION

```
#define CLIPPER_VERSION "6.4.2"
```

7.9.1.2 use_lines

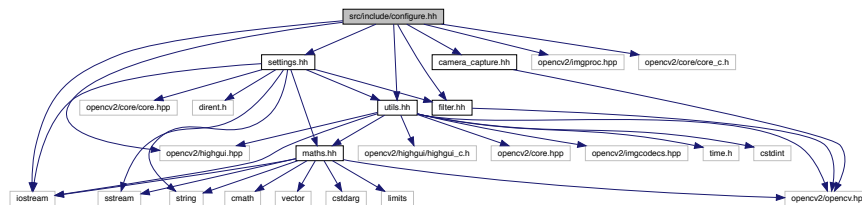
```
#define use_lines
```

7.10 src/include/configure.hh File Reference

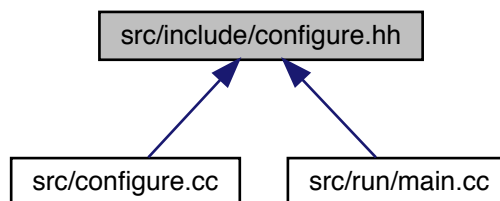
```
#include <iostream>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/core/core_c.h>
#include <utils.hh>
#include <filter.hh>
#include <camera_capture.hh>
```

```
#include <settings.hh>
```

Include dependency graph for `configure.hh`:



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



Functions

- 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, [Settings](#) *s)

7.10.1 Function Documentation

7.10.1.1 `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.

7.10.1.2 show_all_conditions()

```
bool show_all_conditions (
    const Mat & frame,
    Settings * s )
```

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.
<i>s</i>	The Settings to use.

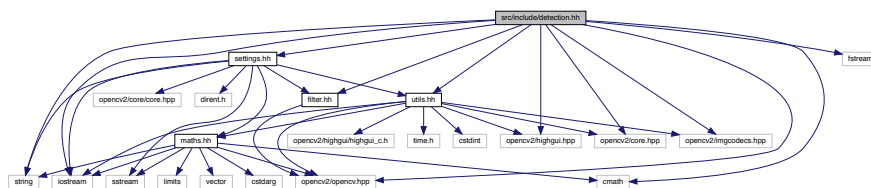
Returns

True if the filters are okay, false otherwise.

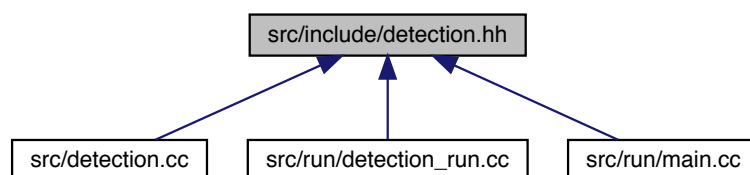
7.11 src/include/detection.hh File Reference

```
#include <utils.hh>
#include <settings.hh>
#include <filter.hh>
#include <iostream>
#include <fstream>
#include <string>
#include <cmath>
#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:



Functions

- `int detection ()`
Loads some images and detects shapes according to different colors.
- `void shape_detection (const Mat &img, const int color, const Mat &un_img)`
Detect shapes inside the image according to the variable 'color'.
- `void erode_dilation (Mat &img, const int color)`
It apply some filtering function for isolate the subject and remove the noise.
- `void find_contours (const Mat &img, Mat original, const int 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 int color, const vector< int > &victims)`
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 Function Documentation

7.11.1.1 crop_number_section()

```
void crop_number_section (
    Mat & ROI )
```

Given an image identify the region of interest(ROI) and crop it out.

Parameters

<code>in, out</code>	<code>ROI</code>	Is the image that the function will going to elaborate.
----------------------	------------------	---

7.11.1.2 detection()

```
int detection ( )
```

Loads some images and detects shapes according to different colors.

Returns

Return 0 if the function reach the end.

7.11.1.3 erode_dilation()

```
void erode_dilation (
    Mat & img,
    const int 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>	Can has 4 value: 0 -> Red 1 -> Green 2 -> Blue 3 -> Black According to the color the filtering functions apply can change in the type and in the order.

7.11.1.4 find_contours()

```
void find_contours (
    const Mat & img,
    Mat original,
    const int 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>	Is the original source of 'img', it is used for showing the detected contours.
in	<i>color</i>	Can has 3 value: 0 -> Red 1 -> Green 2 -> Blue Is used for decid which procedure apply to the image.

7.11.1.5 load_number_template()

```
void load_number_template ( )
```

Load some templates and save them in the global variable 'templates'.

7.11.1.6 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.

7.11.1.7 save_convex_hull()

```
void save_convex_hull (
    const vector< vector< Point >> & contours,
    const int color,
    const vector< int > & victims )
```

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>	Is the parameter according to which the function decide if saved ('color==1') or not ('otherwise') the vector 'victims'.
in	<i>victims</i>	Is a vector that is saved in a xml file.

7.11.1.8 shape_detection()

```
void shape_detection (
    const Mat & img,
    const int color,
    const Mat & un_img )
```

Detect shapes inside the image according to the variable 'color'.

Parameters

in	<i>img</i>	Image on which the research will done.
----	------------	--

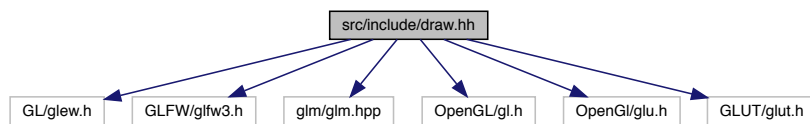
Parameters

<code>in</code>	<code>color</code>	Can has 3 value: 0 -> Red 1 -> Green 2 -> Blue 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

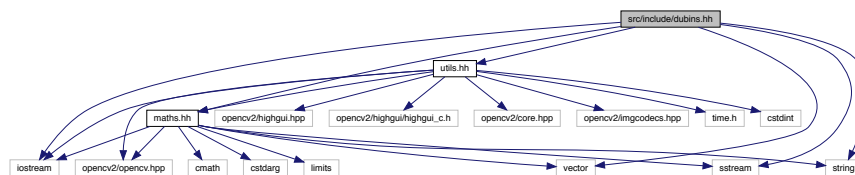
7.12.1.1 int

```
typedef uint unsigned int
```

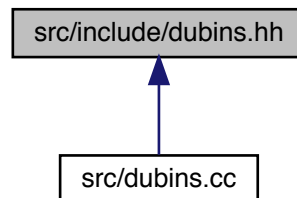
7.13 src/include/dubins.hh File Reference

```
#include "maths.hh"
#include "utils.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 [Dubins](#)< T >

Macros

- [#define](#) [MORE_FUNCTIONS](#)
- [#define](#) [PIECE_LENGTH](#) 2
- [#define](#) [KMAX](#) 1.0

Functions

- static double [sinc](#) (double t)
- [Configuration2](#)< double > [circline](#) (double _L, [Configuration2](#)< double > _P0, double _K)

7.13.1 Macro Definition Documentation

7.13.1.1 KMAX

```
#define KMAX 1.0
```

7.13.1.2 MORE_FUNCTIONS

```
#define MORE_FUNCTIONS
```

7.13.1.3 PIECE_LENGTH

```
#define PIECE_LENGTH 2
```

7.13.2 Function Documentation

7.13.2.1 [circline\(\)](#)

```
Configuration2<double> circline (  
    double _L,  
    Configuration2< double > _P0,  
    double _K )
```

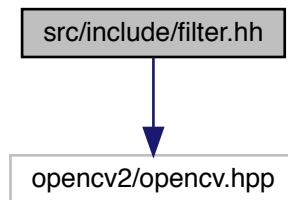
7.13.2.2 [sinc\(\)](#)

```
static double sinc (  
    double t ) [static]
```

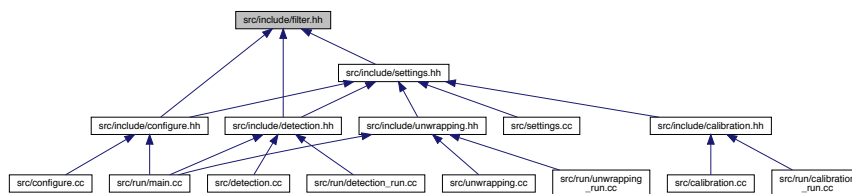
7.14 src/include/filter.hh File Reference

```
#include <opencv2/opencv.hpp>
```

Include dependency graph for filter.hh:



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



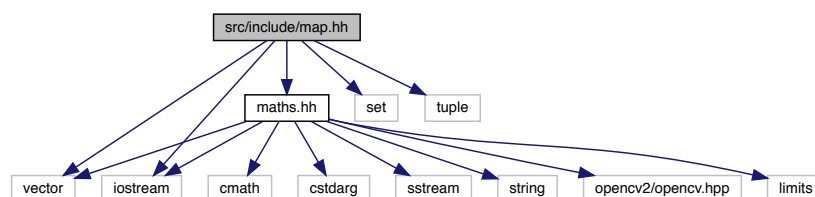
Classes

- class [Filter](#)

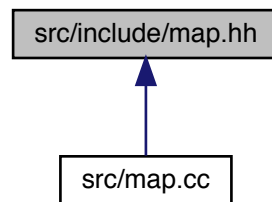
7.15 src/include/map.hh File Reference

```
#include <vector>
#include <set>
#include <tuple>
#include <iostream>
#include <maths.hh>
```

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](#) }

7.15.1 Enumeration Type Documentation

7.15.1.1 OBJ_TYPE

enum [OBJ_TYPE](#)

Enumerator

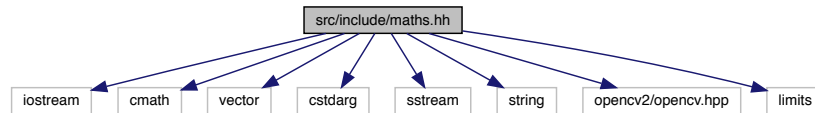
FREE	
VICT	
OBST	
GATE	

7.16 src/include/maths.hh File Reference

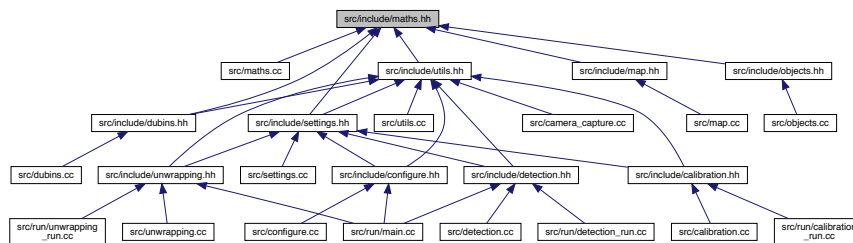
```
#include <iostream>
#include <cmath>
#include <vector>
#include <cstdarg>
#include <sstream>
```

```
#include <string>
#include <opencv2/opencv.hpp>
#include <limits>
```

Include dependency graph for maths.hh:



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



Classes

- class [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.
- class [Tuple< T >](#)
- class [Point2< T >](#)
Class that stores two value to construct a point in 2D. The value is saved in a [Tuple](#).
- class [Configuration2< T1 >](#)
This class stores a configuration, that is a point and an angle.

Macros

- #define [DInf](#) numeric_limits<double>::infinity()
- #define [Epsi](#) numeric_limits<double>::epsilon()
- #define [DEGTORAD](#) M_PI/180
- #define [RADTODEG](#) 180/M_PI
- #define [tupleIter](#) typename vector<T>::iterator
- #define [tupleConstIter](#) const typename vector<T>::iterator

Enumerations

- enum [DISTANCE_TYPE](#) { [EUCLIDEAN](#), [MANHATTAN](#) }

Functions

- bool `equal` (const double &A, const double &B, const double E=`Epsi`)
Function to compare two dubles as $|A - B| < \varepsilon$.
- template<class T >
T `pow2` (const T x)

Variables

- const `Angle A_2PI` = `Angle`(6.283185, `Angle::RAD`)
Default `Angle` for 2pi rad.
- const `Angle A_360` = `Angle`(360.0-`Epsi`, `Angle::DEG`)
Default `Angle` for 360 degree.
- const `Angle A_PI` = `Angle`(M_PI, `Angle::RAD`)
Default `Angle` for pi rad.
- const `Angle A_180` = `Angle`(180, `Angle::DEG`)
Default `Angle` for 180 degree.

7.16.1 Macro Definition Documentation

7.16.1.1 DEGTORAD

```
#define DEGTORAD M_PI/180
```

7.16.1.2 DInf

```
#define DInf numeric_limits<double>::infinity()
```

7.16.1.3 Epsi

```
#define Epsi numeric_limits<double>::epsilon()
```

7.16.1.4 RADTO DEG

```
#define RADTO DEG 180/M_PI
```

7.16.1.5 tupleConstIter

```
#define tupleConstIter const typename vector<T>::iterator
```

7.16.1.6 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 A_180**

```
const Angle A_180 = Angle(180, Angle::DEG)
```

Default [Angle](#) for 180 degree.

7.16.4.2 A_2PI

```
const Angle A_2PI = Angle(6.283185, Angle::RAD)
```

Default [Angle](#) for 2pi rad.

7.16.4.3 A_360

```
const Angle A_360 = Angle(360.0-Epsi, Angle::DEG)
```

Default [Angle](#) for 360 degree.

7.16.4.4 A_PI

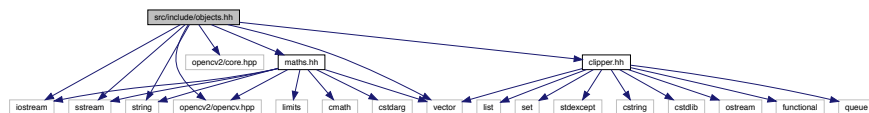
```
const Angle A_PI = Angle(M_PI, Angle::RAD)
```

Default [Angle](#) for pi rad.

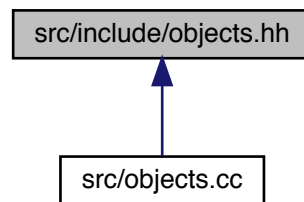
7.17 src/include/objects.hh File Reference

```
#include <iostream>
#include <vector>
#include <sstream>
#include <string>
#include <opencv2/core.hpp>
#include <opencv2/opencv.hpp>
#include "clipper.hh"
#include "maths.hh"
```

Include dependency graph for objects.hh:



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



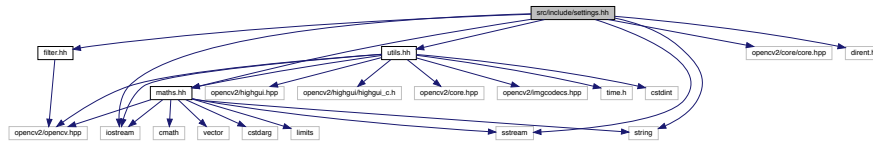
Classes

- class [Object](#)
- class [Obstacle](#)
- class [Victim](#)

7.18 src/include/settings.hh File Reference

```
#include <filter.hh>
#include <maths.hh>
#include <utils.hh>
#include <opencv2/core/core.hpp>
#include <iostream>
#include <string>
#include <dirent.h>
```


Include dependency graph for settings.hh:



```

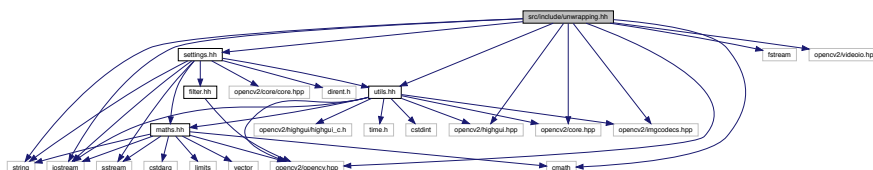
graph TD
    settings_hh["src/include/settings.hh"]
    settings_cc["src/settings.cc"]
    calibration_hh["src/include/calibration.hh"]
    calibration_cc["src/calibration.cc"]
    configure_hh["src/include/configure.hh"]
    configure_cc["src/configure.cc"]
    detection_hh["src/include/detection.hh"]
    detection_cc["src/detection.cc"]
    unwrapping_hh["src/include/unwrapping.hh"]
    unwrapping_cc["src/unwrapping.cc"]
    calibration_run_cc["src/run/calibration_run.cc"]
    main_cc["src/run/main.cc"]
    detection_run_cc["src/run/detection_run.cc"]
    unwrapping_run_cc["src/run/unwrapping_run.cc"]

    settings_hh --> settings_cc
    settings_hh --> calibration_hh
    settings_hh --> configure_hh
    settings_hh --> detection_hh
    settings_hh --> unwrapping_hh
    settings_cc --> calibration_cc
    calibration_hh --> calibration_run_cc
    configure_hh --> configure_cc
    configure_hh --> main_cc
    detection_hh --> main_cc
    detection_hh --> detection_cc
    detection_hh --> detection_run_cc
    detection_hh --> unwrapping_cc
    unwrapping_hh --> detection_run_cc
    unwrapping_hh --> unwrapping_run_cc
  
```

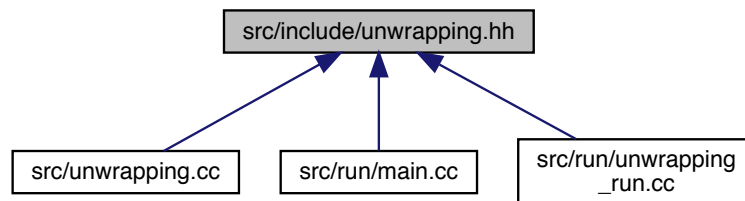
- class `Settings`

- class Settings

Include dependency graph for unwrapping.hh:



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



Functions

- `int unwrapping ()`
Take some images according to a xml and unwrap the black rectangle inside the image after applying undistortion trasformation.
- `void loadCoefficients (const string filename, Mat &camera_matrix, Mat &dist_coeffs)`
Load coefficients from a file.
- `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.

7.19.1 Function Documentation

7.19.1.1 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 vactor of cv::Point to work on.
in	<i>width</i>	The width of the image.
in	<i>height</i>	The height of the image.

7.19.1.2 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.

7.19.1.3 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 the 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

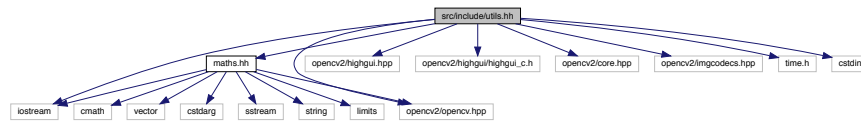
A 0 is return if the function reach the end.

7.20 src/include/utils.hh File Reference

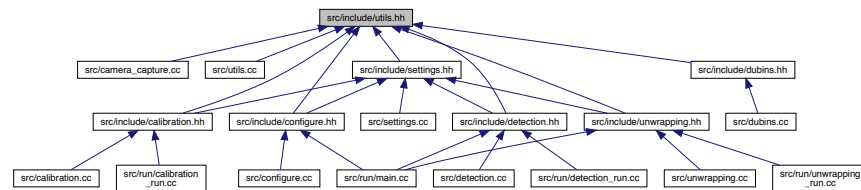
```
#include <maths.hh>
#include <iostream>
#include <opencv2/highgui.hpp>
#include <opencv2/highgui/highgui_c.h>
#include <opencv2/core.hpp>
#include <opencv2/opencv.hpp>
#include <opencv2/imgcodecs.hpp>
#include <time.h>
```

```
#include <stdint>
```

Include dependency graph for `utils.hh`:



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



Namespaces

- [timeutils](#)

Macros

- `#define NAME(x) #x`
Returns the name of the variable.
- `#define COUT(x)`
Print a message to stderr.
- `#define INFO(msg)`
Print the name of a variable and its content. Only if `DEBUG` is defined.

Functions

- `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.
- `void mywaitkey (Tuple< string > windowNames)`
Function to use after `my_imshow()` for keeping the image opened until a key is pressed. When a key is pressed some windows are closed.
- `int64_t timeutils::timespecDiff (struct timespec *timeA_p, struct timespec *timeB_p)`
- `double timeutils::getTimeS ()`

7.20.1 Macro Definition Documentation

7.20.1.1 COUT

```
#define COUT(  
    x )
```

Print a messag to stderr.

7.20.1.2 INFO

```
#define INFO(  
    msg )
```

Print the name of a variable and its content. Only if DEBUG is defined.

7.20.1.3 NAME

```
#define NAME(  
    x ) #x
```

Returns the name of the variable.

7.20.2 Function Documentation

7.20.2.1 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.

7.20.2.2 mywaitkey() [1/3]

```
void mywaitkey ( )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed.

7.20.2.3 mywaitkey() [2/3]

```
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.
-------------------	---

7.20.2.4 mywaitkey() [3/3]

```
void mywaitkey (
    Tuple< string > windowNames )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed. When a key is pressed some windows are closed.

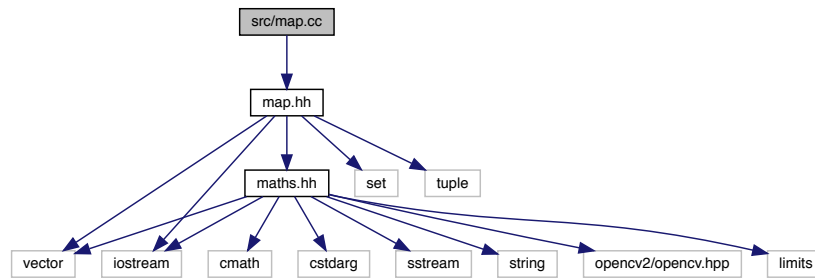
Parameters

<i>windowNames</i>	The names of the windows to close after pressing a key.
--------------------	---

7.21 src/map.cc File Reference

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

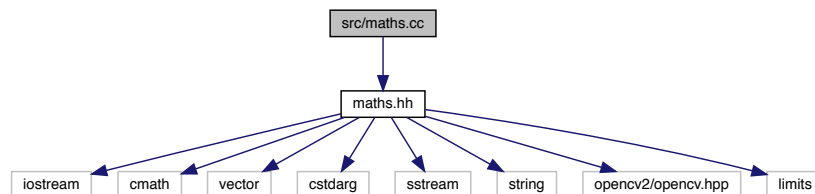
Include dependency graph for map.cc:



7.22 src/math.cc File Reference

```
#include "maths.hh"
```

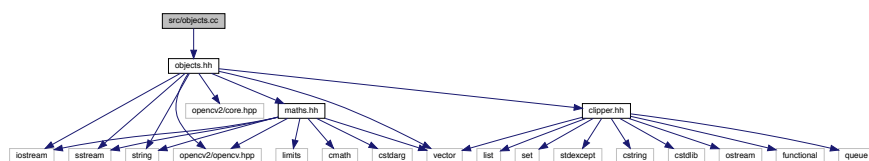
Include dependency graph for maths.cc:



7.23 src/objects.cc File Reference

```
#include "objects.hh"
```

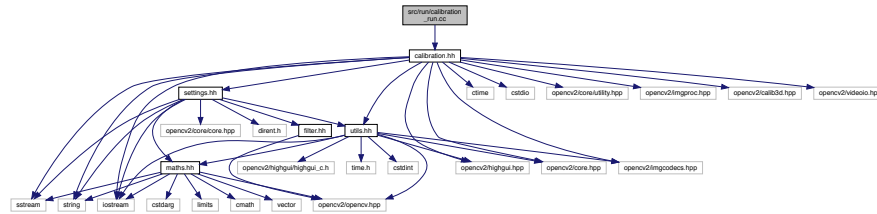
Include dependency graph for objects.cc:



7.24 src/run/calibration_run.cc File Reference

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

Include dependency graph for calibration_run.cc:



Functions

- [int main\(\)](#)

7.24.1 Function Documentation

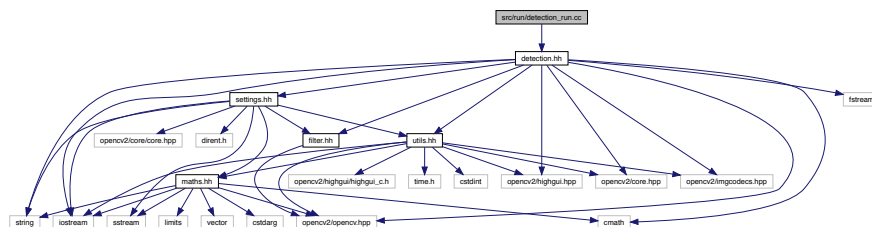
7.24.1.1 main()

```
int main ( )
```

7.25 src/run/detection_run.cc File Reference

```
#include <detection.hh>
```

Include dependency graph for detection_run.cc:



Functions

- [int main\(\)](#)

Functions

- `int main ()`

7.27.1 Function Documentation

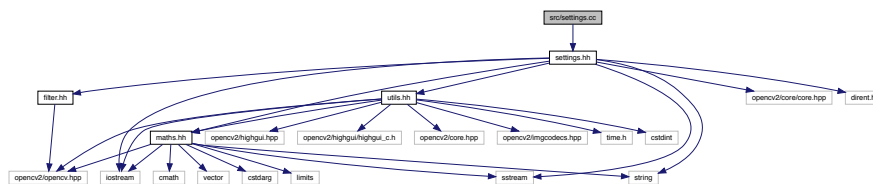
7.27.1.1 main()

```
int main ( )
```

7.28 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-l>
- `void vecToFile (FileStorage &fs, vector< int > x)`

7.28.1 Macro Definition Documentation

7.28.1.1 NPOS

```
#define NPOS string::npos
```

Shortcut for `string::npos`.

7.28.2 Function Documentation

7.28.2.1 getFiles()

```
vector<string> getFiles (
    const string & path )
```

Function to get all files in directory. From <https://stackoverflow.com/questions/612097/how-can-i-get-the>

Parameters

<i>Path</i>	The path to check.
-------------	--------------------

Returns

A vector containing the names of the files in the directory.

7.28.2.2 vecToFile()

```
void vecToFile (
    FileStorage & fs,
    vector< int > x ) [inline]
```

Writes a vector to a file.

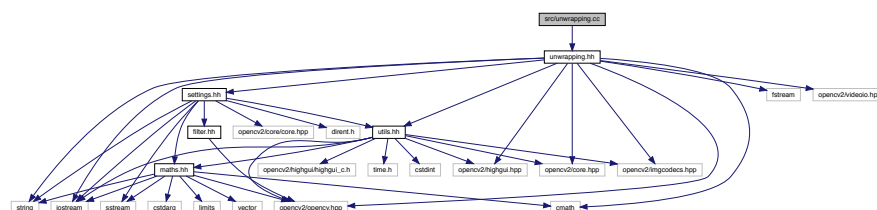
Parameters

<i>fs</i>	The FileStorage where to write the vector.
<i>x</i>	The vector to write.

7.29 src/unwrapping.cc File Reference

```
#include "unwrapping.hh"
```

Include dependency graph for unwrapping.cc:



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.

7.29.1 Macro Definition Documentation

7.29.1.1 AREA_MIN

```
#define AREA_MIN 500
```

7.29.1.2 AREA_RATIO

```
#define AREA_RATIO 0.7
```

7.29.2 Function Documentation

7.29.2.1 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.

7.29.2.2 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 vactor of cv::Point to work on.
in	<i>width</i>	The width of the image.
in	<i>height</i>	The height of the image.

7.29.2.3 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.

7.29.2.4 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 the 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.

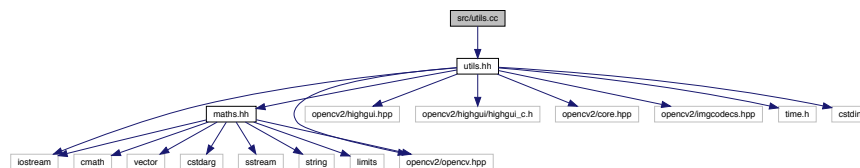
Returns

A 0 is return if the function reach the end.

7.30 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.
- void [mywaitkey](#) (Tuple< string > windowNames)
Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed. When a key is pressed some windows are closed.
- int64_t [timeutils::timespecDiff](#) (struct timespec *timeA_p, struct timespec *timeB_p)
- double [timeutils::getTimeS](#) ()

7.30.1 Function Documentation

7.30.1.1 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.

7.30.1.2 mywaitkey() [1/3]

```
void mywaitkey ( )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed.

7.30.1.3 mywaitkey() [2/3]

```
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.
-------------------	---

7.30.1.4 mywaitkey() [3/3]

```
void mywaitkey (
    Tuple< string > windowNames )
```

Function to use after [my_imshow\(\)](#) for keeping the image opened until a key is pressed. When a key is pressed some windows are closed.

Parameters

<i>windowNames</i>	The names of the windows to close after pressing a key.
--------------------	---

Index

- ~CameraCapture
 - CameraCapture, [57](#)
- ~ClipperBase
 - ClipperLib::ClipperBase, [64](#)
- ~ClipperOffset
 - ClipperLib::ClipperOffset, [70](#)
- ~PolyNode
 - ClipperLib::PolyNode, [138](#)
- ~PolyTree
 - ClipperLib::PolyTree, [140](#)
- ~Settings
 - Settings, [145](#)
- ~clipperException
 - ClipperLib::clipperException, [69](#)
- A_180
 - maths.hh, [207](#)
- A_2PI
 - maths.hh, [207](#)
- A_360
 - maths.hh, [207](#)
- A_PI
 - maths.hh, [207](#)
- Abs
 - ClipperLib, [15](#)
- add
 - Angle, [36](#)
 - Tuple< T >, [162](#)
- AddBoundsToLML
 - ClipperLib::ClipperBase, [64](#)
- addObject
 - Mapp, [114](#)
- AddPath
 - ClipperLib::ClipperBase, [64](#)
 - ClipperLib::ClipperOffset, [70](#)
- AddPaths
 - ClipperLib::ClipperBase, [64](#)
 - ClipperLib::ClipperOffset, [70](#)
- AddPolyNodeToPaths
 - ClipperLib, [15](#)
- addUnMap
 - Settings, [145](#)
- Angle, [33](#)
 - add, [36](#)
 - Angle, [35](#)
 - ANGLE_TYPE, [35](#)
 - checkValue, [36](#)
 - copy, [36](#)
 - cos, [37](#)
 - DEG, [35](#)
 - degToRad, [37](#)
 - div, [37](#)
 - equal, [37](#)
 - get, [38](#)
 - getType, [38](#)
 - getTypeName, [38](#)
 - INVALID, [35](#)
 - mul, [38](#)
 - normalize, [39](#)
 - operator *, [39](#)
 - operator ==, [39](#)
 - operator double, [40](#)
 - operator float, [40](#)
 - operator int, [40](#)
 - operator long, [40](#)
 - operator !=, [41](#)
 - operator < <, [46](#)
 - operator +, [41](#)
 - operator +=, [41](#)
 - operator -, [42](#)
 - operator -=, [42](#)
 - operator /, [43](#)
 - operator /=, [43](#)
 - operator =, [43](#), [44](#)
 - operator ==, [44](#)
 - RAD, [35](#)
 - radToDeg, [44](#)
 - set, [44](#)
 - setType, [45](#)
 - sin, [45](#)
 - sub, [45](#)
 - tan, [46](#)
 - to_string, [46](#)
 - toDeg, [46](#)
 - toRad, [46](#)
- angle
 - Configuration2< T1 >, [74](#)
- ANGLE_TYPE
 - Angle, [35](#)
- ArcTolerance
 - ClipperLib::ClipperOffset, [71](#)
- Area
 - ClipperLib, [16](#)
- AREA_MIN
 - unwrapping.cc, [220](#)
- AREA_RATIO
 - unwrapping.cc, [220](#)
- aspectRatio
 - CalSettings, [52](#)

- atImageList
 - CalSettings, 52
- begin
 - Curve< T >, 84, 85
 - Tuple< T >, 162
- BLACK
 - Settings, 144
- blackMask
 - Settings, 154
- BLUE
 - Settings, 144
- blueMask
 - Settings, 154
- boardSize
 - CalSettings, 52
- Bot
 - ClipperLib::TEdge, 157
- bottom
 - ClipperLib::IntRect, 110
- BottomPt
 - ClipperLib::OutRec, 126
- calcBoardCornerPositions
 - calibration.cc, 172
- calibFixPrincipalPoint
 - CalSettings, 52
- CALIBRATED
 - calibration.hh, 189
- calibration
 - calibration.cc, 172
 - calibration.hh, 189
- calibration.cc
 - calcBoardCornerPositions, 172
 - calibration, 172
 - computeReprojectionErrors, 172
 - read, 173
 - runCalibration, 173
 - runCalibrationAndSave, 174
 - saveCameraParams, 174
- calibration.hh
 - CALIBRATED, 189
 - calibration, 189
 - CAPTURING, 189
 - DETECTION, 189
 - runCalibrationAndSave, 189
- calibration_run.cc
 - main, 216
- calibrationFile
 - Settings, 154
- calibrationPattern
 - CalSettings, 52
- calibZeroTangentDist
 - CalSettings, 52
- CalSettings, 47
 - aspectRatio, 52
 - atImageList, 52
 - boardSize, 52
 - calibFixPrincipalPoint, 52
 - calibrationPattern, 52
 - calibZeroTangentDist, 52
 - CalSettings, 49
 - cameraID, 52
 - CHESSBOARD, 49
 - delay, 53
 - fixK1, 53
 - fixK2, 53
 - fixK3, 53
 - fixK4, 53
 - fixK5, 53
 - flag, 54
 - flipVertical, 54
 - goodInput, 54
 - IMAGE_LIST, 49
 - imageList, 54
 - input, 54
 - inputCapture, 54
 - InputType, 49
 - inputType, 54
 - INVALID, 49
 - isListOfImages, 49
 - nextImage, 50
 - NOT_EXISTING, 49
 - nrFrames, 55
 - outputFileName, 55
 - Pattern, 49
 - read, 50
 - readStringList, 50
 - showUndistorted, 55
 - squareSize, 55
 - useFisheye, 55
 - validate, 51
 - write, 51
 - writeExtrinsics, 55
 - writePoints, 56
- camera_capture.cc
 - DEBUG, 176
 - SDEBUG, 176
- CameraCapture, 56
 - ~CameraCapture, 57
 - CameraCapture, 56
 - grab, 57
 - isAlive, 57
 - isOpened, 57
 - loadCoefficients, 58
 - startCamera, 58
- CameraCapture::input_options_t, 101
 - cameraFPS, 102
 - frameHeight_px, 103
 - frameWidth_px, 103
 - input_options_t, 102
 - nameCamera, 103
- cameraFPS
 - CameraCapture::input_options_t, 102
- cameraID
 - CalSettings, 52
- CAPTURING

- calibration.hh, 189
- cellsFromSegment
 - Mapp, 115
- center
 - Object, 121
- changeBuffer
 - DW, 31
- changeMask
 - Settings, 145, 146
- check
 - Dubins< T >, 90
- checkSegment
 - Mapp, 115
- checkSegmentCollisionWithType
 - Mapp, 115
- checkValue
 - Angle, 36
- CHESSBOARD
 - CalSettings, 49
- ChildCount
 - ClipperLib::PolyNode, 138
- Childs
 - ClipperLib::PolyNode, 139
- cInt
 - ClipperLib, 12
- circline
 - dubins.hh, 201
- clean
 - Settings, 146
- cleanAndRead
 - Settings, 146
- CleanPolygon
 - ClipperLib, 16
- CleanPolygons
 - ClipperLib, 16, 17
- Clear
 - ClipperLib::ClipperBase, 64
 - ClipperLib::ClipperOffset, 70
 - ClipperLib::PolyTree, 140
- Clipper
 - ClipperLib::Clipper, 59
 - ClipperLib::PolyNode, 138
 - ClipperLib::PolyTree, 141
- clipper.cc
 - HORIZONTAL, 179
 - NEAR_ZERO, 179
 - TOLERANCE, 179
- clipper.hh
 - CLIPPER_VERSION, 193
 - use_lines, 193
- CLIPPER_VERSION
 - clipper.hh, 193
- ClipperBase
 - ClipperLib::ClipperBase, 64
- clipperException
 - ClipperLib::clipperException, 69
- ClipperLib, 9
 - Abs, 15
 - AddPolyNodeToPaths, 15
 - Area, 16
 - cInt, 12
 - CleanPolygon, 16
 - CleanPolygons, 16, 17
 - ClipType, 13
 - ClosedPathsFromPolyTree, 17
 - ctDifference, 13
 - ctIntersection, 13
 - ctUnion, 13
 - ctXor, 13
 - def_arc_tolerance, 30
 - Direction, 13
 - DisposeOutPts, 17
 - DistanceFromLineSqrd, 17
 - DistanceSqrd, 17
 - dLeftToRight, 13
 - dRightToLeft, 13
 - DupOutPt, 17
 - E2InsertsBeforeE1, 18
 - EdgeList, 12
 - EdgesAdjacent, 18
 - EdgeSide, 13
 - EndType, 14
 - esLeft, 14
 - esRight, 14
 - etClosedLine, 14
 - etClosedPolygon, 14
 - etOpenButt, 14
 - etOpenRound, 14
 - etOpenSquare, 14
 - ExcludeOp, 18
 - FindNextLocMin, 18
 - FirstIsBottomPt, 18
 - GetBottomPt, 18
 - GetDx, 19
 - GetHorzDirection, 19
 - GetLowermostRec, 19
 - GetMaximaPair, 19
 - GetMaximaPairEx, 19
 - GetNextInAEL, 19
 - GetOverlap, 20
 - GetOverlapSegment, 20
 - GetUnitNormal, 20
 - hiRange, 30
 - HorzSegmentsOverlap, 20
 - InitEdge, 20
 - InitEdge2, 21
 - InitOptions, 14
 - Int128Mul, 21
 - IntersectList, 12
 - IntersectListSort, 21
 - IntersectPoint, 21
 - ioPreserveCollinear, 14
 - ioReverseSolution, 14
 - ioStrictlySimple, 14
 - IsHorizontal, 21
 - IsIntermediate, 21

- IsMaxima, [22](#)
- IsMinima, [22](#)
- JoinHorz, [22](#)
- JoinList, [12](#)
- JoinType, [14](#)
- jtMiter, [14](#)
- jtRound, [14](#)
- jtSquare, [14](#)
- long64, [12](#)
- loRange, [30](#)
- Minkowski, [22](#)
- MinkowskiDiff, [22](#)
- MinkowskiSum, [23](#)
- NodeType, [15](#)
- ntAny, [15](#)
- ntClosed, [15](#)
- ntOpen, [15](#)
- OpenPathsFromPolyTree, [23](#)
- operator<<, [23](#), [24](#)
- Orientation, [24](#)
- OutRec1RightOfOutRec2, [24](#)
- ParseFirstLeft, [24](#)
- Path, [12](#)
- Paths, [12](#)
- pftEvenOdd, [15](#)
- pftNegative, [15](#)
- pftNonZero, [15](#)
- pftPositive, [15](#)
- pi, [30](#)
- PointCount, [24](#)
- PointInPolygon, [25](#)
- PointIsVertex, [25](#)
- PointsAreClose, [25](#)
- Poly2ContainsPoly1, [25](#)
- PolyFillType, [15](#)
- PolyNodes, [12](#)
- PolyOutList, [13](#)
- PolyTreeToPaths, [25](#)
- PolyType, [15](#)
- Pt2IsBetweenPt1AndPt3, [26](#)
- ptClip, [15](#)
- ptSubject, [15](#)
- RangeTest, [26](#)
- RemoveEdge, [26](#)
- ReverseHorizontal, [26](#)
- ReversePath, [26](#)
- ReversePaths, [26](#)
- ReversePolyPtLinks, [27](#)
- Round, [27](#)
- SetDx, [27](#)
- SimplifyPolygon, [27](#)
- SimplifyPolygons, [27](#)
- Skip, [30](#)
- SlopesEqual, [28](#)
- SlopesNearCollinear, [28](#)
- SwapIntersectNodes, [28](#)
- SwapPoints, [29](#)
- SwapPolyIndexes, [29](#)
- SwapSides, [29](#)
- TopX, [29](#)
- TranslatePath, [29](#)
- two_pi, [30](#)
- ulong64, [13](#)
- Unassigned, [30](#)
- UpdateOutPtIdxs, [29](#)
- ClipperLib::Clipper, [58](#)
 - Clipper, [59](#)
 - Execute, [60](#)
 - ExecuteInternal, [60](#)
 - ReverseSolution, [61](#)
 - StrictlySimple, [61](#)
- ClipperLib::ClipperBase, [62](#)
 - ~ClipperBase, [64](#)
 - AddBoundsToLML, [64](#)
 - AddPath, [64](#)
 - AddPaths, [64](#)
 - Clear, [64](#)
 - ClipperBase, [64](#)
 - CreateOutRec, [65](#)
 - DeleteFromAEL, [65](#)
 - DisposeAllOutRecs, [65](#)
 - DisposeLocalMinimaList, [65](#)
 - DisposeOutRec, [65](#)
 - GetBounds, [65](#)
 - InsertScanbeam, [65](#)
 - LocalMinimaPending, [66](#)
 - m_ActiveEdges, [67](#)
 - m_CurrentLM, [67](#)
 - m_edges, [67](#)
 - m_HasOpenPaths, [67](#)
 - m_MinimaList, [67](#)
 - m_PolyOuts, [68](#)
 - m_PreserveCollinear, [68](#)
 - m_Scanbeam, [68](#)
 - m_UseFullRange, [68](#)
 - MinimaList, [63](#)
 - PopLocalMinima, [66](#)
 - PopScanbeam, [66](#)
 - PreserveCollinear, [66](#)
 - ProcessBound, [66](#)
 - Reset, [66](#)
 - ScanbeamList, [63](#)
 - SwapPositionsInAEL, [67](#)
 - UpdateEdgeIntoAEL, [67](#)
- ClipperLib::clipperException, [68](#)
 - ~clipperException, [69](#)
 - clipperException, [69](#)
 - what, [69](#)
- ClipperLib::ClipperOffset, [69](#)
 - ~ClipperOffset, [70](#)
 - AddPath, [70](#)
 - AddPaths, [70](#)
 - ArcTolerance, [71](#)
 - Clear, [70](#)
 - ClipperOffset, [70](#)
 - Execute, [70](#), [71](#)

- MiterLimit, 71
- ClipperLib::DoublePoint, 86
 - DoublePoint, 86
 - X, 87
 - Y, 87
- ClipperLib::Int128, 103
 - hi, 106
 - Int128, 104
 - lo, 106
 - operator !=, 104
 - operator >, 105
 - operator >=, 105
 - operator -, 104
 - operator -=, 104
 - operator double, 105
 - operator <, 105
 - operator <=, 106
 - operator +, 105
 - operator +=, 105
 - operator -, 105
 - operator =, 106
 - operator ==, 106
- ClipperLib::IntersectNode, 107
 - Edge1, 107
 - Edge2, 107
 - Pt, 107
- ClipperLib::IntPoint, 108
 - IntPoint, 108
 - operator !=, 108
 - operator ==, 109
 - X, 109
 - Y, 109
- ClipperLib::IntRect, 109
 - bottom, 110
 - left, 110
 - right, 110
 - top, 110
- ClipperLib::Join, 110
 - OffPt, 111
 - OutPt1, 111
 - OutPt2, 111
- ClipperLib::LocalMinimum, 112
 - LeftBound, 112
 - RightBound, 112
 - Y, 112
- ClipperLib::LocMinSorter, 113
 - operator(), 113
- ClipperLib::OutPt, 124
 - Idx, 124
 - Next, 124
 - Prev, 125
 - Pt, 125
- ClipperLib::OutRec, 125
 - BottomPt, 126
 - FirstLeft, 126
 - Idx, 126
 - IsHole, 126
 - IsOpen, 126
 - PolyNd, 126
 - Pts, 126
- ClipperLib::PolyNode, 136
 - ~PolyNode, 138
 - ChildCount, 138
 - Childs, 139
 - Clipper, 138
 - ClipperOffset, 138
 - Contour, 139
 - GetNext, 138
 - IsHole, 138
 - IsOpen, 138
 - Parent, 139
 - PolyNode, 137
- ClipperLib::PolyTree, 139
 - ~PolyTree, 140
 - Clear, 140
 - Clipper, 141
 - GetFirst, 140
 - Total, 141
- ClipperLib::TEdge, 157
 - Bot, 157
 - Curr, 158
 - Dx, 158
 - Next, 158
 - NextInAEL, 158
 - NextInLML, 158
 - NextInSEL, 158
 - OutIdx, 158
 - PolyTyp, 158
 - Prev, 159
 - PrevInAEL, 159
 - PrevInSEL, 159
 - Side, 159
 - Top, 159
 - WindCnt, 159
 - WindCnt2, 159
 - WindDelta, 159
- ClipperOffset
 - ClipperLib::ClipperOffset, 70
 - ClipperLib::PolyNode, 138
- ClipType
 - ClipperLib, 13
- ClosedPathsFromPolyTree
 - ClipperLib, 17
- COLOR
 - Settings, 144
- computeCenter
 - Object, 119
- computeRadius
 - Object, 119
- computeReprojectionErrors
 - calibration.cc, 172
- Configuration2
 - Configuration2< T1 >, 73, 74
- Configuration2< T1 >, 71
 - angle, 74
 - Configuration2, 73, 74

- copy, 75
- distance, 75
- equal, 75
- EuDistance, 76
- MaDistance, 76
- offset, 77
- offset_angle, 78
- offset_x, 78
- offset_y, 79
- operator Point2< T2 >, 79
- operator<<, 82
- operator=, 79
- operator==, 80
- point, 80
- to_string, 80
- x, 80, 81
- y, 81
- configure
 - configure.cc, 180
 - configure.hh, 194
- configure.cc
 - configure, 180
 - filter, 182
 - on_high_h_thresh_trackbar, 180
 - on_high_s_thresh_trackbar, 181
 - on_high_v_thresh_trackbar, 181
 - on_low_h_thresh_trackbar, 181
 - on_low_s_thresh_trackbar, 181
 - on_low_v_thresh_trackbar, 181
 - show_all_conditions, 181
 - update_trackers, 182
- configure.hh
 - configure, 194
 - show_all_conditions, 194
- Contour
 - ClipperLib::PolyNode, 139
- convexHullFile
 - Settings, 154
- copy
 - Angle, 36
 - Configuration2< T1 >, 75
 - Filter, 98
 - Point2< T >, 129
- cos
 - Angle, 37
- COUT
 - utils.hh, 213
- CreateOutRec
 - ClipperLib::ClipperBase, 65
- crop_number_section
 - detection.cc, 184
 - detection.hh, 196
- ctDifference
 - ClipperLib, 13
- ctIntersection
 - ClipperLib, 13
- ctUnion
 - ClipperLib, 13
- ctXor
 - ClipperLib, 13
- Curr
 - ClipperLib::TEdge, 158
- Curve
 - Curve< T >, 84
- Curve< T >, 82
 - begin, 84, 85
 - Curve, 84
 - end, 85
 - operator<<, 85
 - P0, 86
 - P1, 86
 - to_string, 85
- DEBUG
 - camera_capture.cc, 176
- def_arc_tolerance
 - ClipperLib, 30
- DEG
 - Angle, 35
- DEGTORAD
 - maths.hh, 205
- degToRad
 - Angle, 37
- delay
 - CalSettings, 53
- DeleteFromAEL
 - ClipperLib::ClipperBase, 65
- DETECTION
 - calibration.hh, 189
- detection
 - detection.cc, 184
 - detection.hh, 196
- detection.cc
 - crop_number_section, 184
 - detection, 184
 - EPS_CURVE, 183
 - erode_dilation, 184
 - find_contours, 185
 - load_number_template, 185
 - number_recognition, 185
 - s, 186
 - save_convex_hull, 185
 - shape_detection, 186
 - templates, 186
- detection.hh
 - crop_number_section, 196
 - detection, 196
 - erode_dilation, 196
 - find_contours, 197
 - load_number_template, 197
 - number_recognition, 197
 - save_convex_hull, 198
 - shape_detection, 198
- detection_run.cc
 - main, 217
- dimX
 - Mapp, 117

- dimY
 - Mapp, 117
- DInf
 - maths.hh, 205
- Direction
 - ClipperLib, 13
- DisposeAllOutRecs
 - ClipperLib::ClipperBase, 65
- DisposeLocalMinimalList
 - ClipperLib::ClipperBase, 65
- DisposeOutPts
 - ClipperLib, 17
- DisposeOutRec
 - ClipperLib::ClipperBase, 65
- distance
 - Configuration2< T1 >, 75
 - Point2< T >, 129
 - Tuple< T >, 162
 - unwrapping.cc, 220
- DISTANCE_TYPE
 - maths.hh, 206
- DistanceFromLineSqrd
 - ClipperLib, 17
- DistanceSqrd
 - ClipperLib, 17
- div
 - Angle, 37
- dLeftToRight
 - ClipperLib, 13
- DoublePoint
 - ClipperLib::DoublePoint, 86
- draw.hh
 - int, 199
- dRightToLeft
 - ClipperLib, 13
- Dubins
 - Dubins< T >, 89
- Dubins< T >, 87
 - check, 90
 - Dubins, 89
 - getA1, 90
 - getA2, 90
 - getA3, 90
 - getId, 90
 - getKMax, 90
 - length, 91
 - LRL, 91
 - LSL, 91
 - LSR, 91
 - operator<<, 93
 - rangeSymm, 91
 - RLR, 91
 - RSL, 92
 - RSR, 92
 - scaleFromStandard, 92
 - scaleToStandard, 92
 - shortest_path, 92
 - splitt, 93
 - to_string, 93
- dubins.hh
 - circline, 201
 - KMAX, 201
 - MORE_FUNCTIONS, 201
 - PIECE_LENGTH, 201
 - sinc, 201
- DubinsArc
 - DubinsArc< T1, T2 >, 95
- DubinsArc< T1, T2 >, 94
 - DubinsArc, 95
 - getK, 95
 - length, 95
 - operator<<, 96
 - splitt, 95
 - to_string, 96
- DupOutPt
 - ClipperLib, 17
- DW, 31
 - changeBuffer, 31
 - init, 31
 - map_buffer, 31
 - window, 31
- Dx
 - ClipperLib::TEdge, 158
- E2InsertsBeforeE1
 - ClipperLib, 18
- Edge1
 - ClipperLib::IntersectNode, 107
- Edge2
 - ClipperLib::IntersectNode, 107
- EdgeList
 - ClipperLib, 12
- EdgesAdjacent
 - ClipperLib, 18
- EdgeSide
 - ClipperLib, 13
- end
 - Curve< T >, 85
 - Tuple< T >, 163
- EndType
 - ClipperLib, 14
- EPS_CURVE
 - detection.cc, 183
- Epsi
 - maths.hh, 205
- equal
 - Angle, 37
 - Configuration2< T1 >, 75
 - maths.hh, 206
 - Point2< T >, 130
- erode_dilation
 - detection.cc, 184
 - detection.hh, 196
- esLeft
 - ClipperLib, 14
- esRight
 - ClipperLib, 14

- etClosedLine
 - ClipperLib, 14
- etClosedPolygon
 - ClipperLib, 14
- etOpenButt
 - ClipperLib, 14
- etOpenRound
 - ClipperLib, 14
- etOpenSquare
 - ClipperLib, 14
- EUCLIDEAN
 - maths.hh, 206
- EuDistance
 - Configuration2< T1 >, 76
 - Point2< T >, 130
 - Tuple< T >, 163
- ExcludeOp
 - ClipperLib, 18
- Execute
 - ClipperLib::Clipper, 60
 - ClipperLib::ClipperOffset, 70, 71
- ExecuteInternal
 - ClipperLib::Clipper, 60
- Filter, 96
 - copy, 98
 - Filter, 97, 98
 - High, 98
 - high_h, 100
 - high_s, 100
 - high_v, 100
 - Low, 99
 - low_h, 101
 - low_s, 101
 - low_v, 101
 - operator vector< int >, 99
 - operator<=, 100
 - operator=, 99
 - to_string, 99
- filter
 - configure.cc, 182
- find_contours
 - detection.cc, 185
 - detection.hh, 197
- find_rect
 - unwrapping.cc, 221
 - unwrapping.hh, 210
- FindNextLocMin
 - ClipperLib, 18
- FirstIsBottomPt
 - ClipperLib, 18
- FirstLeft
 - ClipperLib::OutRec, 126
- fixK1
 - CalSettings, 53
- fixK2
 - CalSettings, 53
- fixK3
 - CalSettings, 53
- fixK4
 - CalSettings, 53
- fixK5
 - CalSettings, 53
- flag
 - CalSettings, 54
- flipVertical
 - CalSettings, 54
- frameHeight_px
 - CameraCapture::input_options_t, 103
- frameWidth_px
 - CameraCapture::input_options_t, 103
- FREE
 - map.hh, 203
- GATE
 - map.hh, 203
- get
 - Angle, 38
 - Tuple< T >, 164
- getA1
 - Dubins< T >, 90
- getA2
 - Dubins< T >, 90
- getA3
 - Dubins< T >, 90
- GetBottomPt
 - ClipperLib, 18
- GetBounds
 - ClipperLib::ClipperBase, 65
- GetDx
 - ClipperLib, 19
- getFiles
 - settings.cc, 219
- GetFirst
 - ClipperLib::PolyTree, 140
- GetHorzDirection
 - ClipperLib, 19
- getId
 - Dubins< T >, 90
- getK
 - DubinsArc< T1, T2 >, 95
- getKMax
 - Dubins< T >, 90
- GetLowermostRec
 - ClipperLib, 19
- GetMaximaPair
 - ClipperLib, 19
- GetMaximaPairEx
 - ClipperLib, 19
- GetNext
 - ClipperLib::PolyNode, 138
- GetNextInAEL
 - ClipperLib, 19
- GetOverlap
 - ClipperLib, 20
- GetOverlapSegment
 - ClipperLib, 20
- getPointType

- Mapp, 116
- getTemplates
 - Settings, 146, 147
- getTimeS
 - timeutils, 32
- getType
 - Angle, 38
- getTypeName
 - Angle, 38
- GetUnitNormal
 - ClipperLib, 20
- getValue
 - Victim, 168
- goodInput
 - CalSettings, 54
- grab
 - CameraCapture, 57
- GREEN
 - Settings, 144
- greenMask
 - Settings, 154
- hi
 - ClipperLib::Int128, 106
- High
 - Filter, 98
- high_h
 - Filter, 100
- high_s
 - Filter, 100
- high_v
 - Filter, 100
- hiRange
 - ClipperLib, 30
- HORIZONTAL
 - clipper.cc, 179
- HorzSegmentsOverlap
 - ClipperLib, 20
- Idx
 - ClipperLib::OutPt, 124
 - ClipperLib::OutRec, 126
- IMAGE_LIST
 - CalSettings, 49
- imageList
 - CalSettings, 54
- INFO
 - utils.hh, 213
- init
 - DW, 31
- InitEdge
 - ClipperLib, 20
- InitEdge2
 - ClipperLib, 21
- InitOptions
 - ClipperLib, 14
- input
 - CalSettings, 54
- input_options_t
 - CameraCapture::input_options_t, 102
- inputCapture
 - CalSettings, 54
- InputType
 - CalSettings, 49
- inputType
 - CalSettings, 54
- InsertScanbeam
 - ClipperLib::ClipperBase, 65
- insidePoly
 - Object, 119
- insidePolyApprox
 - Object, 120
- int
 - draw.hh, 199
- Int128
 - ClipperLib::Int128, 104
- Int128Mul
 - ClipperLib, 21
- IntersectList
 - ClipperLib, 12
- IntersectListSort
 - ClipperLib, 21
- IntersectPoint
 - ClipperLib, 21
- IntPoint
 - ClipperLib::IntPoint, 108
- intrinsicCalibrationFile
 - Settings, 155
- INVALID
 - Angle, 35
 - CalSettings, 49
- ioPreserveCollinear
 - ClipperLib, 14
- ioReverseSolution
 - ClipperLib, 14
- ioStrictlySimple
 - ClipperLib, 14
- isAlive
 - CameraCapture, 57
- IsHole
 - ClipperLib::OutRec, 126
 - ClipperLib::PolyNode, 138
- IsHorizontal
 - ClipperLib, 21
- IsIntermediate
 - ClipperLib, 21
- isListOfImages
 - CalSettings, 49
- IsMaxima
 - ClipperLib, 22
- IsMinima
 - ClipperLib, 22
- IsOpen
 - ClipperLib::OutRec, 126
 - ClipperLib::PolyNode, 138
- isOpened
 - CameraCapture, 57

- JoinHorz
 - ClipperLib, [22](#)
- JoinList
 - ClipperLib, [12](#)
- JoinType
 - ClipperLib, [14](#)
- jtMiter
 - ClipperLib, [14](#)
- jtRound
 - ClipperLib, [14](#)
- jtSquare
 - ClipperLib, [14](#)
- kernelSide
 - Settings, [155](#)
- KMAX
 - dubins.hh, [201](#)
- left
 - ClipperLib::IntRect, [110](#)
- LeftBound
 - ClipperLib::LocalMinimum, [112](#)
- length
 - Dubins< T >, [91](#)
 - DubinsArc< T1, T2 >, [95](#)
- lengthX
 - Mapp, [117](#)
- lengthY
 - Mapp, [117](#)
- lo
 - ClipperLib::Int128, [106](#)
- load_number_template
 - detection.cc, [185](#)
 - detection.hh, [197](#)
- loadCoefficients
 - CameraCapture, [58](#)
 - unwrapping.cc, [221](#)
 - unwrapping.hh, [210](#)
- LocalMinimaPending
 - ClipperLib::ClipperBase, [66](#)
- long64
 - ClipperLib, [12](#)
- loRange
 - ClipperLib, [30](#)
- Low
 - Filter, [99](#)
- low_h
 - Filter, [101](#)
- low_s
 - Filter, [101](#)
- low_v
 - Filter, [101](#)
- LRL
 - Dubins< T >, [91](#)
- LSL
 - Dubins< T >, [91](#)
- LSR
 - Dubins< T >, [91](#)
- m_ActiveEdges
 - ClipperLib::ClipperBase, [67](#)
- m_CurrentLM
 - ClipperLib::ClipperBase, [67](#)
- m_edges
 - ClipperLib::ClipperBase, [67](#)
- m_HasOpenPaths
 - ClipperLib::ClipperBase, [67](#)
- m_MinimaList
 - ClipperLib::ClipperBase, [67](#)
- m_PolyOuts
 - ClipperLib::ClipperBase, [68](#)
- m_PreserveCollinear
 - ClipperLib::ClipperBase, [68](#)
- m_Scanbeam
 - ClipperLib::ClipperBase, [68](#)
- m_UseFullRange
 - ClipperLib::ClipperBase, [68](#)
- MaDistance
 - Configuration2< T1 >, [76](#)
 - Point2< T >, [131](#)
 - Tuple< T >, [164](#)
- main
 - calibration_run.cc, [216](#)
 - detection_run.cc, [217](#)
 - main.cc, [217](#)
 - unwrapping_run.cc, [218](#)
- main.cc
 - main, [217](#)
- MANHATTAN
 - maths.hh, [206](#)
- map
 - Mapp, [117](#)
- map.hh
 - FREE, [203](#)
 - GATE, [203](#)
 - OBJ_TYPE, [203](#)
 - OBST, [203](#)
 - VICT, [203](#)
- map_buffer
 - DW, [31](#)
- Mapp, [113](#)
 - addObject, [114](#)
 - cellsFromSegment, [115](#)
 - checkSegment, [115](#)
 - checkSegmentCollisionWithType, [115](#)
 - dimX, [117](#)
 - dimY, [117](#)
 - getPointType, [116](#)
 - lengthX, [117](#)
 - lengthY, [117](#)
 - map, [117](#)
 - Mapp, [114](#)
 - matrixToString, [116](#)
 - pixX, [117](#)
 - pixY, [118](#)
 - printDimensions, [116](#)
 - printMap, [117](#)

- maps
 - Settings, [147](#), [149](#), [150](#)
- mapsFolder
 - Settings, [155](#)
- mapsNames
 - Settings, [155](#)
- mapsUnNames
 - Settings, [155](#)
- maths.hh
 - A_180, [207](#)
 - A_2PI, [207](#)
 - A_360, [207](#)
 - A_PI, [207](#)
 - DEGTORAD, [205](#)
 - DInf, [205](#)
 - DISTANCE_TYPE, [206](#)
 - Epsi, [205](#)
 - equal, [206](#)
 - EUCLIDEAN, [206](#)
 - MANHATTAN, [206](#)
 - pow2, [207](#)
 - RADTODEG, [205](#)
 - tupleConstIter, [205](#)
 - tupleIter, [206](#)
- matrixToString
 - Mapp, [116](#)
- MinimalList
 - ClipperLib::ClipperBase, [63](#)
- Minkowski
 - ClipperLib, [22](#)
- MinkowskiDiff
 - ClipperLib, [22](#)
- MinkowskiSum
 - ClipperLib, [23](#)
- MiterLimit
 - ClipperLib::ClipperOffset, [71](#)
- MORE_FUNCTIONS
 - dubins.hh, [201](#)
- mul
 - Angle, [38](#)
- my_imshow
 - utils.cc, [223](#)
 - utils.hh, [213](#)
- mywaitkey
 - utils.cc, [223](#)
 - utils.hh, [214](#)
- NAME
 - utils.hh, [213](#)
- nameCamera
 - CameraCapture::input_options_t, [103](#)
- NEAR_ZERO
 - clipper.cc, [179](#)
- Next
 - ClipperLib::OutPt, [124](#)
 - ClipperLib::TEdge, [158](#)
- nextImage
 - CalSettings, [50](#)
- NextInAEL
 - ClipperLib::TEdge, [158](#)
- NextInLML
 - ClipperLib::TEdge, [158](#)
- NextInSEL
 - ClipperLib::TEdge, [158](#)
- NodeType
 - ClipperLib, [15](#)
- normalize
 - Angle, [39](#)
- NOT_EXISTING
 - CalSettings, [49](#)
- nPoints
 - Object, [120](#)
- NPOS
 - settings.cc, [218](#)
- nrFrames
 - CalSettings, [55](#)
- ntAny
 - ClipperLib, [15](#)
- ntClosed
 - ClipperLib, [15](#)
- ntOpen
 - ClipperLib, [15](#)
- number_recognition
 - detection.cc, [185](#)
 - detection.hh, [197](#)
- OBJ_TYPE
 - map.hh, [203](#)
- Object, [118](#)
 - center, [121](#)
 - computeCenter, [119](#)
 - computeRadius, [119](#)
 - insidePoly, [119](#)
 - insidePolyApprox, [120](#)
 - nPoints, [120](#)
 - offsetting, [120](#)
 - points, [121](#)
 - radius, [121](#)
 - size, [121](#)
 - toString, [121](#)
- OBST
 - map.hh, [203](#)
- Obstacle, [122](#)
 - Obstacle, [123](#)
 - print, [123](#)
 - toString, [123](#)
- OffPt
 - ClipperLib::Join, [111](#)
- offset
 - Configuration2< T1 >, [77](#)
 - Point2< T >, [131](#), [132](#)
- offset_angle
 - Configuration2< T1 >, [78](#)
- offset_x
 - Configuration2< T1 >, [78](#)
 - Point2< T >, [132](#)
- offset_y
 - Configuration2< T1 >, [79](#)

- Point2< T >, 133
- offsetting
 - Object, 120
- on_high_h_thresh_trackbar
 - configure.cc, 180
- on_high_s_thresh_trackbar
 - configure.cc, 181
- on_high_v_thresh_trackbar
 - configure.cc, 181
- on_low_h_thresh_trackbar
 - configure.cc, 181
- on_low_s_thresh_trackbar
 - configure.cc, 181
- on_low_v_thresh_trackbar
 - configure.cc, 181
- OpenPathsFromPolyTree
 - ClipperLib, 23
- operator !=
 - ClipperLib::Int128, 104
- operator >
 - ClipperLib::Int128, 105
- operator >=
 - ClipperLib::Int128, 105
- operator *
 - Angle, 39
- operator *=
 - Angle, 39
- operator -
 - ClipperLib::Int128, 104
- operator -=
 - ClipperLib::Int128, 104
- operator cv::Point
 - Point2< T >, 133
- operator double
 - Angle, 40
 - ClipperLib::Int128, 105
- operator float
 - Angle, 40
- operator int
 - Angle, 40
- operator long
 - Angle, 40
- operator Point2< T2 >
 - Configuration2< T1 >, 79
- operator vector< int >
 - Filter, 99
- operator vector< T1 >
 - Tuple< T >, 164
- operator!=
 - Angle, 41
 - ClipperLib::IntPoint, 108
 - Point2< T >, 133
- operator<
 - ClipperLib::Int128, 105
 - Point2< T >, 134
- operator<<
 - Angle, 46
 - ClipperLib, 23, 24
- Configuration2< T1 >, 82
- Curve< T >, 85
- Dubins< T >, 93
- DubinsArc< T1, T2 >, 96
- Filter, 100
- Point2< T >, 136
- Settings, 153
- Tuple< T >, 166
- operator<=
 - ClipperLib::Int128, 106
- operator()
 - ClipperLib::LocMinSorter, 113
- operator+
 - Angle, 41
 - ClipperLib::Int128, 105
- operator+=
 - Angle, 41
 - ClipperLib::Int128, 105
- operator-
 - Angle, 42
 - ClipperLib::Int128, 105
- operator-=
 - Angle, 42
- operator/
 - Angle, 43
- operator/=
 - Angle, 43
- operator=
 - Angle, 43, 44
 - ClipperLib::Int128, 106
 - Configuration2< T1 >, 79
 - Filter, 99
 - Point2< T >, 134
- operator==
 - Angle, 44
 - ClipperLib::Int128, 106
 - ClipperLib::IntPoint, 109
 - Configuration2< T1 >, 80
 - Point2< T >, 134
- Orientation
 - ClipperLib, 24
- OutIdx
 - ClipperLib::TEdge, 158
- OutPt1
 - ClipperLib::Join, 111
- OutPt2
 - ClipperLib::Join, 111
- outputFileName
 - CalSettings, 55
- OutRec1RightOfOutRec2
 - ClipperLib, 24
- P0
 - Curve< T >, 86
- P1
 - Curve< T >, 86
- Parent
 - ClipperLib::PolyNode, 139
- ParseFirstLeft

- ClipperLib, 24
- Path
 - ClipperLib, 12
- Paths
 - ClipperLib, 12
- Pattern
 - CalSettings, 49
- pftEvenOdd
 - ClipperLib, 15
- pftNegative
 - ClipperLib, 15
- pftNonZero
 - ClipperLib, 15
- pftPositive
 - ClipperLib, 15
- pi
 - ClipperLib, 30
- PIECE_LENGTH
 - dubins.hh, 201
- pixX
 - Mapp, 117
- pixY
 - Mapp, 118
- point
 - Configuration2< T1 >, 80
- Point2
 - Point2< T >, 128, 129
- Point2< T >, 127
 - copy, 129
 - distance, 129
 - equal, 130
 - EuDistance, 130
 - MaDistance, 131
 - offset, 131, 132
 - offset_x, 132
 - offset_y, 133
 - operator cv::Point, 133
 - operator!=, 133
 - operator<, 134
 - operator<<, 136
 - operator=, 134
 - operator==, 134
 - Point2, 128, 129
 - to_string, 134
 - x, 135
 - y, 135
- PointCount
 - ClipperLib, 24
- PointInPolygon
 - ClipperLib, 25
- PointIsVertex
 - ClipperLib, 25
- points
 - Object, 121
- PointsAreClose
 - ClipperLib, 25
- Poly2ContainsPoly1
 - ClipperLib, 25
- PolyFillType
 - ClipperLib, 15
- PolyNd
 - ClipperLib::OutRec, 126
- PolyNode
 - ClipperLib::PolyNode, 137
- PolyNodes
 - ClipperLib, 12
- PolyOutList
 - ClipperLib, 13
- PolyTreeToPaths
 - ClipperLib, 25
- PolyType
 - ClipperLib::TEdge, 158
- PolyType
 - ClipperLib, 15
- PopLocalMinima
 - ClipperLib::ClipperBase, 66
- PopScanbeam
 - ClipperLib::ClipperBase, 66
- pow2
 - maths.hh, 207
- PreserveCollinear
 - ClipperLib::ClipperBase, 66
- Prev
 - ClipperLib::OutPt, 125
 - ClipperLib::TEdge, 159
- PrevInAEL
 - ClipperLib::TEdge, 159
- PrevInSEL
 - ClipperLib::TEdge, 159
- print
 - Obstacle, 123
 - Victim, 168
- printDimensions
 - Mapp, 116
- printMap
 - Mapp, 117
- ProcessBound
 - ClipperLib::ClipperBase, 66
- Pt
 - ClipperLib::IntersectNode, 107
 - ClipperLib::OutPt, 125
- Pt2IsBetweenPt1AndPt3
 - ClipperLib, 26
- ptClip
 - ClipperLib, 15
- Pts
 - ClipperLib::OutRec, 126
- ptSubject
 - ClipperLib, 15
- RAD
 - Angle, 35
- radius
 - Object, 121
- RADTODEG
 - maths.hh, 205
- radToDeg

- Angle, 44
- rangeSymm
 - Dubins< T >, 91
- RangeTest
 - ClipperLib, 26
- read
 - calibration.cc, 173
 - CalSettings, 50
- readFromFile
 - Settings, 150
- readStringList
 - CalSettings, 50
- RED
 - Settings, 144
- redMask
 - Settings, 155
- remove
 - Tuple< T >, 165
- RemoveEdge
 - ClipperLib, 26
- Reset
 - ClipperLib::ClipperBase, 66
- ReverseHorizontal
 - ClipperLib, 26
- ReversePath
 - ClipperLib, 26
- ReversePaths
 - ClipperLib, 26
- ReversePolyPtLinks
 - ClipperLib, 27
- ReverseSolution
 - ClipperLib::Clipper, 61
- right
 - ClipperLib::IntRect, 110
- RightBound
 - ClipperLib::LocalMinimum, 112
- RLR
 - Dubins< T >, 91
- ROBOT
 - Settings, 144
- robotMask
 - Settings, 156
- Round
 - ClipperLib, 27
- RSL
 - Dubins< T >, 92
- RSR
 - Dubins< T >, 92
- runCalibration
 - calibration.cc, 173
- runCalibrationAndSave
 - calibration.cc, 174
 - calibration.hh, 189
- s
 - detection.cc, 186
- save
 - Settings, 150
- save_convex_hull
 - detection.cc, 185
 - detection.hh, 198
- saveCameraParams
 - calibration.cc, 174
- scaleFromStandard
 - Dubins< T >, 92
- scaleToStandard
 - Dubins< T >, 92
- ScanbeamList
 - ClipperLib::ClipperBase, 63
- SDEBUG
 - camera_capture.cc, 176
- set
 - Angle, 44
 - Tuple< T >, 165
- SetDx
 - ClipperLib, 27
- Settings, 141
 - ~Settings, 145
 - addUnMap, 145
 - BLACK, 144
 - blackMask, 154
 - BLUE, 144
 - blueMask, 154
 - calibrationFile, 154
 - changeMask, 145, 146
 - clean, 146
 - cleanAndRead, 146
 - COLOR, 144
 - convexHullFile, 154
 - getTemplates, 146, 147
 - GREEN, 144
 - greenMask, 154
 - intrinsicCalibrationFile, 155
 - kernelSide, 155
 - maps, 147, 149, 150
 - mapsFolder, 155
 - mapsNames, 155
 - mapsUnNames, 155
 - operator<<, 153
 - readFromFile, 150
 - RED, 144
 - redMask, 155
 - ROBOT, 144
 - robotMask, 156
 - save, 150
 - Settings, 144
 - templates, 156
 - templatesFolder, 156
 - to_string, 151
 - unMaps, 151–153
 - victimMask, 156
 - VICTIMS, 144
 - WHITE, 144
 - whiteMask, 156
 - writeToFile, 153
- settings.cc
 - getFiles, 219

- NPOS, [218](#)
- vecToFile, [219](#)
- setType
 - Angle, [45](#)
- setValue
 - Victim, [168](#)
- shape_detection
 - detection.cc, [186](#)
 - detection.hh, [198](#)
- shortest_path
 - Dubins< T >, [92](#)
- show_all_conditions
 - configure.cc, [181](#)
 - configure.hh, [194](#)
- showUndistorsed
 - CalSettings, [55](#)
- Side
 - ClipperLib::TEdge, [159](#)
- SimplifyPolygon
 - ClipperLib, [27](#)
- SimplifyPolygons
 - ClipperLib, [27](#)
- sin
 - Angle, [45](#)
- sinc
 - dubins.hh, [201](#)
- size
 - Object, [121](#)
 - Tuple< T >, [166](#)
- Skip
 - ClipperLib, [30](#)
- SlopesEqual
 - ClipperLib, [28](#)
- SlopesNearCollinear
 - ClipperLib, [28](#)
- splitIt
 - Dubins< T >, [93](#)
 - DubinsArc< T1, T2 >, [95](#)
- squareSize
 - CalSettings, [55](#)
- src/calibration.cc, [171](#)
- src/camera_capture.cc, [175](#)
- src/clipper.cc, [176](#)
- src/configure.cc, [180](#)
- src/detection.cc, [182](#)
- src/dubins.cc, [187](#)
- src/include/calibration.hh, [187](#)
- src/include/camera_capture.hh, [190](#)
- src/include/clipper.hh, [191](#)
- src/include/configure.hh, [193](#)
- src/include/detection.hh, [195](#)
- src/include/draw.hh, [199](#)
- src/include/dubins.hh, [200](#)
- src/include/filter.hh, [202](#)
- src/include/map.hh, [202](#)
- src/include/maths.hh, [203](#)
- src/include/objects.hh, [208](#)
- src/include/settings.hh, [208](#)
- src/include/unwrapping.hh, [209](#)
- src/include/utils.hh, [211](#)
- src/map.cc, [214](#)
- src/maths.cc, [215](#)
- src/objects.cc, [215](#)
- src/run/calibration_run.cc, [216](#)
- src/run/detection_run.cc, [216](#)
- src/run/main.cc, [217](#)
- src/run/unwrapping_run.cc, [217](#)
- src/settings.cc, [218](#)
- src/unwrapping.cc, [219](#)
- src/utils.cc, [222](#)
- startCamera
 - CameraCapture, [58](#)
- StrictlySimple
 - ClipperLib::Clipper, [61](#)
- sub
 - Angle, [45](#)
- SwapIntersectNodes
 - ClipperLib, [28](#)
- SwapPoints
 - ClipperLib, [29](#)
- SwapPolyIndexes
 - ClipperLib, [29](#)
- SwapPositionsInAEL
 - ClipperLib::ClipperBase, [67](#)
- SwapSides
 - ClipperLib, [29](#)
- tan
 - Angle, [46](#)
- templates
 - detection.cc, [186](#)
 - Settings, [156](#)
- templatesFolder
 - Settings, [156](#)
- timespecDiff
 - timeutils, [32](#)
- timeutils, [32](#)
- getTimeS, [32](#)
- timespecDiff, [32](#)
- to_string
 - Angle, [46](#)
 - Configuration2< T1 >, [80](#)
 - Curve< T >, [85](#)
 - Dubins< T >, [93](#)
 - DubinsArc< T1, T2 >, [96](#)
 - Filter, [99](#)
 - Point2< T >, [134](#)
 - Settings, [151](#)
 - Tuple< T >, [166](#)
- toDeg
 - Angle, [46](#)
- TOLERANCE
 - clipper.cc, [179](#)
- Top
 - ClipperLib::TEdge, [159](#)
- top
 - ClipperLib::IntRect, [110](#)

- TopX
 - ClipperLib, 29
- toRad
 - Angle, 46
- toString
 - Object, 121
 - Obstacle, 123
 - Victim, 168
- Total
 - ClipperLib::PolyTree, 141
- TranslatePath
 - ClipperLib, 29
- Tuple
 - Tuple< T >, 161
- Tuple< T >, 160
 - add, 162
 - begin, 162
 - distance, 162
 - end, 163
 - EuDistance, 163
 - get, 164
 - MaDistance, 164
 - operator vector< T1 >, 164
 - operator<<, 166
 - remove, 165
 - set, 165
 - size, 166
 - to_string, 166
 - Tuple, 161
- tupleConstIter
 - maths.hh, 205
- tupleIter
 - maths.hh, 206
- two_pi
 - ClipperLib, 30
- ulong64
 - ClipperLib, 13
- Unassigned
 - ClipperLib, 30
- unMaps
 - Settings, 151–153
- unwrapping
 - unwrapping.cc, 221
 - unwrapping.hh, 211
- unwrapping.cc
 - AREA_MIN, 220
 - AREA_RATIO, 220
 - distance, 220
 - find_rect, 221
 - loadCoefficients, 221
 - unwrapping, 221
- unwrapping.hh
 - find_rect, 210
 - loadCoefficients, 210
 - unwrapping, 211
- unwrapping_run.cc
 - main, 218
- update_trackers
 - configure.cc, 182
- UpdateEdgeIntoAEL
 - ClipperLib::ClipperBase, 67
- UpdateOutPtIdxs
 - ClipperLib, 29
- use_lines
 - clipper.hh, 193
- useFisheye
 - CalSettings, 55
- utils.cc
 - my_imshow, 223
 - mywaitkey, 223
- utils.hh
 - COUT, 213
 - INFO, 213
 - my_imshow, 213
 - mywaitkey, 214
 - NAME, 213
- validate
 - CalSettings, 51
- value
 - Victim, 169
- vecToFile
 - settings.cc, 219
- VICT
 - map.hh, 203
- Victim, 167
 - getValue, 168
 - print, 168
 - setValue, 168
 - toString, 168
 - value, 169
 - Victim, 168
- victimMask
 - Settings, 156
- VICTIMS
 - Settings, 144
- what
 - ClipperLib::clipperException, 69
- WHITE
 - Settings, 144
- whiteMask
 - Settings, 156
- WindCnt
 - ClipperLib::TEdge, 159
- WindCnt2
 - ClipperLib::TEdge, 159
- WindDelta
 - ClipperLib::TEdge, 159
- window
 - DW, 31
- write
 - CalSettings, 51
- writeExtrinsics
 - CalSettings, 55
- writePoints
 - CalSettings, 56

writeToFile

Settings, [153](#)

X

ClipperLib::DoublePoint, [87](#)

ClipperLib::IntPoint, [109](#)

x

Configuration2< T1 >, [80](#), [81](#)

Point2< T >, [135](#)

Y

ClipperLib::DoublePoint, [87](#)

ClipperLib::IntPoint, [109](#)

ClipperLib::LocalMinimum, [112](#)

y

Configuration2< T1 >, [81](#)

Point2< T >, [135](#)