

Multiscale

1.0.0

Generated by Doxygen 1.8.3.1

Mon Sep 30 2013 18:51:46

Contents

1 Multiscale	1
1.1 Brief description	1
1.2 Contact	1
2 Namespace Index	3
2.1 Namespace List	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 File Index	9
5.1 File List	9
6 Namespace Documentation	11
6.1 multiscale Namespace Reference	11
6.1.1 Enumeration Type Documentation	12
6.1.1.1 NumberIteratorType	12
6.2 multiscale::analysis Namespace Reference	12
6.2.1 Enumeration Type Documentation	13
6.2.1.1 Shape2D	13
6.3 multiscale::video Namespace Reference	13
7 Class Documentation	15
7.1 multiscale::video::AnnularSector Class Reference	15
7.1.1 Detailed Description	16
7.1.2 Constructor & Destructor Documentation	16
7.1.2.1 AnnularSector	16
7.1.2.2 ~AnnularSector	16
7.1.3 Member Function Documentation	16
7.1.3.1 getConcentration	16
7.1.3.2 getEndingAngle	17

7.1.3.3	getEndingRadius	17
7.1.3.4	getStartingAngle	17
7.1.3.5	getStartingRadius	17
7.1.3.6	initialise	17
7.1.3.7	toString	17
7.1.4	Member Data Documentation	17
7.1.4.1	concentration	18
7.1.4.2	endingAngle	18
7.1.4.3	endingRadius	18
7.1.4.4	startingAngle	18
7.1.4.5	startingRadius	18
7.2	multiscale::video::CartesianToConcentrationsConverter Class Reference	18
7.2.1	Detailed Description	20
7.2.2	Constructor & Destructor Documentation	20
7.2.2.1	CartesianToConcentrationsConverter	20
7.2.2.2	~CartesianToConcentrationsConverter	20
7.2.3	Member Function Documentation	20
7.2.3.1	convert	20
7.2.3.2	outputResults	21
7.2.3.3	readConcentrations	21
7.2.3.4	readHeaderLine	22
7.2.3.5	readInputData	22
7.2.4	Member Data Documentation	23
7.2.4.1	concentrations	23
7.2.4.2	height	23
7.2.4.3	inputFilepath	23
7.2.4.4	outputFilepath	23
7.2.4.5	simulationTime	24
7.2.4.6	width	24
7.3	multiscale::CartesianToConcentrationsConverterException Class Reference	24
7.3.1	Detailed Description	25
7.3.2	Constructor & Destructor Documentation	25
7.3.2.1	CartesianToConcentrationsConverterException	25
7.3.2.2	CartesianToConcentrationsConverterException	25
7.4	multiscale::video::CartesianToPolarConverter Class Reference	26
7.4.1	Detailed Description	27
7.4.2	Constructor & Destructor Documentation	27
7.4.2.1	CartesianToPolarConverter	27
7.4.2.2	~CartesianToPolarConverter	27
7.4.3	Member Function Documentation	27

7.4.3.1	convert	27
7.4.3.2	outputResultsAsFile	28
7.4.3.3	outputResultsAsScript	29
7.4.3.4	readConcentrations	29
7.4.3.5	readHeaderLine	30
7.4.3.6	readInputData	30
7.4.3.7	transformToAnnularSectors	31
7.4.4	Member Data Documentation	31
7.4.4.1	annularSectors	31
7.4.4.2	concentrations	31
7.4.4.3	filepath	31
7.4.4.4	nrOfConcentricCircles	32
7.4.4.5	nrOfSectors	32
7.4.4.6	outputfilepath	32
7.4.4.7	simulationTime	32
7.5	multiscale::CartesianToPolarConverterException Class Reference	32
7.5.1	Detailed Description	34
7.5.2	Constructor & Destructor Documentation	34
7.5.2.1	CartesianToPolarConverterException	34
7.5.2.2	CartesianToPolarConverterException	34
7.6	multiscale::analysis::CircularityMeasure Class Reference	34
7.6.1	Detailed Description	35
7.6.2	Member Function Documentation	35
7.6.2.1	compute	35
7.6.2.2	compute	35
7.7	multiscale::analysis::CircularMatFactory Class Reference	35
7.7.1	Detailed Description	38
7.7.2	Constructor & Destructor Documentation	38
7.7.2.1	CircularMatFactory	38
7.7.2.2	~CircularMatFactory	38
7.7.3	Member Function Documentation	38
7.7.3.1	createCircularMask	38
7.7.3.2	createFromViewerImage	39
7.7.3.3	isValidViewerImage	40
7.7.3.4	maxColourBarIntensityFromViewerImage	40
7.7.3.5	processConcentrations	41
7.8	multiscale::CircularMatFactoryException Class Reference	41
7.8.1	Detailed Description	43
7.8.2	Constructor & Destructor Documentation	43
7.8.2.1	CircularMatFactoryException	43

7.8.2.2	<code>CircularMatFactoryException</code>	43
7.9	<code>multiscale::analysis::Cluster</code> Class Reference	43
7.9.1	Detailed Description	47
7.9.2	Constructor & Destructor Documentation	47
7.9.2.1	<code>Cluster</code>	47
7.9.2.2	<code>~Cluster</code>	47
7.9.3	Member Function Documentation	47
7.9.3.1	<code>addEntity</code>	47
7.9.3.2	<code>isValidOriginDependentValues</code>	48
7.9.3.3	<code>fieldNamesToString</code>	48
7.9.3.4	<code>fieldValuesToString</code>	48
7.9.3.5	<code>getEntities</code>	49
7.9.3.6	<code>getEntitiesCentrePoints</code>	49
7.9.3.7	<code>getEntitiesContourPoints</code>	49
7.9.3.8	<code>getEntitiesConvexHull</code>	50
7.9.3.9	<code>getMinAreaEnclosingCircleCentre</code>	50
7.9.3.10	<code>getMinAreaEnclosingCircleRadius</code>	51
7.9.3.11	<code>getMinAreaEnclosingRect</code>	52
7.9.3.12	<code>getMinAreaEnclosingTriangle</code>	52
7.9.3.13	<code>getPileUpDegree</code>	53
7.9.3.14	<code>initialise</code>	53
7.9.3.15	<code>isCircularMeasure</code>	54
7.9.3.16	<code>isRectangularMeasure</code>	54
7.9.3.17	<code>isTriangularMeasure</code>	54
7.9.3.18	<code>setOriginDependentMembers</code>	55
7.9.3.19	<code>updateArea</code>	55
7.9.3.20	<code>updateCentrePoint</code>	56
7.9.3.21	<code>updateClusterednessDegree</code>	56
7.9.3.22	<code>updatePerimeter</code>	56
7.9.3.23	<code>updatePileUpDegree</code>	56
7.9.3.24	<code>updateSpatialCollectionSpecificValues</code>	57
7.9.3.25	<code>validateOriginDependentValues</code>	57
7.9.4	Member Data Documentation	58
7.9.4.1	<code>entities</code>	58
7.9.4.2	<code>minAreaEnclosingCircleCentre</code>	58
7.9.4.3	<code>minAreaEnclosingCircleRadius</code>	58
7.9.4.4	<code>minAreaEnclosingRect</code>	58
7.9.4.5	<code>minAreaEnclosingTriangle</code>	58
7.9.4.6	<code>pileUpDegree</code>	59
7.10	<code>multiscale::analysis::ClusterDetector</code> Class Reference	59

7.10.1	Detailed Description	63
7.10.2	Constructor & Destructor Documentation	63
7.10.2.1	ClusterDetector	63
7.10.2.2	~ClusterDetector	63
7.10.3	Member Function Documentation	63
7.10.3.1	addEntitiesToClusters	63
7.10.3.2	analyseClusters	64
7.10.3.3	analyseClustersOriginDependentValues	64
7.10.3.4	clearPreviousDetectionResults	65
7.10.3.5	computeAveragePileUpDegree	65
7.10.3.6	computeClusterednessIndex	66
7.10.3.7	convertEntities	66
7.10.3.8	convertEpsValue	67
7.10.3.9	convertNonPiledUpEntities	67
7.10.3.10	convertPiledUpEntities	68
7.10.3.11	createDetectorSpecificTrackbars	68
7.10.3.12	detectAndAnalyseClusters	68
7.10.3.13	detectClusters	69
7.10.3.14	detectEntitiesInImage	70
7.10.3.15	getClusterConvexHull	71
7.10.3.16	getClusters	71
7.10.3.17	getEps	71
7.10.3.18	getMinPoints	72
7.10.3.19	getValidMinPointsValue	72
7.10.3.20	initialiseDetectorSpecificFields	73
7.10.3.21	outputResultsToCsvFile	73
7.10.3.22	processImageAndDetect	73
7.10.3.23	setEps	74
7.10.3.24	setMinPoints	75
7.10.3.25	updateClusterOriginDependentValues	75
7.10.4	Member Data Documentation	76
7.10.4.1	avgPileUpDegree	76
7.10.4.2	clusterednessIndex	76
7.10.4.3	clusters	76
7.10.4.4	entityPileupDegree	76
7.10.4.5	eps	77
7.10.4.6	minPoints	77
7.11	multiscale::ClusterException Class Reference	77
7.11.1	Detailed Description	78
7.11.2	Constructor & Destructor Documentation	78

7.11.2.1	ClusterException	78
7.11.2.2	ClusterException	78
7.12	multiscale::analysis::DataPoint Class Reference	78
7.12.1	Detailed Description	80
7.12.2	Constructor & Destructor Documentation	80
7.12.2.1	~DataPoint	80
7.12.3	Member Function Documentation	80
7.12.3.1	distanceTo	80
7.13	multiscale::analysis::DBSCAN Class Reference	81
7.13.1	Detailed Description	82
7.13.2	Member Function Documentation	82
7.13.2.1	addUnclassifiedNodesToSeedsList	82
7.13.2.2	allocateDistanceMatrix	82
7.13.2.3	assignBorderNodesToClusters	83
7.13.2.4	constructDistanceMatrix	83
7.13.2.5	expandCoreCluster	84
7.13.2.6	findClosestCoreDataPoint	85
7.13.2.7	labelUnclassifiedAndNoiseAsBorder	85
7.13.2.8	retrieveNeighbours	86
7.13.2.9	run	86
7.13.2.10	runAlgorithm	87
7.13.3	Member Data Documentation	88
7.13.3.1	distanceMatrix	88
7.13.3.2	eps	88
7.13.3.3	minPoints	88
7.13.3.4	nrOfDataPoints	89
7.14	multiscale::analysis::Detector Class Reference	89
7.14.1	Detailed Description	93
7.14.2	Constructor & Destructor Documentation	93
7.14.2.1	Detector	93
7.14.2.2	~Detector	93
7.14.3	Member Function Documentation	93
7.14.3.1	clearPreviousDetectionResults	93
7.14.3.2	createDetectorSpecificTrackbars	94
7.14.3.3	createTrackbars	94
7.14.3.4	createTrackbarsWindow	94
7.14.3.5	detect	94
7.14.3.6	detect	94
7.14.3.7	detectInDebugMode	94
7.14.3.8	detectInReleaseMode	95

7.14.3.9	displayImage	95
7.14.3.10	displayResultsInWindow	95
7.14.3.11	findGoodIntersectionPoints	95
7.14.3.12	findGoodPointsForAngle	96
7.14.3.13	initialise	96
7.14.3.14	initialiseDetectorSpecificFields	96
7.14.3.15	initialiseDetectorSpecificFieldsIfNotSet	96
7.14.3.16	initialiseDetectorSpecificImageDependentFields	96
7.14.3.17	initialiseImageDependentFields	97
7.14.3.18	initialiseImageOrigin	97
7.14.3.19	isValidInputImage	97
7.14.3.20	minAreaRectCentre	97
7.14.3.21	outputResults	97
7.14.3.22	outputResultsToCsvFile	98
7.14.3.23	outputResultsToCsvFile	98
7.14.3.24	outputResultsToFile	98
7.14.3.25	outputResultsToImage	98
7.14.3.26	polygonAngle	98
7.14.3.27	polygonAngle	99
7.14.3.28	printOutputErrorMessage	99
7.14.3.29	processImageAndDetect	100
7.14.3.30	processPressedKeyRequest	100
7.14.3.31	setDetectorSpecificFieldsInitialisationFlag	100
7.14.3.32	storeOutputImageOnDisk	101
7.14.4	Member Data Documentation	101
7.14.4.1	debugMode	101
7.14.4.2	detectMethodCalled	101
7.14.4.3	detectorSpecificFieldsInitialised	101
7.14.4.4	image	101
7.14.4.5	origin	101
7.14.4.6	outputFilepath	101
7.14.4.7	outputImage	102
7.15	multiscale::DetectorException Class Reference	102
7.15.1	Detailed Description	103
7.15.2	Constructor & Destructor Documentation	103
7.15.2.1	DetectorException	103
7.15.2.2	DetectorException	103
7.16	multiscale::analysis::Entity Class Reference	103
7.16.1	Detailed Description	106
7.16.2	Constructor & Destructor Documentation	106

7.16.2.1	Entity	106
7.16.2.2	Entity	107
7.16.2.3	\sim Entity	107
7.16.3	Member Function Documentation	107
7.16.3.1	isValid	107
7.16.3.2	distanceTo	108
7.16.3.3	distanceTo	108
7.16.3.4	getArea	108
7.16.3.5	getCentre	108
7.16.3.6	getContourPoints	109
7.16.3.7	getPerimeter	109
7.16.3.8	getPileUpDegree	109
7.16.3.9	toString	109
7.16.3.10	validateInputValues	109
7.16.4	Member Data Documentation	110
7.16.4.1	area	110
7.16.4.2	centre	110
7.16.4.3	contourPoints	110
7.16.4.4	perimeter	110
7.16.4.5	pileUpDegree	110
7.17	multiscale::EntityException Class Reference	111
7.17.1	Detailed Description	112
7.17.2	Constructor & Destructor Documentation	112
7.17.2.1	EntityException	112
7.17.2.2	EntityException	112
7.18	multiscale::Geometry2D Class Reference	112
7.18.1	Detailed Description	115
7.18.2	Member Function Documentation	115
7.18.2.1	angleBtwPoints	115
7.18.2.2	angleOfLineWrtOxAxis	116
7.18.2.3	areaOfTriangle	116
7.18.2.4	areCollinear	117
7.18.2.5	areEqualPoints	117
7.18.2.6	areIdenticalLines	118
7.18.2.7	areIdenticalLines	119
7.18.2.8	areOnTheSameSideOfLine	120
7.18.2.9	distanceBtwPoints	120
7.18.2.10	distanceFromPointToLine	121
7.18.2.11	findPointsOnEdge	121
7.18.2.12	inverseTranslate	122

7.18.2.13 <code>isAngleBetween</code>	122
7.18.2.14 <code>isAngleBetweenNonReflex</code>	123
7.18.2.15 <code>isBetweenCoordinates</code>	124
7.18.2.16 <code>isBetweenCoordinates</code>	124
7.18.2.17 <code>isOppositeAngleBetween</code>	124
7.18.2.18 <code>isOppositeAngleBetweenNonReflex</code>	124
7.18.2.19 <code>isPointOnEdge</code>	125
7.18.2.20 <code>isPointOnLineSegment</code>	126
7.18.2.21 <code>lineCircleIntersection</code>	126
7.18.2.22 <code>lineCircleOneIntersectionPoint</code>	127
7.18.2.23 <code>lineCircleTwoIntersectionPoints</code>	128
7.18.2.24 <code>lineEquationDeterminedByPoints</code>	129
7.18.2.25 <code>lineIntersection</code>	129
7.18.2.26 <code>lineIntersection</code>	130
7.18.2.27 <code>lineIntersection</code>	131
7.18.2.28 <code>lineSegmentCircleIntersection</code>	132
7.18.2.29 <code>lineSegmentIntersection</code>	132
7.18.2.30 <code>middlePoint</code>	133
7.18.2.31 <code>minimumDistancePointIndex</code>	134
7.18.2.32 <code>oppositeAngle</code>	134
7.18.2.33 <code>orthogonalLineToAnotherLineEdgePoints</code>	135
7.18.2.34 <code>slopeOfLine</code>	136
7.18.2.35 <code>translate</code>	136
7.19 <code>multiscale::LexicographicNumberIterator</code> Class Reference	137
7.19.1 Detailed Description	139
7.19.2 Constructor & Destructor Documentation	139
7.19.2.1 <code>LexicographicNumberIterator</code>	139
7.19.2.2 <code>~LexicographicNumberIterator</code>	140
7.19.3 Member Function Documentation	140
7.19.3.1 <code>digitsToNumber</code>	140
7.19.3.2 <code>hasNextInitialised</code>	140
7.19.3.3 <code>initialise</code>	141
7.19.3.4 <code>isLargerThanUpperBound</code>	141
7.19.3.5 <code>number</code>	142
7.19.3.6 <code>numberToDigits</code>	142
7.19.3.7 <code>padWithZeros</code>	143
7.19.3.8 <code>resetCurrentNumber</code>	144
7.19.3.9 <code>reverseDigits</code>	144
7.19.4 Member Data Documentation	144
7.19.4.1 <code>currentNumberDigits</code>	144

7.19.4.2	upperBoundDigits	144
7.20	multiscale::analysis::MatFactory Class Reference	145
7.20.1	Detailed Description	147
7.20.2	Constructor & Destructor Documentation	147
7.20.2.1	MatFactory	147
7.20.2.2	\sim MatFactory	147
7.20.3	Member Function Documentation	147
7.20.3.1	convertToIntensity	147
7.20.3.2	create	147
7.20.3.3	createFromViewerImage	148
7.20.3.4	initInputFile	148
7.20.3.5	isValidViewerImage	149
7.20.3.6	maxColourBarIntensityFromViewerImage	149
7.20.3.7	processConcentrations	149
7.20.4	Member Data Documentation	150
7.20.4.1	cols	150
7.20.4.2	rows	150
7.20.4.3	simulationTime	150
7.21	multiscale::MatFactoryException Class Reference	150
7.21.1	Detailed Description	152
7.21.2	Constructor & Destructor Documentation	152
7.21.2.1	MatFactoryException	152
7.21.2.2	\sim MatFactoryException	152
7.22	multiscale::MinEnclosingTriangleFinder Class Reference	152
7.22.1	Detailed Description	156
7.22.2	Constructor & Destructor Documentation	156
7.22.2.1	MinEnclosingTriangleFinder	156
7.22.2.2	\sim MinEnclosingTriangleFinder	156
7.22.3	Member Function Documentation	156
7.22.3.1	advance	156
7.22.3.2	advanceBToRightChain	157
7.22.3.3	areIdenticalLines	158
7.22.3.4	areIntersectingLines	158
7.22.3.5	find	159
7.22.3.6	findGammaIntersectionPoints	160
7.22.3.7	findMinEnclosingTriangle	161
7.22.3.8	findMinEnclosingTriangle	162
7.22.3.9	findVertexCOnSideB	162
7.22.3.10	gamma	163
7.22.3.11	height	164

7.22.3.12 height	165
7.22.3.13 initialise	166
7.22.3.14 initialiseAlgorithmVariables	167
7.22.3.15 initialiseConvexPolygon	167
7.22.3.16 intersects	168
7.22.3.17 intersectsAbove	168
7.22.3.18 intersectsAboveOrBelow	169
7.22.3.19 intersectsBelow	170
7.22.3.20 isFlushAngleBetweenPredecessorAndSuccessor	171
7.22.3.21 isGammaAngleBetween	171
7.22.3.22 isGammaAngleEqualTo	172
7.22.3.23 isLocalMinimalTriangle	173
7.22.3.24 isNotBTangency	173
7.22.3.25 isValidMinimalTriangle	174
7.22.3.26 lineEquationParameters	175
7.22.3.27 middlePointOfSideB	176
7.22.3.28 moveAIfLowAndBIfHigh	176
7.22.3.29 predecessor	177
7.22.3.30 returnMinEnclosingTriangle	178
7.22.3.31 searchForBTangency	179
7.22.3.32 successor	180
7.22.3.33 updateMinEnclosingTriangle	180
7.22.3.34 updateSideB	181
7.22.3.35 updateSidesBA	181
7.22.3.36 updateSidesCA	182
7.22.4 Member Data Documentation	183
7.22.4.1 a	183
7.22.4.2 area	183
7.22.4.3 b	183
7.22.4.4 c	183
7.22.4.5 nrOfPoints	183
7.22.4.6 polygon	184
7.22.4.7 sideAEndVertex	184
7.22.4.8 sideAStartVertex	184
7.22.4.9 sideBEndVertex	184
7.22.4.10 sideBStartVertex	184
7.22.4.11 sideCEndVertex	184
7.22.4.12 sideCStartVertex	184
7.22.4.13 validationFlag	185
7.22.4.14 vertexA	185

7.22.4.15	vertexB	185
7.22.4.16	vertexC	185
7.23	multiscale::MinEnclosingTriangleFinderException Class Reference	185
7.23.1	Detailed Description	187
7.23.2	Constructor & Destructor Documentation	187
7.23.2.1	MinEnclosingTriangleFinderException	187
7.23.2.2	~MinEnclosingTriangleFinderException	187
7.24	multiscale::NumberIterator Class Reference	187
7.24.1	Detailed Description	190
7.24.2	Constructor & Destructor Documentation	190
7.24.2.1	NumberIterator	190
7.24.2.2	~NumberIterator	190
7.24.3	Member Function Documentation	190
7.24.3.1	hasNext	190
7.24.3.2	hasNextInitialised	191
7.24.3.3	hasNextNotInitialised	191
7.24.3.4	init	191
7.24.3.5	initialise	192
7.24.3.6	number	192
7.24.3.7	reset	192
7.24.3.8	resetCurrentNumber	193
7.24.4	Member Data Documentation	193
7.24.4.1	isInitialised	193
7.24.4.2	upperBound	193
7.25	multiscale::Numeric Class Reference	194
7.25.1	Detailed Description	194
7.25.2	Member Function Documentation	195
7.25.2.1	almostEqual	195
7.25.2.2	greaterOrEqual	196
7.25.2.3	lessOrEqual	196
7.25.2.4	maximum	197
7.25.2.5	sign	197
7.25.3	Member Data Documentation	198
7.25.3.1	epsilon	198
7.26	multiscale::NumericRangeManipulator Class Reference	198
7.26.1	Detailed Description	198
7.26.2	Member Function Documentation	198
7.26.2.1	convertFromRange	198
7.27	multiscale::video::PolarCsvToInputFilesConverter Class Reference	199
7.27.1	Detailed Description	202

7.27.2	Constructor & Destructor Documentation	202
7.27.2.1	PolarCsvToInputFilesConverter	202
7.27.2.2	~PolarCsvToInputFilesConverter	202
7.27.3	Member Function Documentation	202
7.27.3.1	computeConcentration	202
7.27.3.2	computeConcentrationWrtArea	202
7.27.3.3	computeNextPositionConcentration	203
7.27.3.4	computeNonScaledConcentration	203
7.27.3.5	computeNormalisedConcentration	203
7.27.3.6	computeScaledConcentration	203
7.27.3.7	computeSimulationTime	204
7.27.3.8	convert	204
7.27.3.9	initInputFile	204
7.27.3.10	initIterators	204
7.27.3.11	initMaximumConcentration	205
7.27.3.12	initOutputFile	205
7.27.3.13	processInputFile	205
7.27.3.14	processLine	206
7.27.3.15	splitFirstPartInConcentrations	206
7.27.3.16	splitLineInConcentrations	206
7.27.3.17	splitOtherPartsInConcentrations	207
7.27.3.18	updateMaximumConcentration	207
7.27.3.19	validateInput	207
7.27.3.20	validateInputLine	207
7.27.3.21	validateSelectedConcentrationIndex	208
7.27.4	Member Data Documentation	208
7.27.4.1	circlesIterator	208
7.27.4.2	concentrationsIndex	208
7.27.4.3	filepath	208
7.27.4.4	maximumConcentration	208
7.27.4.5	nrOfConcentrationsForPosition	208
7.27.4.6	nrOfConcentricCircles	209
7.27.4.7	nrOfSectors	209
7.27.4.8	outputfilepath	209
7.27.4.9	sectorsIterator	209
7.27.4.10	selectedConcentrationIndex	209
7.27.4.11	useLogScaling	209
7.28	multiscale::PolarCsvToInputFilesConverterException Class Reference	209
7.28.1	Detailed Description	211
7.28.2	Constructor & Destructor Documentation	211

7.28.2.1	PolarCsvToInputFilesConverterException	211
7.28.2.2	PolarCsvToInputFilesConverterException	211
7.29	multiscale::video::PolarGnuplotScriptGenerator Class Reference	211
7.29.1	Detailed Description	212
7.29.2	Member Function Documentation	212
7.29.2.1	generateBody	212
7.29.2.2	generateFooter	212
7.29.2.3	generateHeader	213
7.29.2.4	generateScript	213
7.29.2.5	outputContent	214
7.29.2.6	outputFooter	214
7.29.2.7	outputHeader	214
7.29.2.8	readContentTemplate	215
7.30	multiscale::video::RectangularCsvToInputFilesConverter Class Reference	215
7.30.1	Detailed Description	218
7.30.2	Constructor & Destructor Documentation	218
7.30.2.1	RectangularCsvToInputFilesConverter	218
7.30.2.2	~RectangularCsvToInputFilesConverter	218
7.30.3	Member Function Documentation	218
7.30.3.1	computeConcentration	218
7.30.3.2	computeNextPositionConcentration	218
7.30.3.3	computeNonScaledConcentration	218
7.30.3.4	computeNormalisedConcentration	219
7.30.3.5	computeScaledConcentration	219
7.30.3.6	computeSimulationTime	219
7.30.3.7	convert	219
7.30.3.8	initInputFile	220
7.30.3.9	initIterators	220
7.30.3.10	initMaximumConcentration	220
7.30.3.11	initOutputFile	220
7.30.3.12	processInputFile	221
7.30.3.13	processLine	221
7.30.3.14	splitLineInConcentrations	221
7.30.3.15	splitLineInConcentrations	222
7.30.3.16	updateMaximumConcentration	222
7.30.3.17	validateInput	222
7.30.3.18	validateInputLine	222
7.30.3.19	validateSelectedConcentrationIndex	223
7.30.4	Member Data Documentation	223
7.30.4.1	columnsIterator	223

7.30.4.2	concentrationsIndex	223
7.30.4.3	height	223
7.30.4.4	filepath	223
7.30.4.5	maximumConcentration	224
7.30.4.6	nrOfConcentrationsForPosition	224
7.30.4.7	outputfilepath	224
7.30.4.8	rowsIterator	224
7.30.4.9	selectedConcentrationIndex	224
7.30.4.10	useLogScaling	224
7.30.4.11	width	224
7.31	multiscale::RectangularCsvToInputFilesConverterException Class Reference	224
7.31.1	Detailed Description	226
7.31.2	Constructor & Destructor Documentation	226
7.31.2.1	RectangularCsvToInputFilesConverterException	226
7.31.2.2	RectangularCsvToInputFilesConverterException	226
7.32	multiscale::video::RectangularEntityCsvToInputFilesConverter Class Reference	226
7.32.1	Detailed Description	228
7.32.2	Constructor & Destructor Documentation	228
7.32.2.1	RectangularEntityCsvToInputFilesConverter	228
7.32.2.2	~RectangularEntityCsvToInputFilesConverter	229
7.32.3	Member Function Documentation	229
7.32.3.1	computeCoordinate	229
7.32.3.2	computeSimulationTime	229
7.32.3.3	convert	229
7.32.3.4	initInputFile	230
7.32.3.5	initIterators	230
7.32.3.6	initOutputFile	230
7.32.3.7	processInputFile	230
7.32.3.8	processLine	231
7.32.3.9	splitLineInCoordinates	231
7.32.3.10	validateCoordinate	231
7.32.3.11	validateEntitiesGrid	232
7.32.3.12	validateInput	232
7.32.3.13	validateInputLine	232
7.32.3.14	validateMaxNrOfEntitiesPerPosition	233
7.32.3.15	validateSimulationTime	233
7.32.4	Member Data Documentation	233
7.32.4.1	entitiesIterator	233
7.32.4.2	height	233
7.32.4.3	filepath	233

7.32.4.4	maxNrOfEntitiesPerPosition	234
7.32.4.5	nrOfEntities	234
7.32.4.6	outputFilepath	234
7.32.4.7	width	234
7.33	multiscale::RectangularEntityCsvToInputFilesConverterException Class Reference	234
7.33.1	Detailed Description	236
7.33.2	Constructor & Destructor Documentation	236
7.33.2.1	RectangularEntityCsvToInputFilesConverterException	236
7.33.2.2	RectangularEntityCsvToInputFilesConverterException	236
7.34	multiscale::video::RectangularGnuplotScriptGenerator Class Reference	236
7.34.1	Detailed Description	237
7.34.2	Member Function Documentation	237
7.34.2.1	generateBody	237
7.34.2.2	generateFooter	237
7.34.2.3	generateHeader	238
7.34.2.4	generateScript	238
7.34.2.5	outputContent	239
7.34.2.6	outputFooter	239
7.34.2.7	outputHeader	239
7.35	multiscale::analysis::RectangularMatFactory Class Reference	240
7.35.1	Detailed Description	243
7.35.2	Constructor & Destructor Documentation	243
7.35.2.1	RectangularMatFactory	243
7.35.2.2	~RectangularMatFactory	243
7.35.3	Member Function Documentation	243
7.35.3.1	createFromViewerImage	243
7.35.3.2	isValidViewerImage	244
7.35.3.3	maxColourBarIntensityFromViewerImage	244
7.35.3.4	processConcentrations	245
7.36	multiscale::RectangularMatFactoryException Class Reference	246
7.36.1	Detailed Description	247
7.36.2	Constructor & Destructor Documentation	247
7.36.2.1	RectangularMatFactoryException	247
7.36.2.2	RectangularMatFactoryException	247
7.37	multiscale::analysis::Region Class Reference	247
7.37.1	Detailed Description	250
7.37.2	Constructor & Destructor Documentation	251
7.37.2.1	Region	251
7.37.2.2	~Region	251
7.37.3	Member Function Documentation	251

7.37.3.1	areValidInputValues	251
7.37.3.2	fieldNamesToString	252
7.37.3.3	fieldValuesToString	252
7.37.3.4	getDensity	253
7.37.3.5	getPolygon	253
7.37.3.6	isCircularMeasure	253
7.37.3.7	isRectangularMeasure	254
7.37.3.8	isTriangularMeasure	254
7.37.3.9	updateArea	255
7.37.3.10	updateCentrePoint	255
7.37.3.11	updateClusterednessDegree	255
7.37.3.12	updatePerimeter	255
7.37.3.13	updateSpatialCollectionSpecificValues	255
7.37.3.14	validateInputValues	255
7.37.4	Member Data Documentation	256
7.37.4.1	density	256
7.37.4.2	polygon	256
7.38	multiscale::analysis::RegionDetector Class Reference	256
7.38.1	Detailed Description	261
7.38.2	Constructor & Destructor Documentation	261
7.38.2.1	RegionDetector	261
7.38.2.2	\sim RegionDetector	261
7.38.3	Member Function Documentation	261
7.38.3.1	changeContrastAndBrightness	261
7.38.3.2	clearPreviousDetectionResults	262
7.38.3.3	computeAverageMeasures	262
7.38.3.4	convertAlpha	262
7.38.3.5	convertBeta	263
7.38.3.6	createDetectorSpecificTrackbars	263
7.38.3.7	createRegionFromPolygon	263
7.38.3.8	findContoursInImage	264
7.38.3.9	findRegions	265
7.38.3.10	getAlpha	265
7.38.3.11	getBeta	266
7.38.3.12	getBlurKernelSize	266
7.38.3.13	getEpsilon	266
7.38.3.14	getMorphologicalCloselterations	267
7.38.3.15	getOriginXCoordinate	267
7.38.3.16	getOriginYCoordinate	267
7.38.3.17	getRegionAreaThresh	267

7.38.3.18 getRegions	268
7.38.3.19 getThresholdValue	268
7.38.3.20 initialiseDetectorSpecificFields	268
7.38.3.21 initialiseDetectorSpecificImageDependentFields	268
7.38.3.22 isValidRegion	269
7.38.3.23 morphologicalClose	269
7.38.3.24 outputAveragedMeasuresToCsvFile	269
7.38.3.25 outputRegionsToCsvFile	270
7.38.3.26 outputResultsToCsvFile	270
7.38.3.27 outputResultsToImage	271
7.38.3.28 processImageAndDetect	271
7.38.3.29 regionArea	272
7.38.3.30 regionClusterednessDegree	272
7.38.3.31 regionDensity	273
7.38.3.32 regionHolesArea	273
7.38.3.33 setAlpha	274
7.38.3.34 setBeta	275
7.38.3.35 setBlurKernelSize	275
7.38.3.36 setEpsilon	276
7.38.3.37 setMorphologicalCloselterations	277
7.38.3.38 setOriginXCoordinate	277
7.38.3.39 setOriginYCoordinate	278
7.38.3.40 setRegionAreaThresh	278
7.38.3.41 setThresholdValue	279
7.38.3.42 smoothImage	280
7.38.3.43 thresholdImage	280
7.38.4 Member Data Documentation	281
7.38.4.1 alpha	281
7.38.4.2 avgClusterednessDegree	281
7.38.4.3 avgDensity	281
7.38.4.4 beta	281
7.38.4.5 blurKernelSize	281
7.38.4.6 epsilon	282
7.38.4.7 morphologicalCloselterations	282
7.38.4.8 regionAreaThresh	282
7.38.4.9 regions	282
7.38.4.10 thresholdValue	282
7.39 multiscale::RegionException Class Reference	282
7.39.1 Detailed Description	284
7.39.2 Constructor & Destructor Documentation	284

7.39.2.1	RegionException	284
7.39.2.2	RegionException	284
7.40	multiscale::RGBColourGenerator Class Reference	284
7.40.1	Detailed Description	285
7.40.2	Member Function Documentation	285
7.40.2.1	computeRGBValues	285
7.40.2.2	convertHSVToRGB	285
7.40.2.3	convertRGBToString	286
7.40.2.4	generate	286
7.40.2.5	generate	286
7.40.3	Member Data Documentation	286
7.40.3.1	blue	286
7.40.3.2	green	287
7.40.3.3	red	287
7.41	multiscale::analysis::Silhouette Class Reference	287
7.41.1	Detailed Description	288
7.41.2	Member Function Documentation	288
7.41.2.1	computeAverageDissimilarityBtwEntityAndCluster	288
7.41.2.2	computeAverageDissimilarityToOtherClusters	289
7.41.2.3	computeAverageDissimilarityWithinCluster	289
7.41.2.4	computeAverageMeasure	289
7.41.2.5	computeMeasure	290
7.41.2.6	computeOverallAverageMeasure	290
7.42	multiscale::analysis::SimulationClusterDetector Class Reference	290
7.42.1	Detailed Description	293
7.42.2	Constructor & Destructor Documentation	293
7.42.2.1	SimulationClusterDetector	293
7.42.2.2	~SimulationClusterDetector	294
7.42.3	Member Function Documentation	294
7.42.3.1	computePileUpDegreeAtPosition	294
7.42.3.2	detectEntitiesInImage	294
7.42.3.3	getEntityCentrePoint	295
7.42.3.4	getEntityContourPoints	296
7.42.3.5	initialiseDetectorSpecificImageDependentFields	296
7.42.3.6	initialiseThresholdedImage	297
7.42.3.7	isEntityAtPosition	297
7.42.3.8	outputClusterCircularShape	298
7.42.3.9	outputClusterRectangularShape	299
7.42.3.10	outputClusterShape	299
7.42.3.11	outputClusterTolImage	300

7.42.3.12	outputClusterTriangularShape	301
7.42.3.13	outputResultsToImage	301
7.42.4	Member Data Documentation	302
7.42.4.1	entityHeight	302
7.42.4.2	entityWidth	302
7.42.4.3	height	302
7.42.4.4	thresholdedImage	302
7.42.4.5	width	302
7.43	multiscale::SimulationClusterDetectorException Class Reference	303
7.43.1	Detailed Description	304
7.43.2	Constructor & Destructor Documentation	304
7.43.2.1	SimulationClusterDetectorException	304
7.43.2.2	SimulationClusterDetectorException	304
7.44	multiscale::analysis::SpatialCollection2D Class Reference	305
7.44.1	Detailed Description	308
7.44.2	Constructor & Destructor Documentation	308
7.44.2.1	SpatialCollection2D	308
7.44.2.2	~SpatialCollection2D	308
7.44.3	Member Function Documentation	309
7.44.3.1	convertPoints	309
7.44.3.2	fieldValuesToString	309
7.44.3.3	getAngle	309
7.44.3.4	getArea	310
7.44.3.5	getCentre	310
7.44.3.6	getClusterednessDegree	311
7.44.3.7	getDistanceFromOrigin	311
7.44.3.8	getPerimeter	312
7.44.3.9	getShape	312
7.44.3.10	initialise	313
7.44.3.11	isCircularMeasure	313
7.44.3.12	isRectangularMeasure	314
7.44.3.13	isTriangularMeasure	315
7.44.3.14	shapeAsString	315
7.44.3.15	toString	316
7.44.3.16	updateArea	316
7.44.3.17	updateCentrePoint	317
7.44.3.18	updateClusterednessDegree	317
7.44.3.19	updateMeasures	318
7.44.3.20	updateMeasuresIfRequired	319
7.44.3.21	updatePerimeter	321

7.44.3.22 updateShape	321
7.44.3.23 updateSpatialCollectionSpecificValues	322
7.44.4 Member Data Documentation	323
7.44.4.1 angle	323
7.44.4.2 area	323
7.44.4.3 centre	323
7.44.4.4 circularMeasure	324
7.44.4.5 clusterednessDegree	324
7.44.4.6 distanceFromOrigin	324
7.44.4.7 perimeter	324
7.44.4.8 rectangularMeasure	324
7.44.4.9 shape	324
7.44.4.10 triangularMeasure	324
7.44.4.11 updateFlag	325
7.45 multiscale::StandardNumberIterator Class Reference	325
7.45.1 Detailed Description	328
7.45.2 Constructor & Destructor Documentation	328
7.45.2.1 StandardNumberIterator	328
7.45.2.2 ~StandardNumberIterator	328
7.45.3 Member Function Documentation	328
7.45.3.1 hasNextInitialised	328
7.45.3.2 initialise	328
7.45.3.3 number	329
7.45.3.4 resetCurrentNumber	329
7.45.4 Member Data Documentation	329
7.45.4.1 currentNumber	329
7.46 multiscale::StringManipulator Class Reference	329
7.46.1 Detailed Description	330
7.46.2 Member Function Documentation	330
7.46.2.1 filenameFromPath	330
7.46.2.2 replace	331
7.46.2.3 split	332
7.46.2.4 toString	332
8 File Documentation	335
8.1 doc/mainpage.dox File Reference	335
8.2 include/multiscale/analysis/spatial/CircularityMeasure.hpp File Reference	335
8.3 include/multiscale/analysis/spatial/Cluster.hpp File Reference	336
8.3.1 Macro Definition Documentation	337
8.3.1.1 ERR_ORIGIN_DEPENDENT_VALUES	337

8.4	include/multiscale/analysis/spatial/cluster/SimulationClusterDetector.hpp File Reference	337
8.4.1	Macro Definition Documentation	338
8.4.1.1	DATAPOINT_THICKNESS	338
8.4.1.2	DATAPOINT_WIDTH	338
8.4.1.3	ENTITY_THRESH	339
8.4.1.4	THRESHOLD	339
8.4.1.5	THRESHOLD_MAX	339
8.5	include/multiscale/analysis/spatial/ClusterDetector.hpp File Reference	339
8.5.1	Macro Definition Documentation	340
8.5.1.1	EPS_MAX	340
8.5.1.2	EPS_MIN	341
8.5.1.3	EPS_REAL_MAX	341
8.5.1.4	EPS_REAL_MIN	341
8.5.1.5	MIN_POINTS_MAX	341
8.5.1.6	MIN_POINTS_MIN	341
8.5.1.7	OUTPUT_CLUSTEREDNESS	341
8.5.1.8	OUTPUT_PILE_UP	341
8.5.1.9	TRACKBAR_EPS	341
8.5.1.10	TRACKBAR_MINPOINTS	341
8.6	include/multiscale/analysis/spatial/DataPoint.hpp File Reference	342
8.7	include/multiscale/analysis/spatial/DBSCAN.hpp File Reference	343
8.7.1	Macro Definition Documentation	344
8.7.1.1	CLUSTERING_BORDER	344
8.7.1.2	CLUSTERING_NOISE	344
8.7.1.3	CLUSTERING_UNCLASSIFIED	344
8.8	include/multiscale/analysis/spatial/Detector.hpp File Reference	344
8.8.1	Macro Definition Documentation	345
8.8.1.1	ERR_INVALID_IMAGE	345
8.8.1.2	ERR_OUTPUT_FILE	345
8.8.1.3	ERR_OUTPUT_WITHOUT_DETECT	345
8.8.1.4	IMG_EXTENSION	346
8.8.1.5	KEY_ESC	346
8.8.1.6	KEY_SAVE	346
8.8.1.7	OUTPUT_EXTENSION	346
8.8.1.8	WIN_OUTPUT_IMAGE	346
8.9	include/multiscale/analysis/spatial/Entity.hpp File Reference	346
8.9.1	Macro Definition Documentation	348
8.9.1.1	ERR_DISTANCE	348
8.9.1.2	ERR_INPUT	348
8.9.1.3	OUTPUT_SEPARATOR	348

8.10 include/multiscale/analysis/spatial/factory/CircularMatFactory.hpp File Reference	348
8.10.1 Macro Definition Documentation	349
8.10.1.1 COLOURBAR_MAX_X	349
8.10.1.2 COLOURBAR_MAX_Y	350
8.10.1.3 ERR_UNIMPLEMENTED_METHOD	350
8.10.1.4 INPUT_IMG_HEIGHT	350
8.10.1.5 INPUT_IMG_WIDTH	350
8.10.1.6 INTENSITY_MAX	350
8.10.1.7 ROI_RADIUS	350
8.10.1.8 ROI_START_X	350
8.10.1.9 ROI_START_Y	350
8.11 include/multiscale/analysis/spatial/factory/RectangularMatFactory.hpp File Reference	351
8.11.1 Macro Definition Documentation	352
8.11.1.1 COLOURBAR_MAX_X	352
8.11.1.2 COLOURBAR_MAX_Y	352
8.11.1.3 ERR_CONC	352
8.11.1.4 INPUT_IMG_HEIGHT	352
8.11.1.5 INPUT_IMG_WIDTH	352
8.11.1.6 ROI_HEIGHT	352
8.11.1.7 ROI_START_X	352
8.11.1.8 ROI_START_Y	352
8.11.1.9 ROI_WIDTH	353
8.12 include/multiscale/analysis/spatial/MatFactory.hpp File Reference	353
8.12.1 Macro Definition Documentation	354
8.12.1.1 ERR_IMG_RESOLUTION	354
8.12.1.2 ERR_IN_EXTRA_DATA	354
8.12.1.3 ERR_INPUT_OPEN	354
8.13 include/multiscale/analysis/spatial/Region.hpp File Reference	354
8.13.1 Macro Definition Documentation	356
8.13.1.1 CONTOUR_CLOSED	356
8.13.1.2 CONTOUR_ORIENTED	356
8.14 include/multiscale/analysis/spatial/RegionDetector.hpp File Reference	356
8.14.1 Macro Definition Documentation	357
8.14.1.1 ALPHA_MAX	357
8.14.1.2 ALPHA_REAL_MAX	358
8.14.1.3 ALPHA_REAL_MIN	358
8.14.1.4 BETA_MAX	358
8.14.1.5 BETA_REAL_MAX	358
8.14.1.6 BETA_REAL_MIN	358
8.14.1.7 CANNY_THRESH_MAX	358

8.14.1.8 CONTOUR_AREA_ORIENTED	358
8.14.1.9 DISPLAY_LINE_THICKNESS	358
8.14.1.10 ENCLOSING_RECT_VERTICES	358
8.14.1.11 EPSILON_MAX	359
8.14.1.12 INTENSITY_MAX	359
8.14.1.13 KERNEL_MAX	359
8.14.1.14 MORPH_ITER_MAX	359
8.14.1.15 OUTPUT_CLUSTEREDNESS	359
8.14.1.16 OUTPUT_DENSITY	359
8.14.1.17 POLYGON_CLOSED	359
8.14.1.18 REGION_AREA_THRESH_MAX	359
8.14.1.19 THRESHOLD_CLUSTEREDNESS	359
8.14.1.20 THRESHOLD_MAX	359
8.14.1.21 TRACKBAR_ALPHA	360
8.14.1.22 TRACKBAR_BETA	360
8.14.1.23 TRACKBAR_CANNY	360
8.14.1.24 TRACKBAR_EPSILON	360
8.14.1.25 TRACKBAR_KERNEL	360
8.14.1.26 TRACKBAR_MORPH	360
8.14.1.27 TRACKBAR_REGION_AREA_THRESH	360
8.14.1.28 TRACKBAR_THRESHOLD	360
8.14.1.29 USE_CANNY_L2	360
8.15 include/multiscale/analysis/spatial/Shape2D.hpp File Reference	361
8.16 include/multiscale/analysis/spatial/Silhouette.hpp File Reference	361
8.17 include/multiscale/analysis/spatial/SpatialCollection2D.hpp File Reference	362
8.17.1 Macro Definition Documentation	363
8.17.1.1 CONVEX_HULL_CLOCKWISE	363
8.17.1.2 ERR_INPUT	363
8.17.1.3 ERR_UNDEFINED_SHAPE	363
8.17.1.4 OUTPUT_SEPARATOR	364
8.17.1.5 STR_CIRCLE	364
8.17.1.6 STR_RECTANGLE	364
8.17.1.7 STR_TRIANGLE	364
8.17.1.8 STR_UNDEFINED	364
8.18 include/multiscale/exception/CartesianToConcentrationsConverterException.hpp File Reference	364
8.19 include/multiscale/exception/CartesianToPolarConverterException.hpp File Reference	365
8.20 include/multiscale/exception/CircularMatFactoryException.hpp File Reference	366
8.21 include/multiscale/exception/ClusterException.hpp File Reference	367
8.22 include/multiscale/exception/DetectorException.hpp File Reference	368
8.23 include/multiscale/exception/EntityException.hpp File Reference	369

8.24	include/multiscale/exception/MatFactoryException.hpp File Reference	370
8.25	include/multiscale/exception/MinEnclosingTriangleFinderException.hpp File Reference	371
8.26	include/multiscale/exception/PolarCsvToInputFilesConverterException.hpp File Reference	372
8.27	include/multiscale/exception/RectangularCsvToInputFilesConverterException.hpp File Reference	373
8.28	include/multiscale/exception/RectangularEntityCsvToInputFilesConverterException.hpp File Reference	374
8.29	include/multiscale/exception/RectangularMatFactoryException.hpp File Reference	375
8.30	include/multiscale/exception/RegionException.hpp File Reference	376
8.31	include/multiscale/exception/SimulationClusterDetectorException.hpp File Reference	377
8.32	include/multiscale/util/Geometry2D.hpp File Reference	378
8.32.1	Macro Definition Documentation	379
8.32.1.1	MATRIX_START_INDEX	379
8.32.1.2	PI	379
8.33	include/multiscale/util/iterator/LexicographicNumberIterator.hpp File Reference	379
8.34	include/multiscale/util/iterator/NumberIteratorType.hpp File Reference	380
8.35	include/multiscale/util/iterator/StandardNumberIterator.hpp File Reference	381
8.36	include/multiscale/util/MinEnclosingTriangleFinder.hpp File Reference	382
8.36.1	Macro Definition Documentation	383
8.36.1.1	CONVEX_HULL_CLOCKWISE	383
8.36.1.2	ERR_MIDPOINT_SIDE_B	383
8.36.1.3	ERR_SIDE_B_GAMMA	383
8.36.1.4	ERR_TRIANGLE_VERTICES	383
8.36.1.5	ERR_VERTEX_C_ON_SIDE_B	383
8.36.1.6	INTERSECTS_ABOVE	383
8.36.1.7	INTERSECTS_BELOW	383
8.36.1.8	INTERSECTS_CRITICAL	384
8.36.1.9	INTERSECTS_LIMIT	384
8.36.1.10	VALIDATION_SIDE_A_TANGENT	384
8.36.1.11	VALIDATION_SIDE_B_TANGENT	384
8.36.1.12	VALIDATION_SIDES_FLUSH	384
8.37	include/multiscale/util/NumberIterator.hpp File Reference	384
8.38	include/multiscale/util/Numeric.hpp File Reference	385
8.39	include/multiscale/util/NumericRangeManipulator.hpp File Reference	386
8.40	include/multiscale/util/RGBColourGenerator.hpp File Reference	386
8.40.1	Macro Definition Documentation	387
8.40.1.1	HUE_MAX	387
8.40.1.2	HUE_MIN	387
8.40.1.3	SATURATION	387
8.40.1.4	VALUE	388
8.41	include/multiscale/util/StringManipulator.hpp File Reference	388

8.41.1 Macro Definition Documentation	388
8.41.1.1 DIR_SEPARATOR	389
8.42 include/multiscale/video/circular/AnnularSector.hpp File Reference	389
8.43 include/multiscale/video/circular/CartesianToPolarConverter.hpp File Reference	390
8.43.1 Macro Definition Documentation	391
8.43.1.1 ERR_CONC	391
8.43.1.2 ERR_IN_EXTRA_DATA	391
8.43.1.3 ERR_INPUT_OPEN	391
8.43.1.4 ERR_NEG_SIM_TIME	391
8.43.1.5 ERR_NONPOS_DIMENSION	391
8.43.1.6 OUTPUT_FILE_EXTENSION	391
8.43.1.7 RADIUS_MAX	391
8.43.1.8 RADIUS_MIN	392
8.44 include/multiscale/video/circular/PolarCsvToInputFilesConverter.hpp File Reference	392
8.44.1 Macro Definition Documentation	393
8.44.1.1 ERR_INPUT_OPEN	393
8.44.1.2 ERR_INVALID_VALUE_LINE	393
8.44.1.3 ERR_INVALID_VALUE_TOKEN	393
8.44.1.4 ERR_NEG_CONCENTRATION	393
8.44.1.5 ERR_NEG_SIM_TIME	393
8.44.1.6 ERR_NR_CONCENTRATIONS	394
8.44.1.7 ERR_SELECTED_CONCENTRATION_INDEX	394
8.44.1.8 INPUT_FILE_SEPARATOR	394
8.44.1.9 OUTPUT_EXTENSION	394
8.44.1.10 OUTPUT_FILE_SEPARATOR	394
8.44.1.11 OUTPUT_SEPARATOR	394
8.44.1.12 RADIUS_MIN	394
8.45 include/multiscale/video/circular/PolarGnuplotScriptGenerator.hpp File Reference	394
8.45.1 Macro Definition Documentation	396
8.45.1.1 CONTENT_IN	396
8.45.1.2 FOOTER_IN	396
8.45.1.3 GNUPLOT_EXTENSION	396
8.45.1.4 HEADER_IN	396
8.45.1.5 REPLACE_CONTENT_CONCENTRATION	396
8.45.1.6 REPLACE_CONTENT_END_ANGLE	396
8.45.1.7 REPLACE_CONTENT_INDEX	397
8.45.1.8 REPLACE_CONTENT_RADIUS	397
8.45.1.9 REPLACE_CONTENT_START_ANGLE	397
8.45.1.10 REPLACE_HEADER_FILENAME	397
8.45.1.11 REPLACE_HEADER_SIM_TIME	397

8.46 include/multiscale/video/rectangular/CartesianToConcentrationsConverter.hpp File Reference . . .	397
8.46.1 Macro Definition Documentation	398
8.46.1.1 ERR_CONC	398
8.46.1.2 ERR_IN_EXTRA_DATA	398
8.46.1.3 ERR_INPUT_OPEN	398
8.46.1.4 ERR_NEG_SIM_TIME	399
8.46.1.5 ERR_NONPOS_DIMENSION	399
8.46.1.6 OUTPUT_FILE_EXTENSION	399
8.46.1.7 RADIUS_MAX	399
8.46.1.8 RADIUS_MIN	399
8.47 include/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.hpp File Reference . . .	399
8.47.1 Macro Definition Documentation	400
8.47.1.1 ERR_INPUT_OPEN	400
8.47.1.2 ERR_INVALID_VALUE_LINE	401
8.47.1.3 ERR_INVALID_VALUE_TOKEN	401
8.47.1.4 ERR_NEG_CONCENTRATION	401
8.47.1.5 ERR_NEG_SIM_TIME	401
8.47.1.6 ERR_NR_CONCENTRATIONS	401
8.47.1.7 ERR_SELECTED_CONCENTRATION_INDEX	401
8.47.1.8 INPUT_FILE_SEPARATOR	401
8.47.1.9 OUTPUT_EXTENSION	401
8.47.1.10 OUTPUT_FILE_SEPARATOR	401
8.47.1.11 OUTPUT_SEPARATOR	401
8.48 include/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.hpp File Reference	401
8.48.1 Macro Definition Documentation	403
8.48.1.1 ERR_INPUT_OPEN	403
8.48.1.2 ERR_INVALID_NR_ENTITIES	403
8.48.1.3 ERR_INVALID_OX_COORDINATE	403
8.48.1.4 ERR_INVALID_OY_COORDINATE	403
8.48.1.5 ERR_INVALID_VALUE_LINE	403
8.48.1.6 ERR_INVALID_VALUE_TOKEN	403
8.48.1.7 ERR_MAX_NR_ENTITIES	403
8.48.1.8 ERR_NEG_SIM_TIME	403
8.48.1.9 ERR_NR_COORDINATES	404
8.48.1.10 INPUT_FILE_SEPARATOR	404
8.48.1.11 OUTPUT_EXTENSION	404
8.48.1.12 OUTPUT_FILE_SEPARATOR	404
8.48.1.13 OUTPUT_SEPARATOR	404
8.49 include/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.hpp File Reference . . .	404
8.49.1 Macro Definition Documentation	405

8.49.1.1	CONTENT_IN	405
8.49.1.2	FOOTER_IN	405
8.49.1.3	GNUPLOT_EXTENSION	406
8.49.1.4	HEADER_IN	406
8.49.1.5	OUTPUT_SEPARATOR	406
8.49.1.6	REPLACE_DIMENSION_EXTRA	406
8.49.1.7	REPLACE_HEADER_FILENAME	406
8.49.1.8	REPLACE_HEADER_HEIGHT	406
8.49.1.9	REPLACE_HEADER_SIM_TIME	406
8.49.1.10	REPLACE_HEADER_WIDTH	406
8.50	src/multiscale/analysis/spatial/CircularDetectRegions.cpp File Reference	406
8.50.1	Macro Definition Documentation	407
8.50.1.1	CONFIG_FILE	407
8.50.1.2	ERR_CODE	408
8.50.1.3	ERR_MSG	408
8.50.1.4	LABEL_ALPHA	408
8.50.1.5	LABEL_BETA	408
8.50.1.6	LABEL_BLUR_KERNEL_SIZE	408
8.50.1.7	LABEL_EPSILON	408
8.50.1.8	LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS	408
8.50.1.9	LABEL_REGION_AREA_THRESH	408
8.50.1.10	LABEL_ROOT_COMMENT	408
8.50.1.11	LABEL_THRESHOLD_VALUE	409
8.50.1.12	ROOT_COMMENT	409
8.50.2	Function Documentation	409
8.50.2.1	areValidParameters	409
8.50.2.2	initArgumentsConfig	409
8.50.2.3	loadDetectorParameterValues	410
8.50.2.4	loadDetectorParameterValues	411
8.50.2.5	main	411
8.50.2.6	printHelpInformation	412
8.50.2.7	printWrongParameters	413
8.50.2.8	saveDetectorParameterValues	413
8.50.2.9	saveDetectorParameterValues	414
8.51	src/multiscale/analysis/spatial/CircularityMeasure.cpp File Reference	415
8.52	src/multiscale/analysis/spatial/Cluster.cpp File Reference	415
8.53	src/multiscale/analysis/spatial/cluster/SimulationClusterDetector.cpp File Reference	416
8.54	src/multiscale/analysis/spatial/ClusterDetector.cpp File Reference	416
8.55	src/multiscale/analysis/spatial/DBSCAN.cpp File Reference	417
8.56	src/multiscale/analysis/spatial/Detector.cpp File Reference	418

8.57 src/multiscale/analysis/spatial/Entity.cpp File Reference	418
8.58 src/multiscale/analysis/spatial/factory/CircularMatFactory.cpp File Reference	418
8.59 src/multiscale/analysis/spatial/factory/RectangularMatFactory.cpp File Reference	419
8.60 src/multiscale/analysis/spatial/MatFactory.cpp File Reference	419
8.61 src/multiscale/analysis/spatial/RectangularDetectRegions.cpp File Reference	420
8.61.1 Macro Definition Documentation	421
8.61.1.1 CONFIG_FILE	421
8.61.1.2 ERR_CODE	421
8.61.1.3 ERR_MSG	421
8.61.1.4 LABEL_ALPHA	421
8.61.1.5 LABEL_BETA	421
8.61.1.6 LABEL_BLUR_KERNEL_SIZE	421
8.61.1.7 LABEL_EPSILON	421
8.61.1.8 LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS	422
8.61.1.9 LABEL_REGION_AREA_THRESH	422
8.61.1.10 LABEL_ROOT_COMMENT	422
8.61.1.11 LABEL_THRESHOLD_VALUE	422
8.61.1.12 ROOT_COMMENT	422
8.61.2 Function Documentation	422
8.61.2.1 areValidParameters	422
8.61.2.2 initArgumentsConfig	423
8.61.2.3 loadDetectorParameterValues	423
8.61.2.4 loadDetectorParameterValues	423
8.61.2.5 main	424
8.61.2.6 printHelpInformation	425
8.61.2.7 printWrongParameters	425
8.61.2.8 saveDetectorParameterValues	426
8.61.2.9 saveDetectorParameterValues	426
8.62 src/multiscale/analysis/spatial/Region.cpp File Reference	427
8.63 src/multiscale/analysis/spatial/RegionDetector.cpp File Reference	427
8.63.1 Function Documentation	428
8.63.1.1 convertVertices	428
8.64 src/multiscale/analysis/spatial/Silhouette.cpp File Reference	428
8.65 src/multiscale/analysis/spatial/SimulationDetectClusters.cpp File Reference	429
8.65.1 Macro Definition Documentation	430
8.65.1.1 CONFIG_FILE	430
8.65.1.2 ERR_CODE	430
8.65.1.3 ERR_MSG	430
8.65.1.4 LABEL_EPS	430
8.65.1.5 LABEL_MINPOINTS	430

8.65.1.6	LABEL_ROOT_COMMENT	430
8.65.1.7	ROOT_COMMENT	430
8.65.2	Function Documentation	431
8.65.2.1	isValidParameters	431
8.65.2.2	initArgumentsConfig	431
8.65.2.3	loadDetectorParameterValues	431
8.65.2.4	loadDetectorParameterValues	431
8.65.2.5	main	432
8.65.2.6	printHelpInformation	433
8.65.2.7	printWrongParameters	433
8.65.2.8	saveDetectorParameterValues	434
8.65.2.9	saveDetectorParameterValues	434
8.66	src/multiscale/analysis/spatial/SpatialCollection2D.cpp File Reference	434
8.67	src/multiscale/util/Geometry2D.cpp File Reference	435
8.68	src/multiscale/util/iterator/LexicographicNumberIterator.cpp File Reference	435
8.69	src/multiscale/util/iterator/StandardNumberIterator.cpp File Reference	436
8.70	src/multiscale/util/MinEnclosingTriangleFinder.cpp File Reference	437
8.71	src/multiscale/util/NumberIterator.cpp File Reference	437
8.72	src/multiscale/util/Numeric.cpp File Reference	437
8.73	src/multiscale/util/RGBColourGenerator.cpp File Reference	438
8.74	src/multiscale/util/StringManipulator.cpp File Reference	438
8.75	src/multiscale/video/circular/AnnularSector.cpp File Reference	439
8.75.1	Macro Definition Documentation	439
8.75.1.1	SEPARATOR	439
8.76	src/multiscale/video/circular/CartesianToPolarConverter.cpp File Reference	440
8.77	src/multiscale/video/circular/MapCartesianToPolarScript.cpp File Reference	440
8.77.1	Macro Definition Documentation	441
8.77.1.1	ERR_CODE	441
8.77.1.2	ERR_MSG	441
8.77.2	Function Documentation	441
8.77.2.1	isValidParameters	441
8.77.2.2	initArgumentsConfig	441
8.77.2.3	isValidOutputType	442
8.77.2.4	main	442
8.77.2.5	printHelpInformation	442
8.77.2.6	printWrongParameters	442
8.78	src/multiscale/video/circular/PolarCsvToInputFilesConverter.cpp File Reference	443
8.79	src/multiscale/video/circular/PolarGnuplotScriptGenerator.cpp File Reference	443
8.80	src/multiscale/video/circular/PolarMapCsvToInputFiles.cpp File Reference	444
8.80.1	Macro Definition Documentation	444

8.80.1.1	ERR_CODE	444
8.80.1.2	ERR_MSG	445
8.80.2	Function Documentation	445
8.80.2.1	isValidParameters	445
8.80.2.2	initArgumentsConfig	445
8.80.2.3	isValidNrOfConcentrationsForPosition	445
8.80.2.4	main	446
8.80.2.5	printHelpInformation	446
8.80.2.6	printWrongParameters	446
8.80.2.7	setLogScaling	446
8.80.2.8	setNumberIteratorType	447
8.80.2.9	setSelectedConcentrationIndex	447
8.81	src/multiscale/video/rectangular/CartesianToConcentrationsConverter.cpp File Reference	448
8.82	src/multiscale/video/rectangular/MapCartesianToScript.cpp File Reference	448
8.82.1	Macro Definition Documentation	449
8.82.1.1	ERR_CODE	449
8.82.1.2	ERR_MSG	449
8.82.2	Function Documentation	449
8.82.2.1	isValidParameters	449
8.82.2.2	initArgumentsConfig	449
8.82.2.3	main	449
8.82.2.4	printHelpInformation	450
8.82.2.5	printWrongParameters	450
8.83	src/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.cpp File Reference	450
8.84	src/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.cpp File Reference	451
8.85	src/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.cpp File Reference	451
8.86	src/multiscale/video/rectangular/RectangularMapCsvToInputFiles.cpp File Reference	452
8.86.1	Macro Definition Documentation	452
8.86.1.1	ERR_CODE	452
8.86.1.2	ERR_MSG	453
8.86.2	Function Documentation	453
8.86.2.1	isValidParameters	453
8.86.2.2	initArgumentsConfig	453
8.86.2.3	isValidNrOfConcentrationsForPosition	453
8.86.2.4	main	454
8.86.2.5	printHelpInformation	454
8.86.2.6	printWrongParameters	454
8.86.2.7	setLogScaling	454
8.86.2.8	setNumberIteratorType	454
8.86.2.9	setSelectedConcentrationIndex	454

8.87 src/multiscale/video/rectangular/RectangularMapEntityCsvToInputFiles.cpp File Reference	455
8.87.1 Macro Definition Documentation	455
8.87.1.1 ERR_CODE	455
8.87.1.2 ERR_MSG	455
8.87.2 Function Documentation	456
8.87.2.1 isValidParameters	456
8.87.2.2 initArgumentsConfig	456
8.87.2.3 main	456
8.87.2.4 printHelpInformation	457
8.87.2.5 printWrongParameters	457
8.87.2.6 setNumberIteratorType	457
Index	457

Chapter 1

Multiscale

1.1 Brief description

The "Multiscale" software is a multiscale model checker implemented during the PhD research project carried out by Ovidiu Parvu, Brunel University, London, United Kingdom, October 2012 - present.

1.2 Contact

For more information, comments, recommendations or suggestions please visit the author's [institutional web page](#), where contact details are provided.

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

multiscale	11
multiscale::analysis	12
multiscale::video	13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

multiscale::video::AnnularSector	15
multiscale::video::CartesianToConcentrationsConverter	18
multiscale::video::CartesianToPolarConverter	26
multiscale::analysis::CircularityMeasure	34
multiscale::analysis::DataPoint	78
multiscale::analysis::Entity	103
multiscale::analysis::DBSCAN	81
multiscale::analysis::Detector	89
multiscale::analysis::ClusterDetector	59
multiscale::analysis::SimulationClusterDetector	290
multiscale::analysis::RegionDetector	256
multiscale::Geometry2D	112
multiscale::analysis::MatFactory	145
multiscale::analysis::CircularMatFactory	35
multiscale::analysis::RectangularMatFactory	240
multiscale::MinEnclosingTriangleFinder	152
multiscale::NumberIterator	187
multiscale::LexicographicNumberIterator	137
multiscale::StandardNumberIterator	325
multiscale::Numeric	194
multiscale::NumericRangeManipulator	198
multiscale::video::PolarCsvToInputFilesConverter	199
multiscale::video::PolarGnuplotScriptGenerator	211
multiscale::video::RectangularCsvToInputFilesConverter	215
multiscale::video::RectangularEntityCsvToInputFilesConverter	226
multiscale::video::RectangularGnuplotScriptGenerator	236
multiscale::RGBColourGenerator	284
runtime_error	
multiscale::CartesianToConcentrationsConverterException	24
multiscale::CartesianToPolarConverterException	32
multiscale::CircularMatFactoryException	41
multiscale::ClusterException	77
multiscale::DetectorException	102
multiscale::EntityException	111
multiscale::MatFactoryException	150
multiscale::MinEnclosingTriangleFinderException	185
multiscale::PolarCsvToInputFilesConverterException	209

multiscale::RectangularCsvToInputFilesConverterException	224
multiscale::RectangularEntityCsvToInputFilesConverterException	234
multiscale::RectangularMatFactoryException	246
multiscale::RegionException	282
multiscale::SimulationClusterDetectorException	303
multiscale::analysis::Silhouette	287
multiscale::analysis::SpatialCollection2D	305
multiscale::analysis::Cluster	43
multiscale::analysis::Region	247
multiscale::StringManipulator	329

Chapter 4

Class Index

4.1 Class List

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

multiscale::video::AnnularSector	An annular sector is the basic element in the considered circular geometry	15
multiscale::video::CartesianToConcentrationsConverter	Scale the values of the rectangular geometry grid cells	18
multiscale::CartesianToConcentrationsConverterException	Exception class for the CartesianToConcentrationsConverter class	24
multiscale::video::CartesianToPolarConverter	Converter from the rectangular geometry grid cells to annular sectors	26
multiscale::CartesianToPolarConverterException	Exception class for the CartesianToPolarConverter class	32
multiscale::analysis::CircularityMeasure	Class for computing the circularity measure for the given collection of points	34
multiscale::analysis::CircularMatFactory	Class for creating a Mat object considering a circular grid	35
multiscale::CircularMatFactoryException	Exception class for the CircularMatFactory instances	41
multiscale::analysis::Cluster	Class for representing a cluster of entities in an image	43
multiscale::analysis::ClusterDetector	Class for detecting clusters in 2D images	59
multiscale::ClusterException	Exception class for the Cluster instances	77
multiscale::analysis::DataPoint		78
multiscale::analysis::DBSCAN		81
multiscale::analysis::Detector		89
multiscale::DetectorException	Exception class for the Detector class	102
multiscale::analysis::Entity	Class for representing an entity in an image (e.g. cell, organism etc.)	103
multiscale::EntityException	Exception class for the Entity instances	111
multiscale::Geometry2D	Two-dimensional geometric operations	112
multiscale::LexicographicNumberIterator	Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "_"	137
multiscale::analysis::MatFactory	Class for creating a Mat object	145

multiscale::MatFactoryException	Exception class for the MatFactory class	150
multiscale::MinEnclosingTriangleFinder	Class for computing the minimum area enclosing triangle for a given polygon	152
multiscale::MinEnclosingTriangleFinderException	Exception class for the minimum area enclosing triangle module	185
multiscale::NumberIterator	Abstract class representing a number iterator	187
multiscale::Numeric	Class for manipulating numbers (shorts, ints, floats, doubles etc.)	194
multiscale::NumericRangeManipulator	Operations for ranges of numeric values	198
multiscale::video::PolarCsvToInputFilesConverter	Csv file to input file converter considering polar coordinates	199
multiscale::PolarCsvToInputFilesConverterException	Exception class for the PolarCsvToInputFilesConverter class	209
multiscale::video::PolarGnuplotScriptGenerator	Gnuplot script generator from the provided annular sectors	211
multiscale::video::RectangularCsvToInputFilesConverter	Csv file to input file converter considering cartesian coordinates	215
multiscale::RectangularCsvToInputFilesConverterException	Exception class for the RectangularCsvToInputFilesConverter class	224
multiscale::video::RectangularEntityCsvToInputFilesConverter	Csv entity file to input file converter considering cartesian coordinates	226
multiscale::RectangularEntityCsvToInputFilesConverterException	Exception class for the RectangularEntityCsvToInputFilesConverter class	234
multiscale::video::RectangularGnuplotScriptGenerator	Gnuplot script generator from the provided concentrations considering a rectangular geometry	236
multiscale::analysis::RectangularMatFactory	Class for creating a Mat object considering a rectangular grid	240
multiscale::RectangularMatFactoryException	Exception class for the RectangularMatFactory instances	246
multiscale::analysis::Region	Class for representing a region	247
multiscale::analysis::RegionDetector	Class for detecting regions of high intensity in grayscale images	256
multiscale::RegionException	Exception class for the Region instances	282
multiscale::RGBColourGenerator	Generate a RGB colour	284
multiscale::analysis::Silhouette		287
multiscale::analysis::SimulationClusterDetector	Class for detecting clusters in 2D images obtained from simulations	290
multiscale::SimulationClusterDetectorException	Exception class for the SimulationClusterDetector instances	303
multiscale::analysis::SpatialCollection2D	Class for representing a collection of objects in 2D space	305
multiscale::StandardNumberIterator	Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached	325
multiscale::StringManipulator	Class for manipulating strings	329

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

include/multiscale/analysis/spatial/CircularityMeasure.hpp	335
include/multiscale/analysis/spatial/Cluster.hpp	336
include/multiscale/analysis/spatial/ClusterDetector.hpp	339
include/multiscale/analysis/spatial/DataPoint.hpp	342
include/multiscale/analysis/spatial/DBSCAN.hpp	343
include/multiscale/analysis/spatial/Detector.hpp	344
include/multiscale/analysis/spatial/Entity.hpp	346
include/multiscale/analysis/spatial/MatFactory.hpp	353
include/multiscale/analysis/spatial/Region.hpp	354
include/multiscale/analysis/spatial/RegionDetector.hpp	356
include/multiscale/analysis/spatial/Shape2D.hpp	361
include/multiscale/analysis/spatial/Silhouette.hpp	361
include/multiscale/analysis/spatial/SpatialCollection2D.hpp	362
include/multiscale/analysis/spatial/cluster/SimulationClusterDetector.hpp	337
include/multiscale/analysis/spatial/factory/CircularMatFactory.hpp	348
include/multiscale/analysis/spatial/factory/RectangularMatFactory.hpp	351
include/multiscale/exception/CartesianToConcentrationsConverterException.hpp	364
include/multiscale/exception/CartesianToPolarConverterException.hpp	365
include/multiscale/exception/CircularMatFactoryException.hpp	366
include/multiscale/exception/ClusterException.hpp	367
include/multiscale/exception/DetectorException.hpp	368
include/multiscale/exception/EntityException.hpp	369
include/multiscale/exception/MatFactoryException.hpp	370
include/multiscale/exception/MinEnclosingTriangleFinderException.hpp	371
include/multiscale/exception/PolarCsvToInputFilesConverterException.hpp	372
include/multiscale/exception/RectangularCsvToInputFilesConverterException.hpp	373
include/multiscale/exception/RectangularEntityCsvToInputFilesConverterException.hpp	374
include/multiscale/exception/RectangularMatFactoryException.hpp	375
include/multiscale/exception/RegionException.hpp	376
include/multiscale/exception/SimulationClusterDetectorException.hpp	377
include/multiscale/util/Geometry2D.hpp	378
include/multiscale/util/MinEnclosingTriangleFinder.hpp	382
include/multiscale/util/NumberIterator.hpp	384
include/multiscale/util/Numeric.hpp	385
include/multiscale/util/NumericRangeManipulator.hpp	386
include/multiscale/util/RGBColourGenerator.hpp	386
include/multiscale/util/StringManipulator.hpp	388
include/multiscale/util/iterator/LexicographicNumberIterator.hpp	379

include/multiscale/util/iterator/NumberIteratorType.hpp	380
include/multiscale/util/iterator/StandardNumberIterator.hpp	381
include/multiscale/video/circular/AnnularSector.hpp	389
include/multiscale/video/circular/CartesianToPolarConverter.hpp	390
include/multiscale/video/circular/PolarCsvToInputFilesConverter.hpp	392
include/multiscale/video/circular/PolarGnuplotScriptGenerator.hpp	394
include/multiscale/video/rectangular/CartesianToConcentrationsConverter.hpp	397
include/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.hpp	399
include/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.hpp	401
include/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.hpp	404
src/multiscale/analysis/spatial/CircularDetectRegions.cpp	406
src/multiscale/analysis/spatial/CircularityMeasure.cpp	415
src/multiscale/analysis/spatial/Cluster.cpp	415
src/multiscale/analysis/spatial/ClusterDetector.cpp	416
src/multiscale/analysis/spatial/DBSCAN.cpp	417
src/multiscale/analysis/spatial/Detector.cpp	418
src/multiscale/analysis/spatial/Entity.cpp	418
src/multiscale/analysis/spatial/MatFactory.cpp	419
src/multiscale/analysis/spatial/RectangularDetectRegions.cpp	420
src/multiscale/analysis/spatial/Region.cpp	427
src/multiscale/analysis/spatial/RegionDetector.cpp	427
src/multiscale/analysis/spatial/Silhouette.cpp	428
src/multiscale/analysis/spatial/SimulationDetectClusters.cpp	429
src/multiscale/analysis/spatial/SpatialCollection2D.cpp	434
src/multiscale/analysis/spatial/cluster/SimulationClusterDetector.cpp	416
src/multiscale/analysis/spatial/factory/CircularMatFactory.cpp	418
src/multiscale/analysis/spatial/factory/RectangularMatFactory.cpp	419
src/multiscale/util/Geometry2D.cpp	435
src/multiscale/util/MinEnclosingTriangleFinder.cpp	437
src/multiscale/util/NumberIterator.cpp	437
src/multiscale/util/Numeric.cpp	437
src/multiscale/util/RGBColourGenerator.cpp	438
src/multiscale/util/StringManipulator.cpp	438
src/multiscale/util/iterator/LexicographicNumberIterator.cpp	435
src/multiscale/util/iterator/StandardNumberIterator.cpp	436
src/multiscale/video/circular/AnnularSector.cpp	439
src/multiscale/video/circular/CartesianToPolarConverter.cpp	440
src/multiscale/video/circular/MapCartesianToPolarScript.cpp	440
src/multiscale/video/circular/PolarCsvToInputFilesConverter.cpp	443
src/multiscale/video/circular/PolarGnuplotScriptGenerator.cpp	443
src/multiscale/video/circular/PolarMapCsvToInputFiles.cpp	444
src/multiscale/video/rectangular/CartesianToConcentrationsConverter.cpp	448
src/multiscale/video/rectangular/MapCartesianToScript.cpp	448
src/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.cpp	450
src/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.cpp	451
src/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.cpp	451
src/multiscale/video/rectangular/RectangularMapCsvToInputFiles.cpp	452
src/multiscale/video/rectangular/RectangularMapEntityCsvToInputFiles.cpp	455

Chapter 6

Namespace Documentation

6.1 multiscale Namespace Reference

Namespaces

- namespace [analysis](#)
- namespace [video](#)

Classes

- class [CartesianToConcentrationsConverterException](#)
Exception class for the CartesianToConcentrationsConverter class.
- class [CartesianToPolarConverterException](#)
Exception class for the CartesianToPolarConverter class.
- class [CircularMatFactoryException](#)
Exception class for the CircularMatFactory instances.
- class [ClusterException](#)
Exception class for the Cluster instances.
- class [DetectorException](#)
Exception class for the Detector class.
- class [EntityException](#)
Exception class for the Entity instances.
- class [MatFactoryException](#)
Exception class for the MatFactory class.
- class [MinEnclosingTriangleFinderException](#)
Exception class for the minimum area enclosing triangle module.
- class [PolarCsvToInputFilesConverterException](#)
Exception class for the PolarCsvToInputFilesConverter class.
- class [RectangularCsvToInputFilesConverterException](#)
Exception class for the RectangularCsvToInputFilesConverter class.
- class [RectangularEntityCsvToInputFilesConverterException](#)
Exception class for the RectangularEntityCsvToInputFilesConverter class.
- class [RectangularMatFactoryException](#)
Exception class for the RectangularMatFactory instances.
- class [RegionException](#)
Exception class for the Region instances.
- class [SimulationClusterDetectorException](#)

- class [Geometry2D](#)
Two-dimensional geometric operations.
- class [LexicographicNumberIterator](#)
Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "—".
- class [StandardNumberIterator](#)
Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.
- class [MinEnclosingTriangleFinder](#)
Class for computing the minimum area enclosing triangle for a given polygon.
- class [NumberIterator](#)
Abstract class representing a number iterator.
- class [Numeric](#)
Class for manipulating numbers (shorts, ints, floats, doubles etc.)
- class [NumericRangeManipulator](#)
Operations for ranges of numeric values.
- class [RGBColourGenerator](#)
Generate a RGB colour.
- class [StringManipulator](#)
Class for manipulating strings.

Enumerations

- enum [NumberIteratorType](#) { **STANDARD** = 1, **LEXICOGRAPHIC** = 2 }
The type of the number iterator.

6.1.1 Enumeration Type Documentation

6.1.1.1 enum multiscale::NumberIteratorType

The type of the number iterator.

Enumerator

- STANDARD** Standard number iterator
- LEXICOGRAPHIC** Lexicographic number iterator

Definition at line 7 of file NumberIteratorType.hpp.

6.2 multiscale::analysis Namespace Reference

Classes

- class [CircularityMeasure](#)
Class for computing the circularity measure for the given collection of points.
- class [SimulationClusterDetector](#)
Class for detecting clusters in 2D images obtained from simulations.
- class [Cluster](#)
Class for representing a cluster of entities in an image.
- class [ClusterDetector](#)
Class for detecting clusters in 2D images.

- class [DataPoint](#)
- class [DBSCAN](#)
- class [Detector](#)
- class [Entity](#)
Class for representing an entity in an image (e.g. cell, organism etc.)
- class [CircularMatFactory](#)
Class for creating a Mat object considering a circular grid.
- class [RectangularMatFactory](#)
Class for creating a Mat object considering a rectangular grid.
- class [MatFactory](#)
Class for creating a Mat object.
- class [Region](#)
Class for representing a region.
- class [RegionDetector](#)
Class for detecting regions of high intensity in grayscale images.
- class [Silhouette](#)
- class [SpatialCollection2D](#)
Class for representing a collection of objects in 2D space.

Enumerations

- enum [Shape2D](#) { **Triangle** = 1, **Rectangle** = 2, **Circle** = 3, **Undefined** = 4 }
Enumeration for determining the type of a 2D shape.

6.2.1 Enumeration Type Documentation

6.2.1.1 enum multiscale::analysis::Shape2D

Enumeration for determining the type of a 2D shape.

Enumerator

- Triangle** Triangular 2D shape
Rectangle Rectangular 2D shape
Circle Circular 2D shape
Undefined Undefined 2D shape

Definition at line 10 of file Shape2D.hpp.

6.3 multiscale::video Namespace Reference

Classes

- class [AnnularSector](#)
An annular sector is the basic element in the considered circular geometry.
- class [CartesianToPolarConverter](#)
Converter from the rectangular geometry grid cells to annular sectors.
- class [PolarCsvToInputFilesConverter](#)
Csv file to input file converter considering polar coordinates.
- class [PolarGnuplotScriptGenerator](#)

Gnuplot script generator from the provided annular sectors.

- class [CartesianToConcentrationsConverter](#)

Scale the values of the rectangular geometry grid cells.

- class [RectangularCsvToInputFilesConverter](#)

Csv file to input file converter considering cartesian coordinates.

- class [RectangularEntityCsvToInputFilesConverter](#)

Csv entity file to input file converter considering cartesian coordinates.

- class [RectangularGnuplotScriptGenerator](#)

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

Chapter 7

Class Documentation

7.1 multiscale::video::AnnularSector Class Reference

An annular sector is the basic element in the considered circular geometry.

```
#include <AnnularSector.hpp>
```

Collaboration diagram for multiscale::video::AnnularSector:

multiscale::video:: AnnularSector
- startingRadius - endingRadius - startingAngle - endingAngle - concentration
+ AnnularSector() + ~AnnularSector() + initialise() + getConcentration() + getEndingAngle() + getEndingRadius() + getStartingAngle() + getStartingRadius() + toString()

Public Member Functions

- [AnnularSector \(\)](#)
- [~AnnularSector \(\)](#)

- void `initialise` (double `startingRadius`, double `endingRadius`, double `startingAngle`, double `endingAngle`, double `concentration`)
Initialise the members of the class.
- double `getConcentration` () const
Get the value of the concentration.
- double `getEndingAngle` () const
Get the value of the ending angle.
- double `getEndingRadius` () const
Get the value of the ending radius.
- double `getStartingAngle` () const
Get the value of the starting angle.
- double `getStartingRadius` () const
Get the value of the starting radius.
- string `toString` ()
Get the string representation of the annular sector.

Private Attributes

- double `startingRadius`
- double `endingRadius`
- double `startingAngle`
- double `endingAngle`
- double `concentration`

7.1.1 Detailed Description

An annular sector is the basic element in the considered circular geometry.

More information about annuli and sectors of annuli can be found online (e.g. Wikipedia).

Definition at line 16 of file AnnularSector.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AnnularSector::AnnularSector ()

Definition at line 11 of file AnnularSector.cpp.

References `concentration`, `endingAngle`, `endingRadius`, `startingAngle`, and `startingRadius`.

7.1.2.2 AnnularSector::~AnnularSector ()

Definition at line 19 of file AnnularSector.cpp.

7.1.3 Member Function Documentation

7.1.3.1 double AnnularSector::getConcentration () const

Get the value of the concentration.

Definition at line 30 of file AnnularSector.cpp.

References `concentration`.

7.1.3.2 double AnnularSector::getEndingAngle() const

Get the value of the ending angle.

Definition at line 34 of file AnnularSector.cpp.

References endingAngle.

7.1.3.3 double AnnularSector::getEndingRadius() const

Get the value of the ending radius.

Definition at line 38 of file AnnularSector.cpp.

References endingRadius.

7.1.3.4 double AnnularSector::getStartingAngle() const

Get the value of the starting angle.

Definition at line 42 of file AnnularSector.cpp.

References startingAngle.

7.1.3.5 double AnnularSector::getStartingRadius() const

Get the value of the starting radius.

Definition at line 46 of file AnnularSector.cpp.

References startingRadius.

7.1.3.6 void AnnularSector::initialise(double *startingRadius*, double *endingRadius*, double *startingAngle*, double *endingAngle*, double *concentration*)

Initialise the members of the class.

Parameters

<i>startingRadius</i>	Starting radius
<i>endingRadius</i>	Ending radius
<i>startingAngle</i>	Starting angle
<i>endingAngle</i>	Ending angle
<i>concentration</i>	Concentration

Definition at line 21 of file AnnularSector.cpp.

References concentration, endingAngle, endingRadius, startingAngle, and startingRadius.

7.1.3.7 string AnnularSector::toString()

Get the string representation of the annular sector.

Definition at line 50 of file AnnularSector.cpp.

References concentration, endingAngle, endingRadius, SEPARATOR, startingAngle, and startingRadius.

7.1.4 Member Data Documentation

7.1.4.1 double multiscale::video::AnnularSector::concentration [private]

Definition at line 24 of file AnnularSector.hpp.

Referenced by AnnularSector(), getConcentration(), initialise(), and toString().

7.1.4.2 double multiscale::video::AnnularSector::endingAngle [private]

Definition at line 23 of file AnnularSector.hpp.

Referenced by AnnularSector(), getEndingAngle(), initialise(), and toString().

7.1.4.3 double multiscale::video::AnnularSector::endingRadius [private]

Definition at line 21 of file AnnularSector.hpp.

Referenced by AnnularSector(), getEndingRadius(), initialise(), and toString().

7.1.4.4 double multiscale::video::AnnularSector::startingAngle [private]

Definition at line 22 of file AnnularSector.hpp.

Referenced by AnnularSector(), getStartingAngle(), initialise(), and toString().

7.1.4.5 double multiscale::video::AnnularSector::startingRadius [private]

Definition at line 20 of file AnnularSector.hpp.

Referenced by AnnularSector(), getStartingRadius(), initialise(), and toString().

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

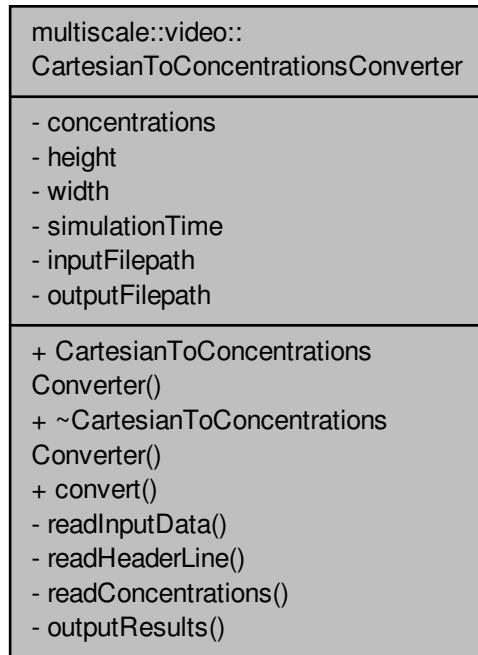
- include/multiscale/video/circular/[AnnularSector.hpp](#)
- src/multiscale/video/circular/[AnnularSector.cpp](#)

7.2 multiscale::video::CartesianToConcentrationsConverter Class Reference

Scale the values of the rectangular geometry grid cells.

```
#include <CartesianToConcentrationsConverter.hpp>
```

Collaboration diagram for multiscale::video::CartesianToConcentrationsConverter:



Public Member Functions

- [CartesianToConcentrationsConverter](#) (const string &[inputfilepath](#), const string &[outputfilepath](#))
- [~CartesianToConcentrationsConverter](#) ()
- void [convert](#) ()

Start the conversion.

Private Member Functions

- void [readInputData](#) ()
Read the input data.
- void [readHeaderLine](#) (ifstream &fin)
Read the header line.
- void [readConcentrations](#) (ifstream &fin)
Read the concentrations.
- void [outputResults](#) ()
Output the results.

Private Attributes

- vector< double > [concentrations](#)
- unsigned long [height](#)
- unsigned long [width](#)

- double `simulationTime`
- string `inputFilepath`
- string `outputFilepath`

7.2.1 Detailed Description

Scale the values of the rectangular geometry grid cells.

Definition at line 26 of file `CartesianToConcentrationsConverter.hpp`.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 `CartesianToConcentrationsConverter::CartesianToConcentrationsConverter (const string & inputFilepath, const string & outputFilepath)`

Definition at line 15 of file `CartesianToConcentrationsConverter.cpp`.

References `height`, `simulationTime`, and `width`.

7.2.2.2 `CartesianToConcentrationsConverter::~CartesianToConcentrationsConverter ()`

Definition at line 24 of file `CartesianToConcentrationsConverter.cpp`.

7.2.3 Member Function Documentation

7.2.3.1 `void CartesianToConcentrationsConverter::convert ()`

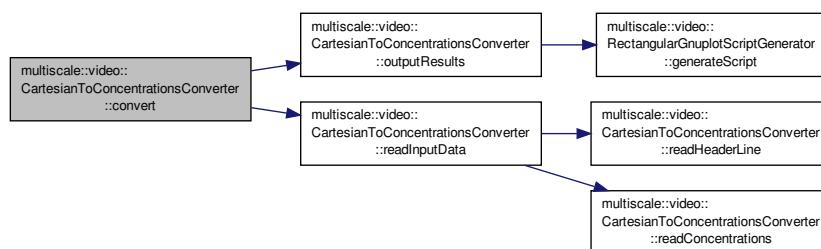
Start the conversion.

Definition at line 26 of file `CartesianToConcentrationsConverter.cpp`.

References `outputResults()`, and `readInputData()`.

Referenced by `main()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.2 void CartesianToConcentrationsConverter::outputResults() [private]

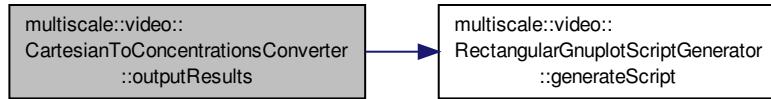
Output the results.

Definition at line 84 of file `CartesianToConcentrationsConverter.cpp`.

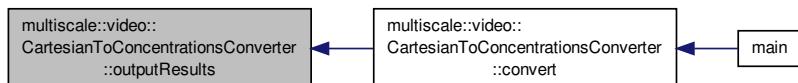
References `concentrations`, `multiscale::video::RectangularGnuplotScriptGenerator::generateScript()`, `height`, `outputFilepath`, `simulationTime`, and `width`.

Referenced by `convert()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.3 void CartesianToConcentrationsConverter::readConcentrations(ifstream & fin) [private]

Read the concentrations.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 64 of file `CartesianToConcentrationsConverter.cpp`.

References concentrations, `ERR_CONC`, height, and width.

Referenced by `readInputData()`.

Here is the caller graph for this function:



7.2.3.4 void `CartesianToConcentrationsConverter::readHeaderLine (ifstream & fin) [private]`

Read the header line.

The header line contains values for number of concentric circles, number of sectors and simulation time

Parameters

<code>fin</code>	Input file stream
------------------	-------------------

Definition at line 55 of file `CartesianToConcentrationsConverter.cpp`.

References `ERR_NEG_SIM_TIME`, `ERR_NONPOS_DIMENSION`, height, `simulationTime`, and `width`.

Referenced by `readInputData()`.

Here is the caller graph for this function:



7.2.3.5 void `CartesianToConcentrationsConverter::readInputData () [private]`

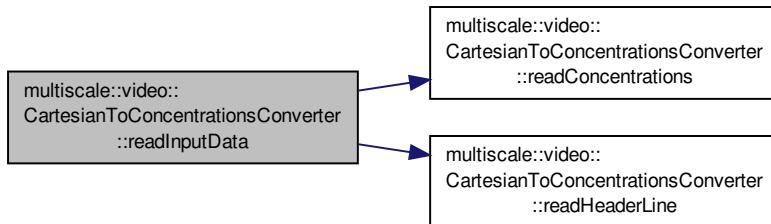
Read the input data.

Definition at line 31 of file `CartesianToConcentrationsConverter.cpp`.

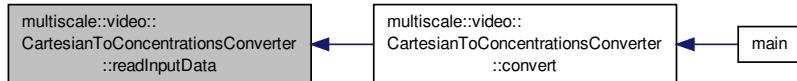
References `ERR_IN_EXTRA_DATA`, `ERR_INPUT_OPEN`, `inputFilepath`, `readConcentrations()`, and `readHeaderLine()`.

Referenced by `convert()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.4 Member Data Documentation

7.2.4.1 `vector<double> multiscale::video::CartesianToConcentrationsConverter::concentrations` [private]

Concentrations received as input

Definition at line 30 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `outputResults()`, and `readConcentrations()`.

7.2.4.2 `unsigned long multiscale::video::CartesianToConcentrationsConverter::height` [private]

Height of the grid

Definition at line 32 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, `readConcentrations()`, and `readHeaderLine()`.

7.2.4.3 `string multiscale::video::CartesianToConcentrationsConverter::inputFilepath` [private]

Path to the input file

Definition at line 36 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `readInputData()`.

7.2.4.4 `string multiscale::video::CartesianToConcentrationsConverter::outputFilepath` [private]

Path to the output file

Definition at line 37 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `outputResults()`.

7.2.4.5 double multiscale::video::CartesianToConcentrationsConverter::simulationTime [private]

Simulation time

Definition at line 34 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, and `readHeaderLine()`.

7.2.4.6 unsigned long multiscale::video::CartesianToConcentrationsConverter::width [private]

Width of the grid

Definition at line 33 of file `CartesianToConcentrationsConverter.hpp`.

Referenced by `CartesianToConcentrationsConverter()`, `outputResults()`, `readConcentrations()`, and `readHeaderLine()`.

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

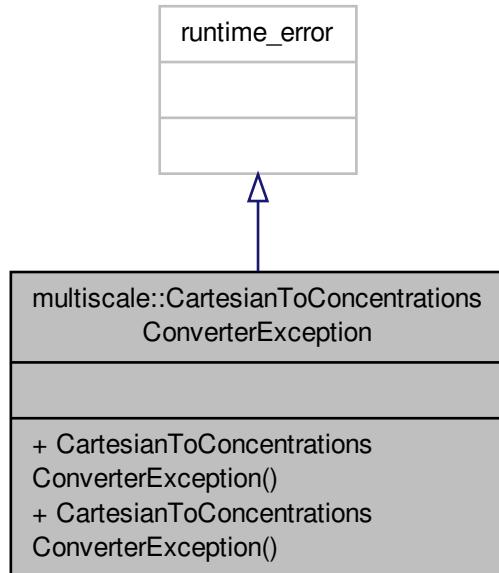
- `include/multiscale/video/rectangular/CartesianToConcentrationsConverter.hpp`
- `src/multiscale/video/rectangular/CartesianToConcentrationsConverter.cpp`

7.3 multiscale::CartesianToConcentrationsConverterException Class Reference

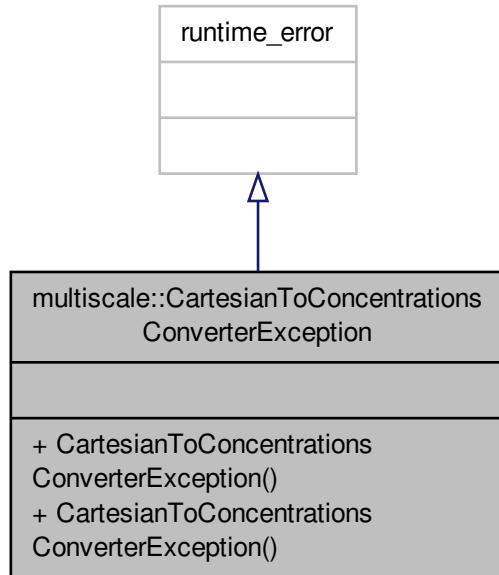
Exception class for the `CartesianToConcentrationsConverter` class.

```
#include <CartesianToConcentrationsConverterException.hpp>
```

Inheritance diagram for `multiscale::CartesianToConcentrationsConverterException`:



Collaboration diagram for multiscale::CartesianToConcentrationsConverterException:



Public Member Functions

- [CartesianToConcentrationsConverterException](#) (const string &msg)
- [CartesianToConcentrationsConverterException](#) (const char *msg)

7.3.1 Detailed Description

Exception class for the `CartesianToConcentrationsConverter` class.

Definition at line 13 of file `CartesianToConcentrationsConverterException.hpp`.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 multiscale::CartesianToConcentrationsConverterException::CartesianToConcentrationsConverterException (const string & msg) [inline]

Definition at line 17 of file `CartesianToConcentrationsConverterException.hpp`.

7.3.2.2 multiscale::CartesianToConcentrationsConverterException::CartesianToConcentrationsConverterException (const char * msg) [inline]

Definition at line 18 of file `CartesianToConcentrationsConverterException.hpp`.

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

- [include/multiscale/exception/CartesianToConcentrationsConverterException.hpp](#)

7.4 multiscale::video::CartesianToPolarConverter Class Reference

Converter from the rectangular geometry grid cells to annular sectors.

```
#include <CartesianToPolarConverter.hpp>
```

Collaboration diagram for multiscale::video::CartesianToPolarConverter:

multiscale::video:: CartesianToPolarConverter
<ul style="list-style-type: none"> - annularSectors - concentrations - nrOfConcentricCircles - nrOfSectors - simulationTime - inputfilepath - outputfilepath
<ul style="list-style-type: none"> + CartesianToPolarConverter() + ~CartesianToPolarConverter() + convert() - readInputData() - readHeaderLine() - readConcentrations() - transformToAnnularSectors() - outputResultsAsFile() - outputResultsAsScript()

Public Member Functions

- [CartesianToPolarConverter](#) (const string &[inputfilepath](#), const string &[outputfilepath](#))
- [~CartesianToPolarConverter](#) ()
- void [convert](#) (bool [outputToScript](#))

Start the conversion.

Private Member Functions

- void [readInputData](#) ()
Read the input data.
- void [readHeaderLine](#) (ifstream &[fin](#))
Read the header line.
- void [readConcentrations](#) (ifstream &[fin](#))
Read the concentrations.
- void [transformToAnnularSectors](#) ()
Convert the concentrations to annular sectors.

- void `outputResultsAsFile ()`
Output the results as a plain file.
- void `outputResultsAsScript ()`
Output the results as a gnuplot script.

Private Attributes

- vector< `AnnularSector` > `annularSectors`
- vector< double > `concentrations`
- unsigned long `nrOfConcentricCircles`
- unsigned long `nrOfSectors`
- double `simulationTime`
- string `inputFilepath`
- string `outputFilepath`

7.4.1 Detailed Description

Converter from the rectangular geometry grid cells to annular sectors.

Definition at line 28 of file `CartesianToPolarConverter.hpp`.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 `CartesianToPolarConverter::CartesianToPolarConverter (const string & inputFilepath, const string & outputFilepath)`

Definition at line 15 of file `CartesianToPolarConverter.cpp`.

References `nrOfConcentricCircles`, `nrOfSectors`, and `simulationTime`.

7.4.2.2 `CartesianToPolarConverter::~CartesianToPolarConverter ()`

Definition at line 24 of file `CartesianToPolarConverter.cpp`.

7.4.3 Member Function Documentation

7.4.3.1 `void CartesianToPolarConverter::convert (bool outputToScript)`

Start the conversion.

Parameters

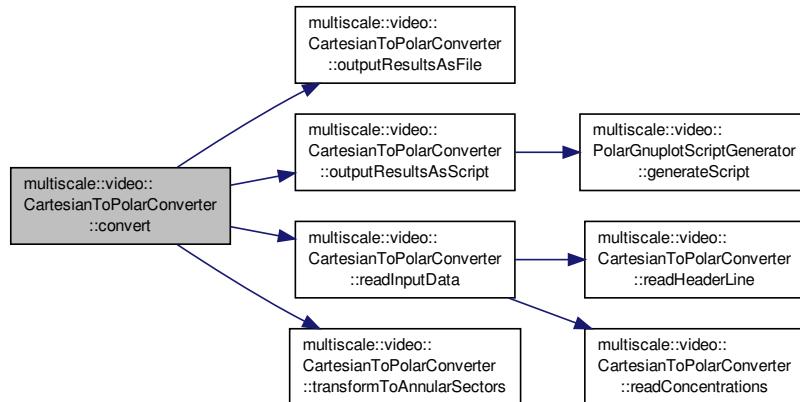
<code>outputToScript</code>	Output to script or to plain file
-----------------------------	-----------------------------------

Definition at line 26 of file `CartesianToPolarConverter.cpp`.

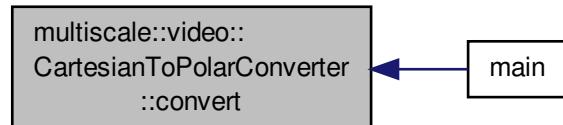
References `outputResultsAsFile()`, `outputResultsAsScript()`, `readInputData()`, and `transformToAnnularSectors()`.

Referenced by `main()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.2 void `CartesianToPolarConverter::outputResultsAsFile()` [private]

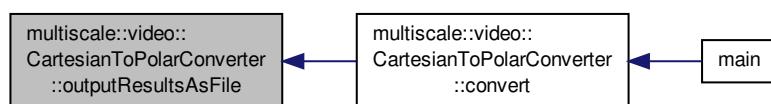
Output the results as a plain file.

Definition at line 115 of file `CartesianToPolarConverter.cpp`.

References `annularSectors`, `OUTPUT_FILE_EXTENSION`, and `outputFilepath`.

Referenced by `convert()`.

Here is the caller graph for this function:



7.4.3.3 void CartesianToPolarConverter::outputResultsAsScript() [private]

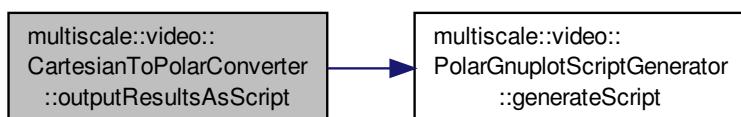
Output the results as a gnuplot script.

Definition at line 130 of file CartesianToPolarConverter.cpp.

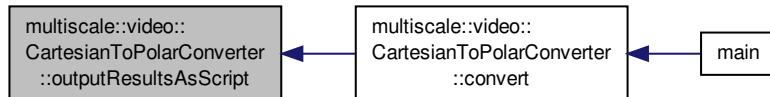
References annularSectors, multiscale::video::PolarGnuplotScriptGenerator::generateScript(), outputFilepath, and simulationTime.

Referenced by convert().

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.4 void CartesianToPolarConverter::readConcentrations(ifstream & fin) [private]

Read the concentrations.

Parameters

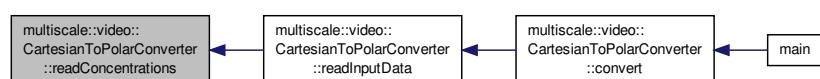
<i>fin</i>	Input file stream
------------	-------------------

Definition at line 70 of file CartesianToPolarConverter.cpp.

References concentrations, ERR_CONC, nrOfConcentricCircles, and nrOfSectors.

Referenced by readInputData().

Here is the caller graph for this function:



7.4.3.5 void `CartesianToPolarConverter::readHeaderLine(ifstream & fin)` [private]

Read the header line.

The header line contains values for number of concentric circles, number of sectors and simulation time

Parameters

<code>fin</code>	Input file stream
------------------	-------------------

Definition at line 61 of file `CartesianToPolarConverter.cpp`.

References `ERR_NEG_SIM_TIME`, `ERR_NONPOS_DIMENSION`, `nrOfConcentricCircles`, `nrOfSectors`, and `simulationTime`.

Referenced by `readInputData()`.

Here is the caller graph for this function:



7.4.3.6 void `CartesianToPolarConverter::readInputData()` [private]

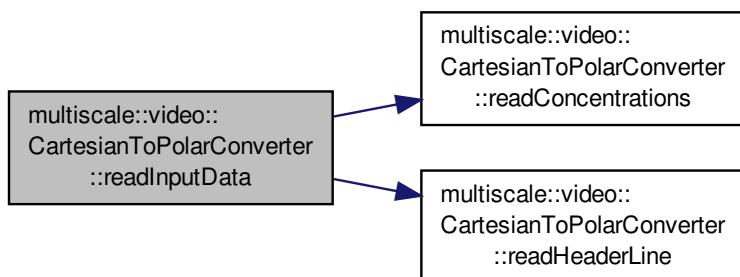
Read the input data.

Definition at line 37 of file `CartesianToPolarConverter.cpp`.

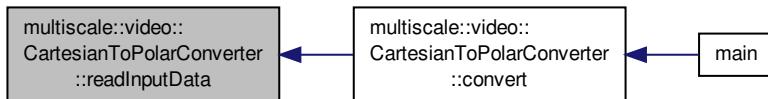
References `ERR_IN_EXTRA_DATA`, `ERR_INPUT_OPEN`, `inputFilepath`, `readConcentrations()`, and `readHeaderLine()`.

Referenced by `convert()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.7 void `CartesianToPolarConverter::transformToAnnularSectors()` [private]

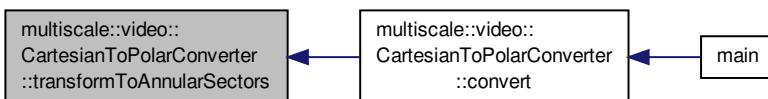
Convert the concentrations to annular sectors.

Definition at line 90 of file `CartesianToPolarConverter.cpp`.

References `annularSectors`, `concentrations`, `nrOfConcentricCircles`, `nrOfSectors`, `RADIUS_MAX`, and `RADIUS_MIN`.

Referenced by `convert()`.

Here is the caller graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 `vector<AnnularSector> multiscale::video::CartesianToPolarConverter::annularSectors` [private]

Resulting annular sectors

Definition at line 32 of file `CartesianToPolarConverter.hpp`.

Referenced by `outputResultsAsFile()`, `outputResultsAsScript()`, and `transformToAnnularSectors()`.

7.4.4.2 `vector<double> multiscale::video::CartesianToPolarConverter::concentrations` [private]

Concentrations received as input

Definition at line 33 of file `CartesianToPolarConverter.hpp`.

Referenced by `readConcentrations()`, and `transformToAnnularSectors()`.

7.4.4.3 `string multiscale::video::CartesianToPolarConverter::inputFilepath` [private]

Path to the input file

Definition at line 39 of file `CartesianToPolarConverter.hpp`.

Referenced by `readInputData()`.

7.4.4.4 unsigned long multiscale::video::CartesianToPolarConverter::nrOfConcentricCircles [private]

Number of concentric circles

Definition at line 35 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `readConcentrations()`, `readHeaderLine()`, and `transformToAnnularSectors()`.

7.4.4.5 unsigned long multiscale::video::CartesianToPolarConverter::nrOfSectors [private]

Number of sectors

Definition at line 36 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `readConcentrations()`, `readHeaderLine()`, and `transformToAnnularSectors()`.

7.4.4.6 string multiscale::video::CartesianToPolarConverter::outputFilepath [private]

Path to the output file

Definition at line 40 of file `CartesianToPolarConverter.hpp`.

Referenced by `outputResultsAsFile()`, and `outputResultsAsScript()`.

7.4.4.7 double multiscale::video::CartesianToPolarConverter::simulationTime [private]

Simulation time corresponding to the input data

Definition at line 37 of file `CartesianToPolarConverter.hpp`.

Referenced by `CartesianToPolarConverter()`, `outputResultsAsScript()`, and `readHeaderLine()`.

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

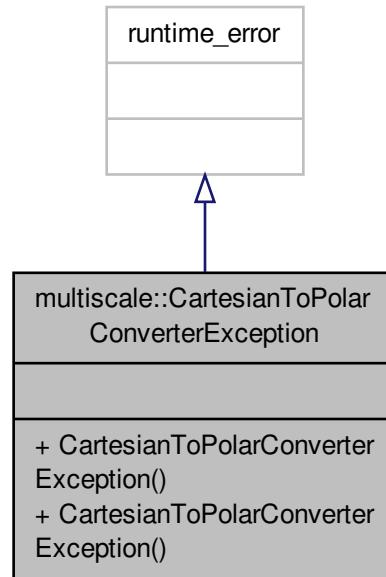
- include/multiscale/video/circular/[CartesianToPolarConverter.hpp](#)
- src/multiscale/video/circular/[CartesianToPolarConverter.cpp](#)

7.5 multiscale::CartesianToPolarConverterException Class Reference

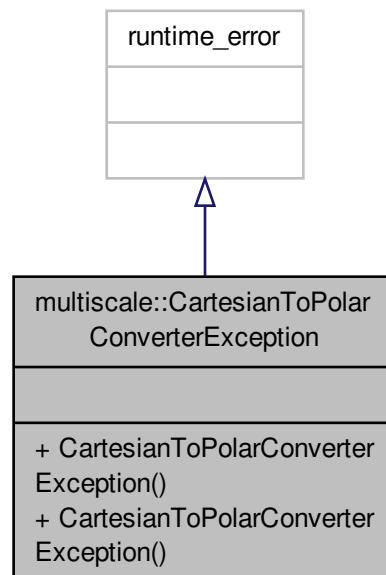
Exception class for the `CartesianToPolarConverter` class.

```
#include <CartesianToPolarConverterException.hpp>
```

Inheritance diagram for multiscale::CartesianToPolarConverterException:



Collaboration diagram for multiscale::CartesianToPolarConverterException:



Public Member Functions

- [CartesianToPolarConverterException](#) (const string &msg)
- [CartesianToPolarConverterException](#) (const char *msg)

7.5.1 Detailed Description

Exception class for the `CartesianToPolarConverter` class.

Definition at line 13 of file `CartesianToPolarConverterException.hpp`.

7.5.2 Constructor & Destructor Documentation

- 7.5.2.1 `multiscale::CartesianToPolarConverterException::CartesianToPolarConverterException (const string & msg)`
 [inline]

Definition at line 17 of file `CartesianToPolarConverterException.hpp`.

- 7.5.2.2 `multiscale::CartesianToPolarConverterException::CartesianToPolarConverterException (const char * msg)`
 [inline]

Definition at line 18 of file `CartesianToPolarConverterException.hpp`.

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

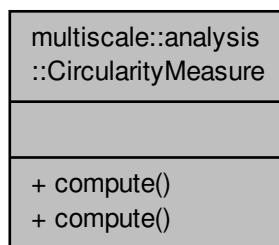
- [include/multiscale/exception/CartesianToPolarConverterException.hpp](#)

7.6 multiscale::analysis::CircularityMeasure Class Reference

Class for computing the circularity measure for the given collection of points.

```
#include <CircularityMeasure.hpp>
```

Collaboration diagram for `multiscale::analysis::CircularityMeasure`:



Static Public Member Functions

- static double [compute](#) (const vector< Point2f > &points)
Compute circularity measure for the given collection of points.

- static double `compute` (const vector< Point > &points)

Compute circularity measure for the given collection of points.

7.6.1 Detailed Description

Class for computing the circularity measure for the given collection of points.

Definition at line 18 of file CircularityMeasure.hpp.

7.6.2 Member Function Documentation

7.6.2.1 double CircularityMeasure::compute (const vector< Point2f > & points) [static]

Compute circularity measure for the given collection of points.

The circularity measure is equal to the standard circularity measure described in the following paper:

Joviša Žunić, Kaoru Hirota, Paul L. Rosin, A Hu moment invariant as a shape circularity measure, Pattern Recognition, Volume 43, Issue 1, January 2010, Pages 47-57, ISSN 0031-3203, <http://dx.doi.org/10.1016/j.patcog.2009.06.017>.

Definition at line 7 of file CircularityMeasure.cpp.

References PI.

7.6.2.2 double CircularityMeasure::compute (const vector< Point > & points) [static]

Compute circularity measure for the given collection of points.

The circularity measure is equal to the standard circularity measure described in the following paper:

Joviša Žunić, Kaoru Hirota, Paul L. Rosin, A Hu moment invariant as a shape circularity measure, Pattern Recognition, Volume 43, Issue 1, January 2010, Pages 47-57, ISSN 0031-3203, <http://dx.doi.org/10.1016/j.patcog.2009.06.017>.

Definition at line 23 of file CircularityMeasure.cpp.

References PI.

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

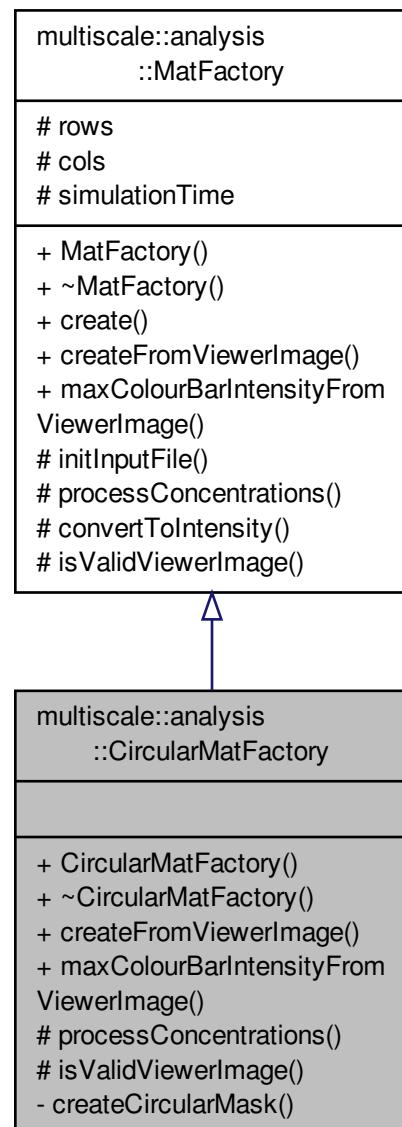
- include/multiscale/analysis/spatial/CircularityMeasure.hpp
- src/multiscale/analysis/spatial/CircularityMeasure.cpp

7.7 multiscale::analysis::CircularMatFactory Class Reference

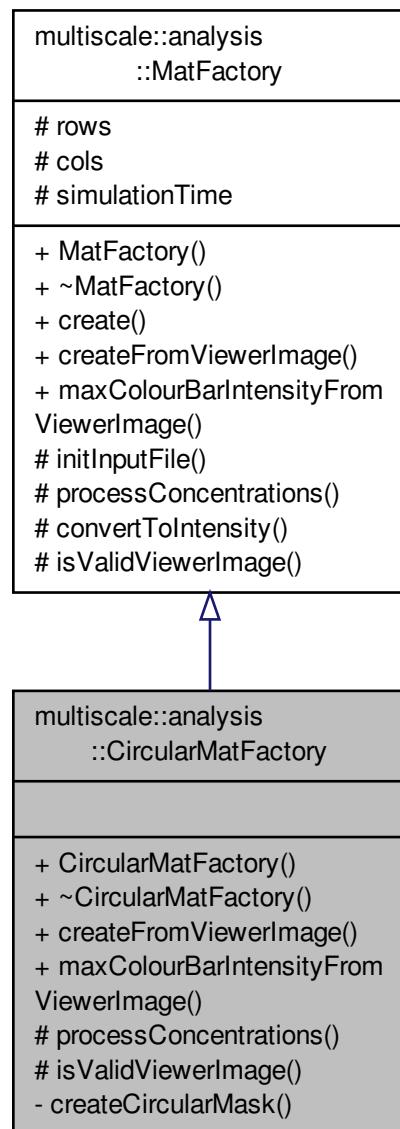
Class for creating a Mat object considering a circular grid.

```
#include <CircularMatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::CircularMatFactory:



Collaboration diagram for multiscale::analysis::CircularMatFactory:



Public Member Functions

- [CircularMatFactory \(\)](#)
- [~CircularMatFactory \(\)](#)
- Mat [createFromViewerImage](#) (const string &inputFile) override
Create a Mat object from the image file obtained from the CircularGeometryViewer.
- double [maxColourBarIntensityFromViewerImage](#) (const string &inputFile) override
Get the maximum grayscale intensity of the colour bar in the image.

Protected Member Functions

- `unsigned char * processConcentrations (ifstream &fin) override`
`Process the concentrations from the input file.`
- `bool isValidViewerImage (const Mat &image) override`
`Check if the image generated by the viewer has the required resolution.`

Private Member Functions

- `Mat createCircularMask (unsigned int originX, unsigned int originY, unsigned int radius, const Mat &image)`
`Create a mask with 255 intensity pixels inside the circle with origin at (originX, originY) and the given radius.`

Additional Inherited Members

7.7.1 Detailed Description

Class for creating a Mat object considering a circular grid.

Definition at line 29 of file CircularMatFactory.hpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 CircularMatFactory::CircularMatFactory ()

Definition at line 9 of file CircularMatFactory.cpp.

7.7.2.2 CircularMatFactory::~CircularMatFactory ()

Definition at line 11 of file CircularMatFactory.cpp.

7.7.3 Member Function Documentation

7.7.3.1 Mat CircularMatFactory::createCircularMask (unsigned int *originX*, unsigned int *originY*, unsigned int *radius*, const Mat & *image*) [private]

Create a mask with 255 intensity pixels inside the circle with origin at (originX, originY) and the given radius.

All the other pixels have intensity zero.

The original image is provided only for getting the size correctly

Parameters

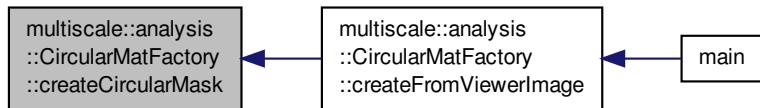
<code>originX</code>	The x coordinate for the origin
<code>originY</code>	The y coordinate for the origin
<code>radius</code>	The size of the radius
<code>image</code>	The original image

Definition at line 44 of file CircularMatFactory.cpp.

References INTENSITY_MAX.

Referenced by createFromViewerImage().

Here is the caller graph for this function:



7.7.3.2 Mat CircularMatFactory::createFromViewerImage (const string & *inputFile*) [override], [virtual]

Create a Mat object from the image file obtained from the CircularGeometryViewer.

Create the Mat instance from the given image file

Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

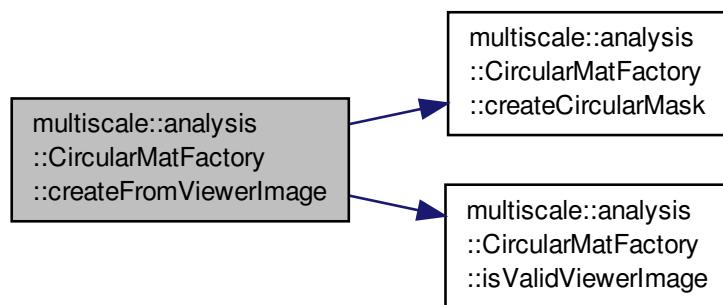
Implements [multiscale::analysis::MatFactory](#).

Definition at line 13 of file CircularMatFactory.cpp.

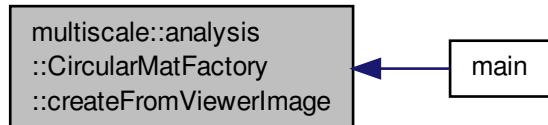
References `createCircularMask()`, `isValidViewerImage()`, `ROI_RADIUS`, `ROI_START_X`, and `ROI_START_Y`.

Referenced by `main()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.7.3.3 bool CircularMatFactory::isValidViewerImage (const Mat & *image*) [override], [protected], [virtual]

Check if the image generated by the viewer has the required resolution.

Parameters

<i>image</i>	Image generated by the viewer
--------------	-------------------------------

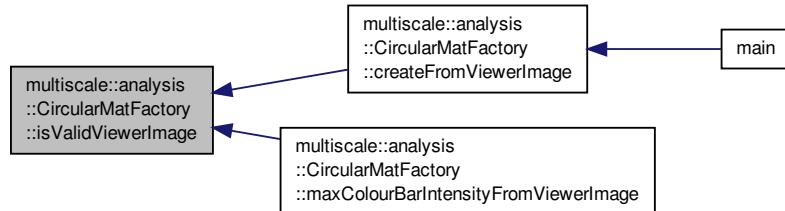
Implements [multiscale::analysis::MatFactory](#).

Definition at line 53 of file CircularMatFactory.cpp.

References ERR_IMG_RESOLUTION, ERR_INPUT_OPEN, INPUT_IMG_HEIGHT, and INPUT_IMG_WIDTH.

Referenced by `createFromViewerImage()`, and `maxColourBarIntensityFromViewerImage()`.

Here is the caller graph for this function:



7.7.3.4 double CircularMatFactory::maxColourBarIntensityFromViewerImage (const string & *inputFile*) [override], [virtual]

Get the maximum grayscale intensity of the colour bar in the image.

Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

Implements [multiscale::analysis::MatFactory](#).

Definition at line 32 of file CircularMatFactory.cpp.

References COLOURBAR_MAX_X, COLOURBAR_MAX_Y, and isValidViewerImage().

Here is the call graph for this function:



7.7.3.5 `unsigned char * CircularMatFactory::processConcentrations (ifstream & fin)` [override], [protected], [virtual]

Process the concentrations from the input file.

REMARK: This method is not implemented and throws an error when called.

Parameters

<code>fin</code>	Input file stream from which the concentrations are read
------------------	--

Implements [multiscale::analysis::MatFactory](#).

Definition at line 40 of file CircularMatFactory.cpp.

References `ERR_UNIMPLEMENTED_METHOD`.

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

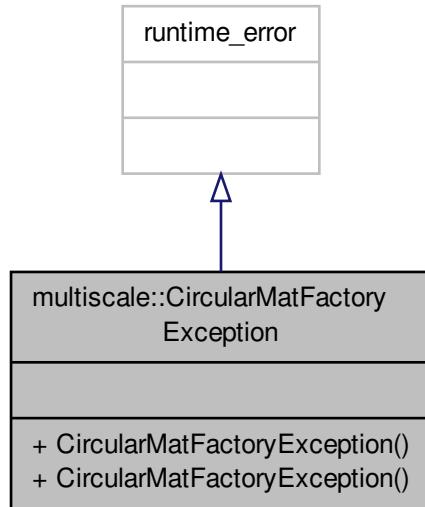
- include/multiscale/analysis/spatial/factory/[CircularMatFactory.hpp](#)
- src/multiscale/analysis/spatial/factory/[CircularMatFactory.cpp](#)

7.8 multiscale::CircularMatFactoryException Class Reference

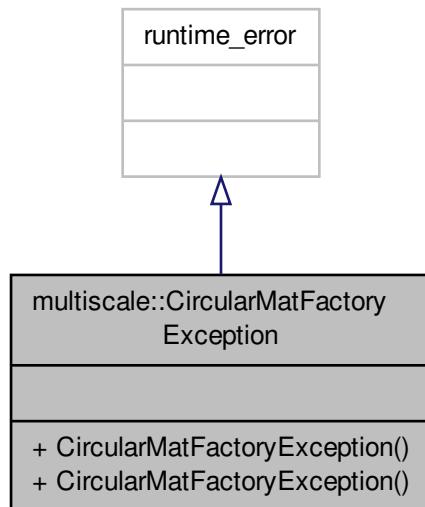
Exception class for the CircularMatFactory instances.

```
#include <CircularMatFactoryException.hpp>
```

Inheritance diagram for multiscale::CircularMatFactoryException:



Collaboration diagram for multiscale::CircularMatFactoryException:



Public Member Functions

- `CircularMatFactoryException` (const string &msg)
- `CircularMatFactoryException` (const char *msg)

7.8.1 Detailed Description

Exception class for the CircularMatFactory instances.

Definition at line 13 of file CircularMatFactoryException.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 `multiscale::CircularMatFactoryException::CircularMatFactoryException (const string & msg) [inline]`

Definition at line 17 of file CircularMatFactoryException.hpp.

7.8.2.2 `multiscale::CircularMatFactoryException::CircularMatFactoryException (const char * msg) [inline]`

Definition at line 18 of file CircularMatFactoryException.hpp.

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

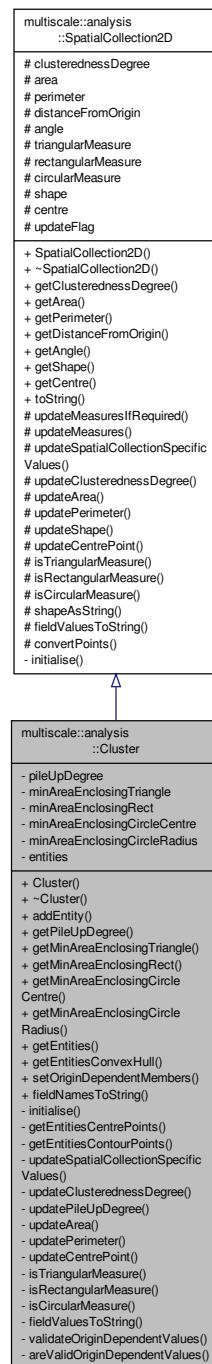
- [include/multiscale/exception/CircularMatFactoryException.hpp](#)

7.9 multiscale::analysis::Cluster Class Reference

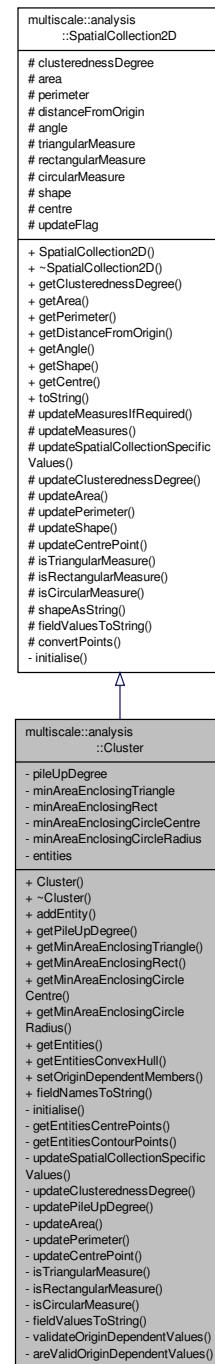
Class for representing a cluster of entities in an image.

```
#include <Cluster.hpp>
```

Inheritance diagram for multiscale::analysis::Cluster:



Collaboration diagram for multiscale::analysis::Cluster:



Public Member Functions

- `Cluster ()`
- `~Cluster ()`
- void `addEntity (const Entity &entity)`
 - Add a new entity to the cluster.*
- double `getPileUpDegree ()`

- `vector< Point2f > getMinAreaEnclosingTriangle ()`
Get the minimum area enclosing triangle.
- `RotatedRect getMinAreaEnclosingRect ()`
Get the minimum area enclosing rectangle.
- `Point2f getMinAreaEnclosingCircleCentre ()`
Get the minimum area enclosing circle centre.
- `float getMinAreaEnclosingCircleRadius ()`
Get the minimum area enclosing circle radius.
- `vector< Entity > getEntities () const`
Get the collection of underlying entities.
- `vector< Point2f > getEntitiesConvexHull ()`
Get the convex hull enclosing the collection of entities' contour points.
- `void setOriginDependentMembers (double distanceFromOrigin, double angleWrtOrigin)`
Set the values of the origin dependent members.

Static Public Member Functions

- `static string fieldNamesToString ()`
Get a string representation of all the field names printed in the "toString" method.

Private Member Functions

- `void initialise ()`
Initialisation function for the class.
- `vector< Point2f > getEntitiesCentrePoints ()`
Get the collection of entities' centres.
- `vector< Point2f > getEntitiesContourPoints ()`
Get the collection of entities' contour points.
- `void updateSpatialCollectionSpecificValues () override`
Update the values of all measures.
- `void updateClusterednessDegree () override`
Update the value of the clusteredness degree.
- `void updatePileUpDegree ()`
Update the value of the pile up degree.
- `void updateArea () override`
Update the value of the area.
- `void updatePerimeter () override`
Update the value of the perimeter.
- `void updateCentrePoint () override`
Update the point defining the centre of the cluster.
- `double isTriangularMeasure () override`
Get the measure that the cluster has a triangular shape.
- `double isRectangularMeasure () override`
Get the measure that the cluster has a rectangular shape.
- `double isCircularMeasure () override`
Get the measure that the cluster has a circular shape.
- `string fieldValuesToString () override`
Get a string representation of all the field values.
- `void validateOriginDependentValues (double distanceFromOrigin, double angleWrtOrigin)`
Validate the origin dependent values (i.e. non-negative)
- `bool areValidOriginDependentValues (double distanceFromOrigin, double angleWrtOrigin)`
Check if the origin dependent values are valid (i.e. non-negative)

Private Attributes

- double `pileUpDegree`
- vector< `Point2f` > `minAreaEnclosingTriangle`
- RotatedRect `minAreaEnclosingRect`
- `Point2f minAreaEnclosingCircleCentre`
- float `minAreaEnclosingCircleRadius`
- vector< `Entity` > `entities`

Additional Inherited Members

7.9.1 Detailed Description

Class for representing a cluster of entities in an image.

Definition at line 23 of file Cluster.hpp.

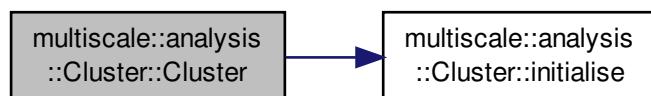
7.9.2 Constructor & Destructor Documentation

7.9.2.1 Cluster::Cluster ()

Definition at line 11 of file Cluster.cpp.

References initialise().

Here is the call graph for this function:



7.9.2.2 Cluster::~Cluster ()

Definition at line 15 of file Cluster.cpp.

7.9.3 Member Function Documentation

7.9.3.1 void Cluster::addEntity (const Entity & entity)

Add a new entity to the cluster.

Definition at line 17 of file Cluster.cpp.

References entities, and multiscale::analysis::SpatialCollection2D::updateFlag.

7.9.3.2 bool Cluster::isValidOriginDependentValues (double *distanceFromOrigin*, double *angleWrtOrigin*) [private]

Check if the origin dependent values are valid (i.e. non-negative)

Parameters

<i>distanceFromOrigin</i>	Distance from the origin
<i>angleWrtOrigin</i>	Angle with respect to the origin

Definition at line 229 of file Cluster.cpp.

Referenced by validateOriginDependentValues().

Here is the caller graph for this function:



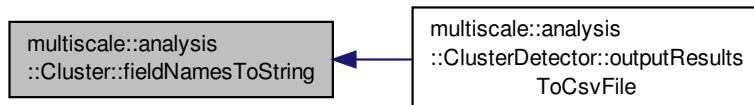
7.9.3.3 string Cluster::fieldNamesToString () [static]

Get a string representation of all the field names printed in the "toString" method.

Definition at line 75 of file Cluster.cpp.

Referenced by multiscale::analysis::ClusterDetector::outputResultsToCsvFile().

Here is the caller graph for this function:



7.9.3.4 string Cluster::fieldValuesToString () [override], [private], [virtual]

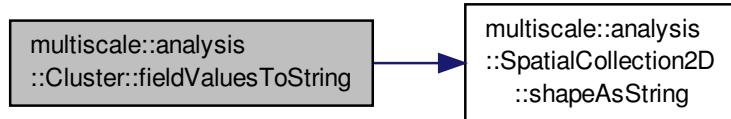
Get a string representation of all the field values.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 208 of file Cluster.cpp.

References multiscale::analysis::SpatialCollection2D::angle, multiscale::analysis::SpatialCollection2D::area, multiscale::analysis::SpatialCollection2D::centre, multiscale::analysis::SpatialCollection2D::circularMeasure, multiscale::analysis::SpatialCollection2D::clusterednessDegree, multiscale::analysis::SpatialCollection2D::distanceFromOrigin, OUTPUT_SEPARATOR, multiscale::analysis::SpatialCollection2D::perimeter, pileUpDegree, multiscale::analysis::SpatialCollection2D::rectangularMeasure, multiscale::analysis::SpatialCollection2D::shapeAsString(), and multiscale::analysis::SpatialCollection2D::triangularMeasure.

Here is the call graph for this function:



7.9.3.5 `vector< Entity > Cluster::getEntities() const`

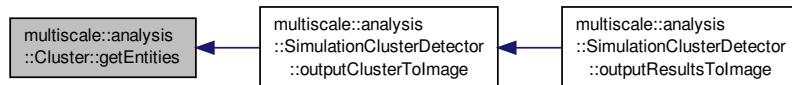
Get the collection of underlying entities.

Definition at line 53 of file Cluster.cpp.

References entities.

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterToImage().

Here is the caller graph for this function:



7.9.3.6 `vector< Point2f > Cluster::getEntitiesCentrePoints() [private]`

Get the collection of entities' centres.

Definition at line 94 of file Cluster.cpp.

References entities.

7.9.3.7 `vector< Point2f > Cluster::getEntitiesContourPoints() [private]`

Get the collection of entities' contour points.

Definition at line 104 of file Cluster.cpp.

References entities.

Referenced by getEntitiesConvexHull(), isCircularMeasure(), and isRectangularMeasure().

Here is the caller graph for this function:



7.9.3.8 vector< Point2f > Cluster::getEntitiesConvexHull()

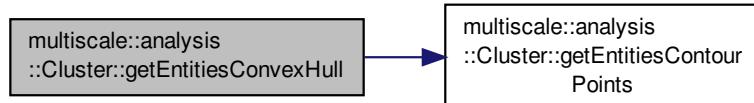
Get the convex hull enclosing the collection of entities' contour points.

Definition at line 57 of file Cluster.cpp.

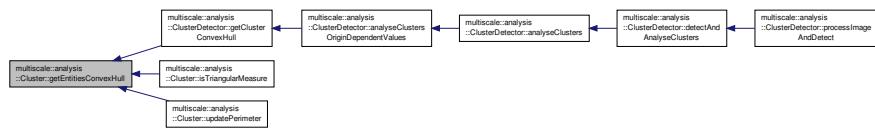
References CONVEX_HULL_CLOCKWISE, entities, and getEntitiesContourPoints().

Referenced by multiscale::analysis::ClusterDetector::getClusterConvexHull(), isTriangularMeasure(), and updatePerimeter().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.9 Point2f Cluster::getMinAreaEnclosingCircleCentre ()

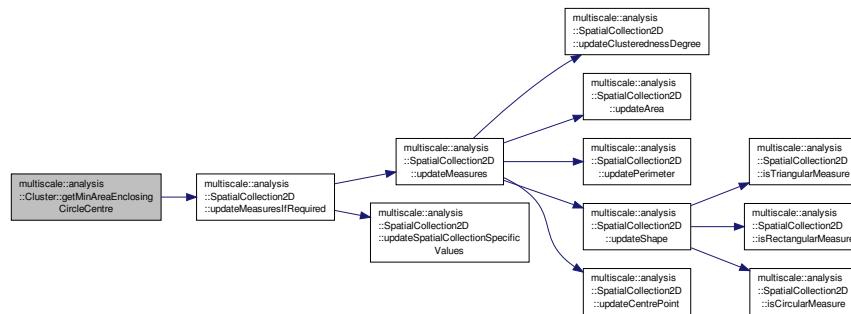
Get the minimum area enclosing circle centre.

Definition at line 41 of file Cluster.cpp.

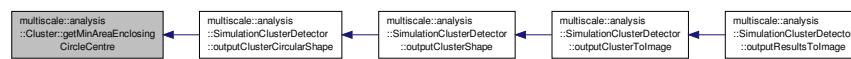
References minAreaEnclosingCircleCentre, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterCircularShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.10 float Cluster::getMinAreaEnclosingCircleRadius ()

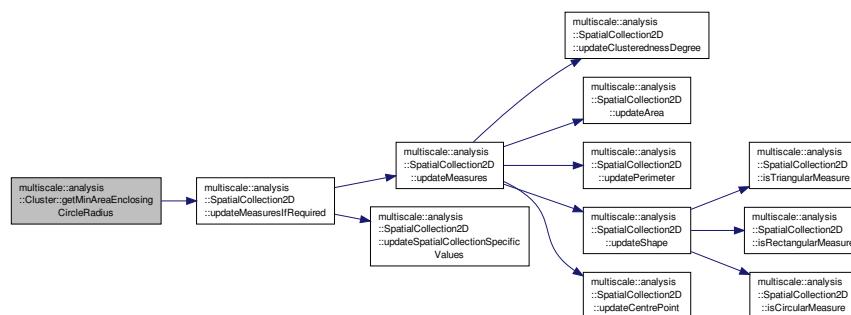
Get the minimum area enclosing circle radius.

Definition at line 47 of file Cluster.cpp.

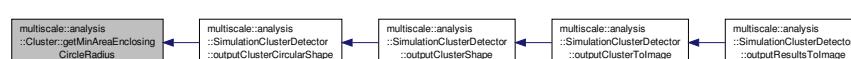
References minAreaEnclosingCircleRadius, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterCircularShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.11 `RotatedRect Cluster::getMinAreaEnclosingRect()`

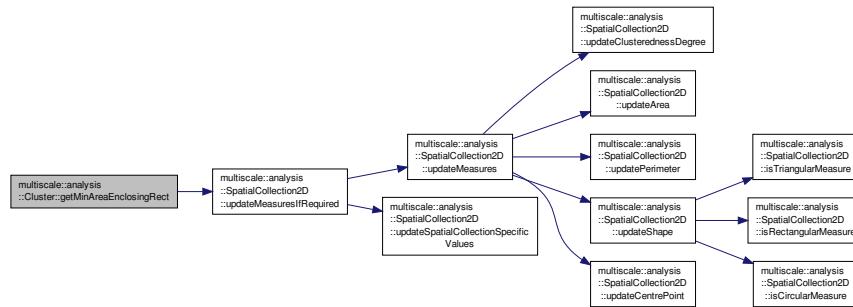
Get the minimum area enclosing rectangle.

Definition at line 35 of file Cluster.cpp.

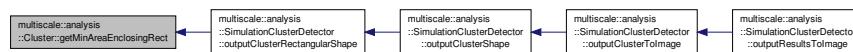
References minAreaEnclosingRect, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterRectangularShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.12 `vector< Point2f > Cluster::getMinAreaEnclosingTriangle()`

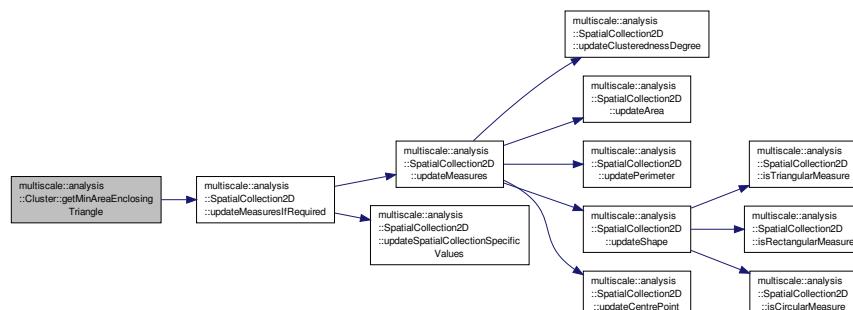
Get the minimum area enclosing triangle.

Definition at line 29 of file Cluster.cpp.

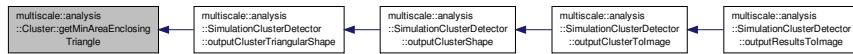
References minAreaEnclosingTriangle, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterTriangularShape().

Here is the call graph for this function:



Here is the caller graph for this function:



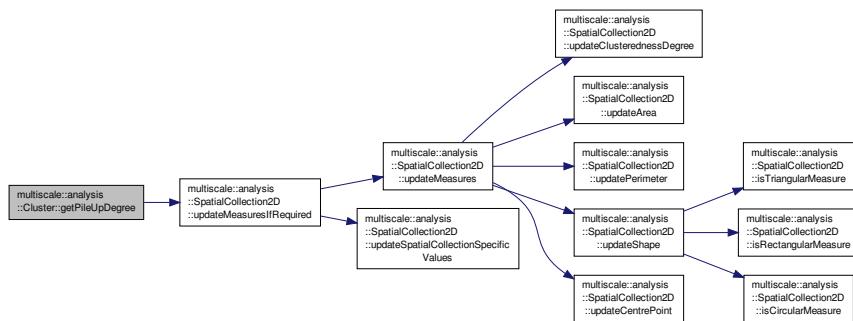
7.9.3.13 double Cluster::getPileUpDegree ()

Get the degree of pile up.

Definition at line 23 of file Cluster.cpp.

References pileUpDegree, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Here is the call graph for this function:



7.9.3.14 void Cluster::initialise () [private]

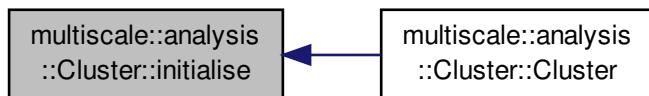
Initialisation function for the class.

Definition at line 79 of file Cluster.cpp.

References multiscale::analysis::SpatialCollection2D::angle, multiscale::analysis::SpatialCollection2D::clusterednessDegree, multiscale::analysis::SpatialCollection2D::distanceFromOrigin, entities, minAreaEnclosingCircleRadius, minAreaEnclosingTriangle, pileUpDegree, and multiscale::analysis::SpatialCollection2D::updateFlag.

Referenced by Cluster().

Here is the caller graph for this function:



7.9.3.15 double Cluster::isCircularMeasure() [override], [private], [virtual]

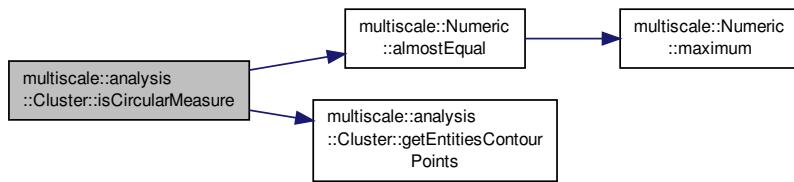
Get the measure that the cluster has a circular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 196 of file Cluster.cpp.

References `multiscale::Numeric::almostEqual()`, `multiscale::analysis::SpatialCollection2D::area`, `getEntitiesContourPoints()`, `minAreaEnclosingCircleCentre`, `minAreaEnclosingCircleRadius`, and `PI`.

Here is the call graph for this function:



7.9.3.16 double Cluster::isRectangularMeasure() [override], [private], [virtual]

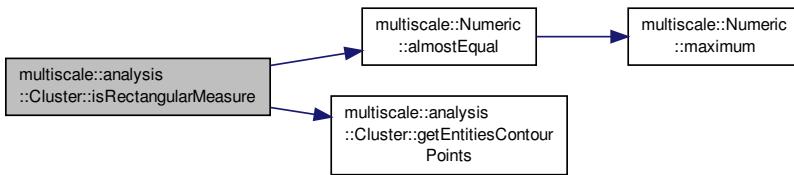
Get the measure that the cluster has a rectangular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 184 of file Cluster.cpp.

References `multiscale::Numeric::almostEqual()`, `multiscale::analysis::SpatialCollection2D::area`, `getEntitiesContourPoints()`, and `minAreaEnclosingRect`.

Here is the call graph for this function:



7.9.3.17 double Cluster::isTriangularMeasure() [override], [private], [virtual]

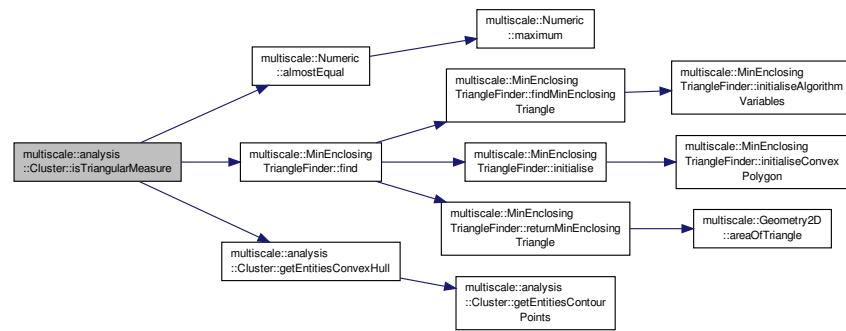
Get the measure that the cluster has a triangular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 175 of file Cluster.cpp.

References `multiscale::Numeric::almostEqual()`, `multiscale::analysis::SpatialCollection2D::area`, `multiscale::MinEnclosingTriangleFinder::find()`, `getEntitiesConvexHull()`, and `minAreaEnclosingTriangle`.

Here is the call graph for this function:



7.9.3.18 void Cluster::setOriginDependentMembers (double *distanceFromOrigin*, double *angleWrtOrigin*)

Set the values of the origin dependent members.

Parameters

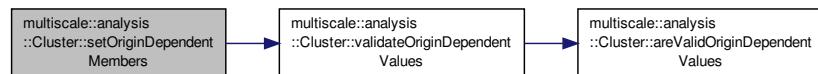
<i>distanceFromOrigin</i>	Distance from the origin
<i>angleWrtOrigin</i>	Angle with respect to the origin

Definition at line 68 of file Cluster.cpp.

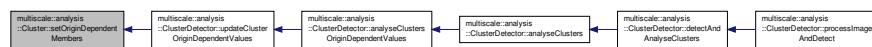
References multiscale::analysis::SpatialCollection2D::angle, multiscale::analysis::SpatialCollection2D::distanceFromOrigin, and validateOriginDependentValues().

Referenced by multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.19 void Cluster::updateArea () [override], [private], [virtual]

Update the value of the area.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 150 of file Cluster.cpp.

References multiscale::analysis::SpatialCollection2D::area, and entities.

7.9.3.20 void Cluster::updateCentrePoint() [override], [private], [virtual]

Update the point defining the centre of the cluster.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 164 of file Cluster.cpp.

References multiscale::analysis::SpatialCollection2D::centre, and entities.

7.9.3.21 void Cluster::updateClusterednessDegree() [override], [private], [virtual]

Update the value of the clusteredness degree.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 120 of file Cluster.cpp.

References multiscale::analysis::SpatialCollection2D::clusterednessDegree, and entities.

7.9.3.22 void Cluster::updatePerimeter() [override], [private], [virtual]

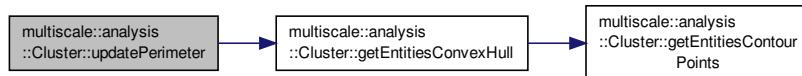
Update the value of the perimeter.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 158 of file Cluster.cpp.

References getEntitiesConvexHull(), and multiscale::analysis::SpatialCollection2D::perimeter.

Here is the call graph for this function:



7.9.3.23 void Cluster::updatePileUpDegree() [private]

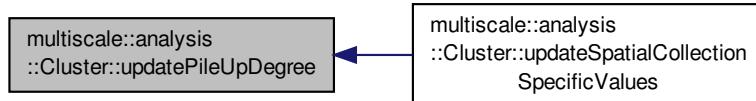
Update the value of the pile up degree.

Definition at line 140 of file Cluster.cpp.

References entities, and pileUpDegree.

Referenced by updateSpatialCollectionSpecificValues().

Here is the caller graph for this function:



7.9.3.24 void Cluster::updateSpatialCollectionSpecificValues () [override], [private], [virtual]

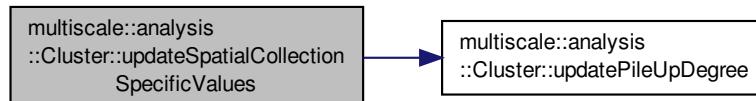
Update the values of all measures.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 116 of file Cluster.cpp.

References updatePileUpDegree().

Here is the call graph for this function:



7.9.3.25 void Cluster::validateOriginDependentValues (double distanceFromOrigin, double angleWrtOrigin) [private]

Validate the origin dependent values (i.e. non-negative)

Parameters

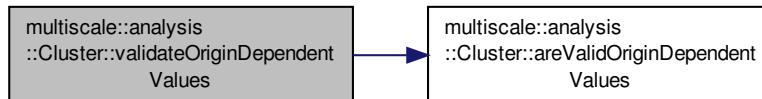
<i>distanceFrom- Origin</i>	Distance from the origin
<i>angleWrtOrigin</i>	Angle with respect to the origin

Definition at line 223 of file Cluster.cpp.

References areValidOriginDependentValues(), and ERR_ORIGIN_DEPENDENT_VALUES.

Referenced by setOriginDependentMembers().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.4 Member Data Documentation

7.9.4.1 `vector<Entity> multiscale::analysis::Cluster::entities` [private]

Entities which belong to this cluster

Definition at line 36 of file Cluster.hpp.

Referenced by addEntity(), getEntities(), getEntitiesCentrePoints(), getEntitiesContourPoints(), getEntitiesConvexHull(), initialise(), updateArea(), updateCentrePoint(), updateClusterednessDegree(), and updatePileUpDegree().

7.9.4.2 `Point2f multiscale::analysis::Cluster::minAreaEnclosingCircleCentre` [private]

The minimum area enclosing circle centre point

Definition at line 33 of file Cluster.hpp.

Referenced by getMinAreaEnclosingCircleCentre(), and isCircularMeasure().

7.9.4.3 `float multiscale::analysis::Cluster::minAreaEnclosingCircleRadius` [private]

The minimum area enclosing circle radius

Definition at line 34 of file Cluster.hpp.

Referenced by getMinAreaEnclosingCircleRadius(), initialise(), and isCircularMeasure().

7.9.4.4 `RotatedRect multiscale::analysis::Cluster::minAreaEnclosingRect` [private]

The minimum area enclosing rectangle

Definition at line 31 of file Cluster.hpp.

Referenced by getMinAreaEnclosingRect(), and isRectangularMeasure().

7.9.4.5 `vector<Point2f> multiscale::analysis::Cluster::minAreaEnclosingTriangle` [private]

The minimum area enclosing triangle

Definition at line 29 of file Cluster.hpp.

Referenced by getMinAreaEnclosingTriangle(), initialise(), and isTriangularMeasure().

7.9.4.6 double multiscale::analysis::Cluster::pileUpDegree [private]

Degree of pile up

Definition at line 27 of file Cluster.hpp.

Referenced by fieldValuesToString(), getPileUpDegree(), initialise(), and updatePileUpDegree().

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

- include/multiscale/analysis/spatial/[Cluster.hpp](#)

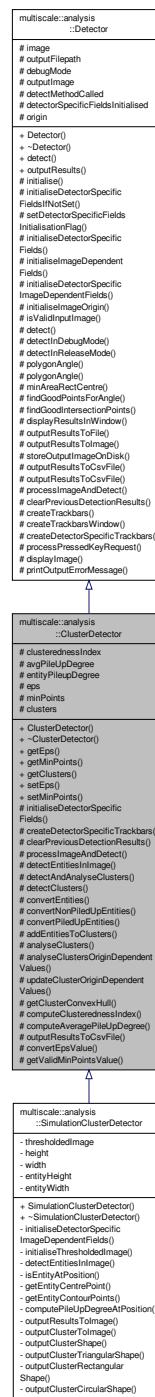
- src/multiscale/analysis/spatial/[Cluster.cpp](#)

7.10 multiscale::analysis::ClusterDetector Class Reference

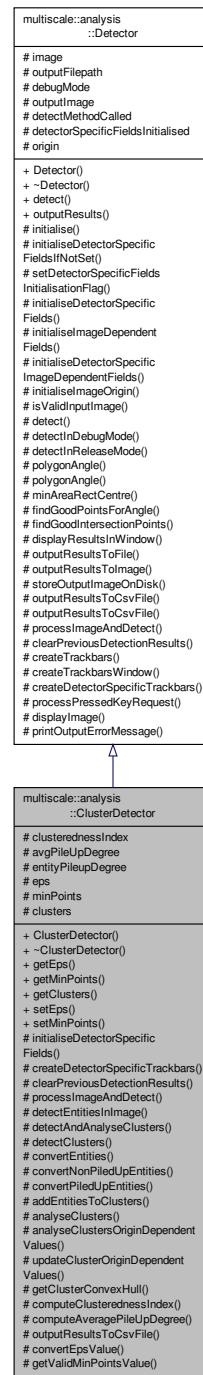
Class for detecting clusters in 2D images.

```
#include <ClusterDetector.hpp>
```

Inheritance diagram for multiscale::analysis::ClusterDetector:



Collaboration diagram for multiscale::analysis::ClusterDetector:



Public Member Functions

- `ClusterDetector (int maxPileupNumber, double maxPileupIntensity, bool debugMode=false)`
- virtual `~ClusterDetector ()`
- double `getEps ()`

Get the value of the clustering algorithm parameter eps.

- int `getMinPoints ()`

- `vector< Cluster > const & getClusters ()`
Get the value of the clustering algorithm parameter MinPoints.
- `void setEps (double eps)`
Set the value of the clustering algorithm parameter eps.
- `void setMinPoints (int minPoints)`
Set the value of the clustering algorithm parameter MinPoints.

Protected Member Functions

- `void initialiseDetectorSpecificFields () override`
Initialise clustering values.
- `void createDetectorSpecificTrackbars () override`
Create the trackbars.
- `void clearPreviousDetectionResults () override`
Clear the clusters from the previous detection.
- `void processImageAndDetect () override`
Process the provided image and detect clusters in it.
- `virtual void detectEntitiesInImage (vector< Entity > &entities)=0`
Detect the entities in the image.
- `void detectAndAnalyseClusters (const vector< Entity > &entities, vector< Cluster > &clusters)`
Detect and analyse the clusters of entities in the image.
- `void detectClusters (const vector< Entity > &entities, vector< int > &clusterIndexes, int &nrofClusters)`
Detect the clusters of entities in the image.
- `vector< shared_ptr< DataPoint > > convertEntities (const vector< Entity > &entities)`
Convert the entities to the format required by the DBSCAN class.
- `void convertNonPiledUpEntities (const vector< Entity > &entities, vector< shared_ptr< DataPoint > > &dataPoints)`
Convert the non pile up entities to the format required by the DBSCAN class.
- `void convertPiledUpEntities (const vector< Entity > &entities, vector< shared_ptr< DataPoint > > &dataPoints)`
Convert the entities to the required format by the DBSCAN class.
- `void addEntitiesToClusters (const vector< Entity > &entities, const vector< int > &clusterIndexes, int nrofClusters, vector< Cluster > &clusters)`
Add the entities to the clusters as indicated by the clusterIndexes parameter.
- `void analyseClusters (vector< Cluster > &clusters)`
Analyse the clusters.
- `void analyseClustersOriginDependentValues (vector< Cluster > &clusters)`
Analyse the clusters and compute the origin dependent values.
- `void updateClusterOriginDependentValues (Cluster &cluster, const vector< Point > &clusterConvexHull)`
Update the cluster and compute the origin dependent values considering the convex hull.
- `vector< Point > getClusterConvexHull (Cluster &cluster)`
Return the convex hull of the given cluster.
- `double computeClusterednessIndex (const vector< Cluster > &clusters)`
Compute the clusteredness index for all the entities detected in the image.
- `double computeAveragePileUpDegree (vector< Cluster > &clusters)`
Compute the average pile up degree for all entities in the image.
- `void outputResultsToCsvFile (ofstream &fout) override`
Output the information computed for the clusters to a csv file.
- `double convertEpsValue ()`
Convert the value of eps from integer to double.
- `int getValidMinPointsValue ()`
Return non-zero value for minPoints.

Protected Attributes

- double clusterednessIndex
- double avgPileUpDegree
- double entityPileupDegree
- int `eps`
- int `minPoints`
- vector< `Cluster` > `clusters`

7.10.1 Detailed Description

Class for detecting clusters in 2D images.

Definition at line 34 of file ClusterDetector.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 ClusterDetector::ClusterDetector (int `maxPileupNumber`, double `maxPileupIntensity`, bool `debugMode = false`)

Parameters

<code>debugMode</code>	Flag indicating if detector should run in debug mode or not
<code>maxPileup-Number</code>	The maximum number of entities which can occupy a grid position at the same time
<code>maxPileup-Intensity</code>	The grayscale intensity of a maximally piled up grid position

Definition at line 15 of file ClusterDetector.cpp.

References `avgPileUpDegree`, `clusterednessIndex`, `entityPileupDegree`, `eps`, and `minPoints`.

7.10.2.2 ClusterDetector::~ClusterDetector () [virtual]

Definition at line 25 of file ClusterDetector.cpp.

7.10.3 Member Function Documentation

7.10.3.1 void ClusterDetector::addEntitiesToClusters (const vector< Entity > & `entities`, const vector< int > & `clusterIndexes`, int `nrOfClusters`, vector< Cluster > & `clusters`) [protected]

Add the entities to the clusters as indicated by the `clusterIndexes` parameter.

Add the entities to the clusters as indicated by the `clusterIndexes` parameter

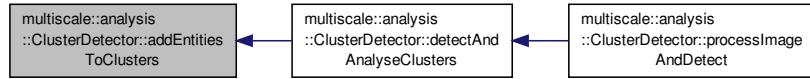
Parameters

<code>entities</code>	Entities detected in the image
<code>clusterIndexes</code>	Indexes to which cluster each entity belongs
<code>nrOfClusters</code>	Total number of clusters
<code>clusters</code>	Collection of clusters, each one with the updated measures

Definition at line 111 of file ClusterDetector.cpp.

Referenced by `detectAndAnalyseClusters()`.

Here is the caller graph for this function:



7.10.3.2 void ClusterDetector::analyseClusters (vector< Cluster > & clusters) [protected]

Analyse the clusters.

Analyse the clusters and compute the angle and distance from the centre, average clusteredness degree and pile up degree

Parameters

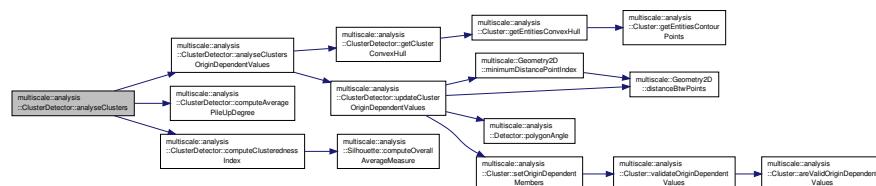
<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 122 of file ClusterDetector.cpp.

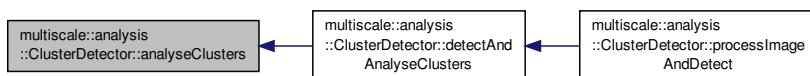
References analyseClustersOriginDependentValues(), avgPileUpDegree, clusterednessIndex, computeAveragePileUpDegree(), and computeClusterednessIndex().

Referenced by detectAndAnalyseClusters().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.3 void ClusterDetector::analyseClustersOriginDependentValues (vector< Cluster > & clusters) [protected]

Analyse the clusters and compute the origin dependent values.

The values which depend on the origin point are the distance of the cluster from the centre and the angle

Parameters

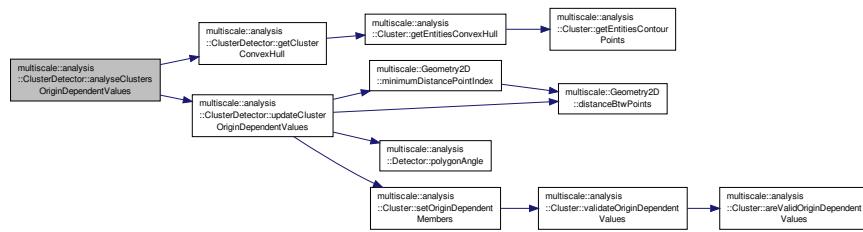
<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 129 of file ClusterDetector.cpp.

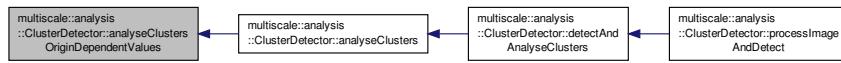
References getClusterConvexHull(), and updateClusterOriginDependentValues().

Referenced by analyseClusters().

Here is the call graph for this function:



Here is the caller graph for this function:

**7.10.3.4 void ClusterDetector::clearPreviousDetectionResults() [override], [protected], [virtual]**

Clear the clusters from the previous detection.

Implements [multiscale::analysis::Detector](#).

Definition at line 61 of file ClusterDetector.cpp.

References clusters.

7.10.3.5 double ClusterDetector::computeAveragePileUpDegree(vector< Cluster > & clusters) [protected]

Compute the average pile up degree for all entities in the image.

Compute the average pile up degree for all entities in the image as the sum of the average pile up degrees of all clusters divided by the number of clusters

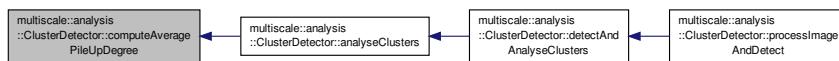
Parameters

<i>clusters</i>	Clusters of entities detected in the image
-----------------	--

Definition at line 162 of file ClusterDetector.cpp.

Referenced by analyseClusters().

Here is the caller graph for this function:



7.10.3.6 double ClusterDetector::computeClusterednessIndex (const vector< Cluster > & clusters) [protected]

Compute the clusteredness index for all the entities detected in the image.

Compute the clusteredness index for all the entities detected in the image using [Silhouette](#) cluster validity index

Parameters

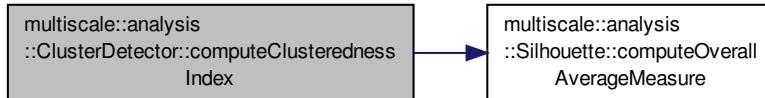
<i>clusters</i>	Collection of clusters, each one with the updated measures
-----------------	--

Definition at line 158 of file ClusterDetector.cpp.

References multiscale::analysis::Silhouette::computeOverallAverageMeasure().

Referenced by analyseClusters().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.7 vector< shared_ptr< DataPoint > > ClusterDetector::convertEntities (const vector< Entity > & entities) [protected]

Convert the entities to the format required by the [DBSCAN](#) class.

Parameters

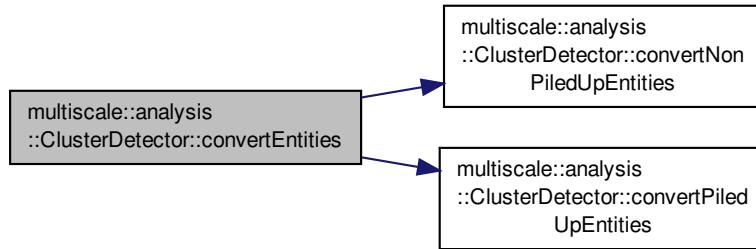
<i>entities</i>	Entities detected in the image
-----------------	--------------------------------

Definition at line 85 of file ClusterDetector.cpp.

References convertNonPiledUpEntities(), and convertPiledUpEntities().

Referenced by detectClusters().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.8 double ClusterDetector::convertEpsValue() [protected]

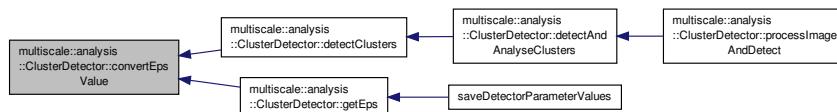
Convert the value of eps from integer to double.

Definition at line 191 of file ClusterDetector.cpp.

References eps, EPS_MAX, EPS_MIN, EPS_REAL_MAX, and EPS_REAL_MIN.

Referenced by detectClusters(), and getEps().

Here is the caller graph for this function:



7.10.3.9 void ClusterDetector::convertNonPiledUpEntities (const vector< Entity > & entities, vector< shared_ptr< DataPoint > > & dataPoints) [protected]

Convert the non pile up entities to the format required by the [DBSCAN](#) class.

Parameters

<i>entities</i>	Entities detected in the image
<i>dataPoints</i>	Collection of DataPoint instances required by the DBSCAN class

Definition at line 94 of file ClusterDetector.cpp.

Referenced by convertEntities().

Here is the caller graph for this function:



7.10.3.10 void ClusterDetector::convertPiledUpEntities (const vector< Entity > & entities, vector< shared_ptr< DataPoint > > & dataPoints) [protected]

Convert the entities to the required format by the [DBSCAN](#) class.

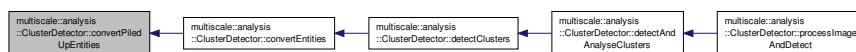
Parameters

<i>entities</i>	Entities detected in the image
<i>dataPoints</i>	Collection of DataPoint instances required by the DBSCAN class

Definition at line 100 of file ClusterDetector.cpp.

Referenced by convertEntities().

Here is the caller graph for this function:



7.10.3.11 void ClusterDetector::createDetectorSpecificTrackbars () [override], [protected], [virtual]

Create the trackbars.

Implements [multiscale::analysis::Detector](#).

Definition at line 56 of file ClusterDetector.cpp.

References eps, EPS_MAX, MIN_POINTS_MAX, minPoints, TRACKBAR_EPS, TRACKBAR_MINPOINTS, and W_IN_OUTPUT_IMAGE.

7.10.3.12 void ClusterDetector::detectAndAnalyseClusters (const vector< Entity > & entities, vector< Cluster > & clusters) [protected]

Detect and analyse the clusters of entities in the image.

Detect and analyse the clusters of entities in the image

Parameters

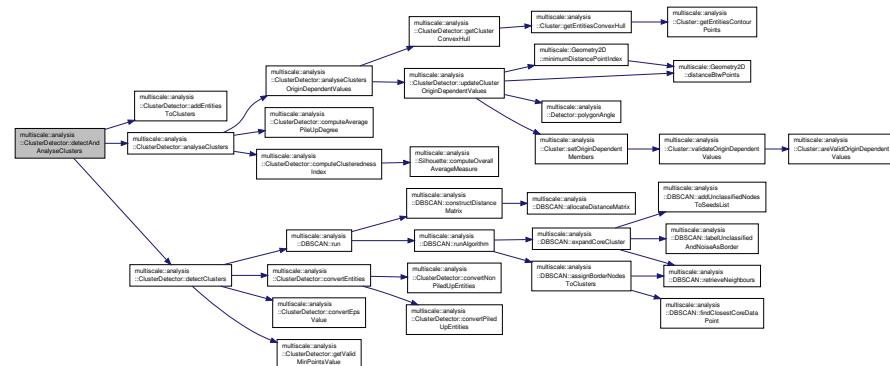
<i>entities</i>	Entities detected in the image
<i>clusters</i>	Clusters of entities detected in the image

Definition at line 72 of file ClusterDetector.cpp.

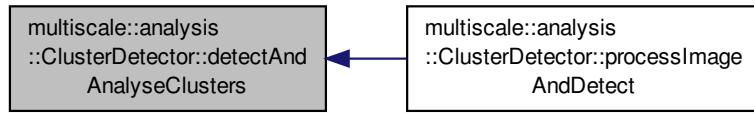
References addEntitiesToClusters(), analyseClusters(), CLUSTERING_UNCLASSIFIED, and detectClusters().

Referenced by processImageAndDetect().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.13 void ClusterDetector::detectClusters (const vector< Entity > & entities, vector< int > & clusterIndexes, int & nrOfClusters) [protected]

Detect the clusters of entities in the image.

Detect the clusters of entities in the image using Density Based scan (DBscan) clustering algorithm Clusters start from index 1, because cluster 0 contains only noise data/points.

Parameters

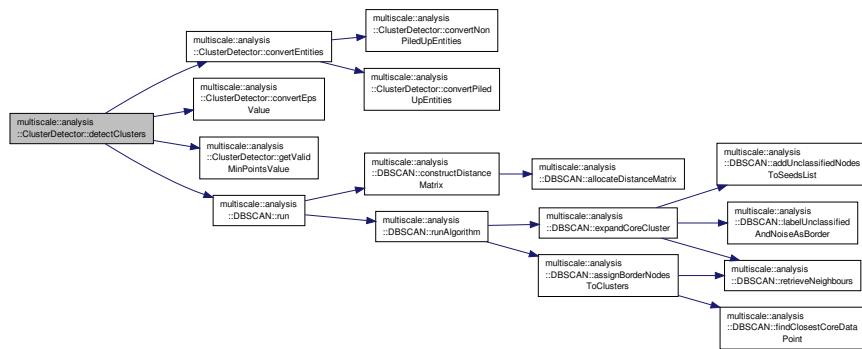
<i>entities</i>	Entities detected in the image
<i>clusterIndexes</i>	Indexes to which cluster each entity belongs
<i>nrOfClusters</i>	Total number of clusters

Definition at line 81 of file ClusterDetector.cpp.

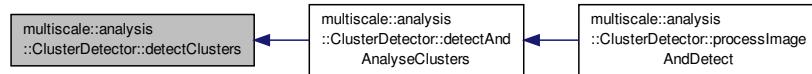
References `convertEntities()`, `convertEpsValue()`, `getValidMinPointsValue()`, and `multiscale::analysis::DBSCAN::run()`.

Referenced by `detectAndAnalyseClusters()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.14 virtual void multiscale::analysis::ClusterDetector::detectEntitiesInImage (vector< Entity > & entities) [protected], [pure virtual]

Detect the entities in the image.

Detect the entities in the image, compute their centre point and degree of pile up

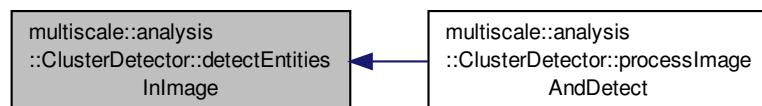
Parameters

<i>entities</i>	Entities detected in the image
-----------------	--------------------------------

Implemented in [multiscale::analysis::SimulationClusterDetector](#).

Referenced by [processImageAndDetect\(\)](#).

Here is the caller graph for this function:



7.10.3.15 `vector< Point > ClusterDetector::getClusterConvexHull (Cluster & cluster) [protected]`

Return the convex hull of the given cluster.

Parameters

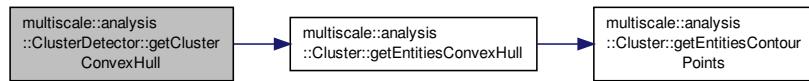
<code>cluster</code>	The given cluster
----------------------	-------------------

Definition at line 148 of file ClusterDetector.cpp.

References multiscale::analysis::Cluster::getEntitiesConvexHull().

Referenced by analyseClustersOriginDependentValues().

Here is the call graph for this function:



Here is the caller graph for this function:

7.10.3.16 `vector< Cluster > const & ClusterDetector::getClusters ()`

Get a const reference to the vector of detected clusters.

Definition at line 35 of file ClusterDetector.cpp.

References clusters.

7.10.3.17 `double ClusterDetector::getEps ()`

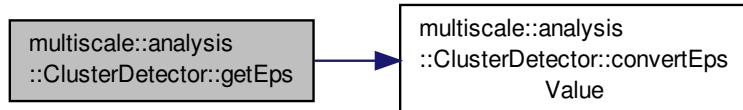
Get the value of the clustering algorithm parameter eps.

Definition at line 27 of file ClusterDetector.cpp.

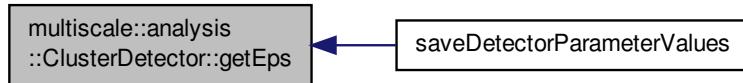
References convertEpsValue().

Referenced by saveDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.18 int ClusterDetector::getMinPoints()

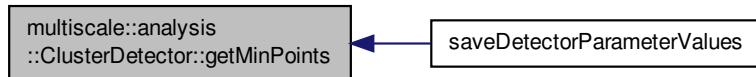
Get the value of the clustering algorithm parameter MinPoints.

Definition at line 31 of file ClusterDetector.cpp.

References minPoints.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.10.3.19 int ClusterDetector::getValidMinPointsValue() [protected]

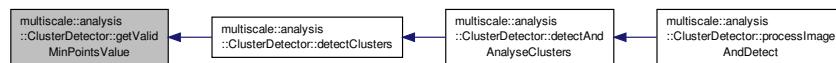
Return non-zero value for minPoints.

Definition at line 195 of file ClusterDetector.cpp.

References minPoints.

Referenced by detectClusters().

Here is the caller graph for this function:



7.10.3.20 void ClusterDetector::initialiseDetectorSpecificFields() [override], [protected], [virtual]

Initialise clustering values.

Implements [multiscale::analysis::Detector](#).

Definition at line 51 of file ClusterDetector.cpp.

References eps, and minPoints.

7.10.3.21 void ClusterDetector::outputResultsToCsvFile(ostream & fOut) [override], [protected], [virtual]

Output the information computed for the clusters to a csv file.

Parameters

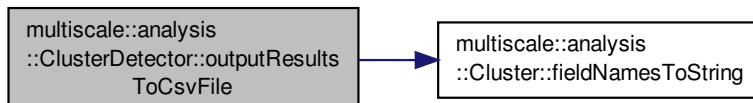
<i>fOut</i>	Output file stream
-------------	--------------------

Implements [multiscale::analysis::Detector](#).

Definition at line 177 of file ClusterDetector.cpp.

References avgPileUpDegree, clusterednessIndex, clusters, multiscale::analysis::Cluster::fieldNamesToString(), OUTPUT_CLUSTEREDNESS, and OUTPUT_PILE_UP.

Here is the call graph for this function:



7.10.3.22 void ClusterDetector::processImageAndDetect() [override], [protected], [virtual]

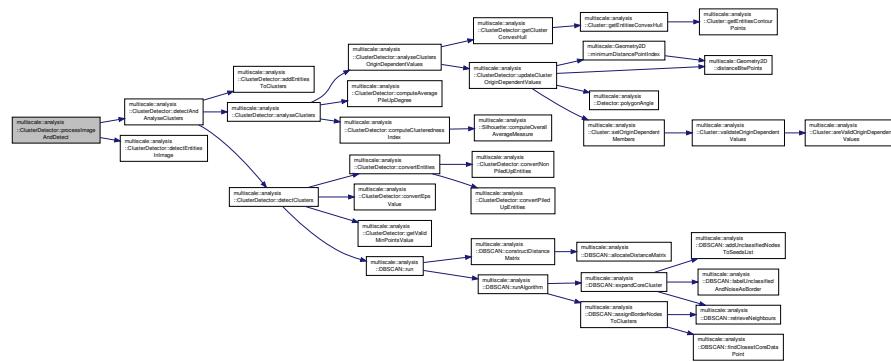
Process the provided image and detect clusters in it.

Implements [multiscale::analysis::Detector](#).

Definition at line 65 of file ClusterDetector.cpp.

References clusters, detectAndAnalyseClusters(), and detectEntitiesInImage().

Here is the call graph for this function:



7.10.3.23 void ClusterDetector::setEps (double eps)

Set the value of the clustering algorithm parameter `eps`.

Parameters

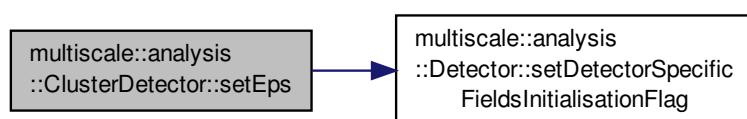
eps Value of the clustering algorithm parameter *eps*

Definition at line 39 of file ClusterDetector.cpp.

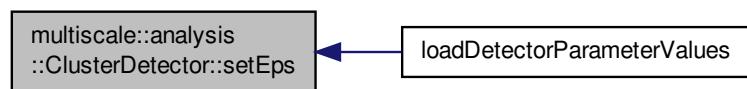
References `eps`, `EPS_MAX`, `EPS_MIN`, `EPS_REAL_MAX`, `EPS_REAL_MIN`, and `multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()`.

Referenced by `loadDetectorParameterValues()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.24 void ClusterDetector::setMinPoints (int minPoints)

Set the value of the clustering algorithm parameter MinPoints.

Parameters

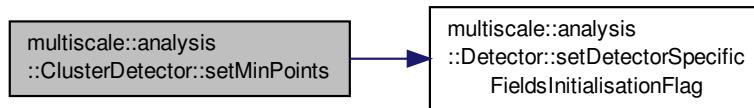
<i>minPoints</i>	Value of the clustering algorithm parameter MinPoints
------------------	---

Definition at line 45 of file ClusterDetector.cpp.

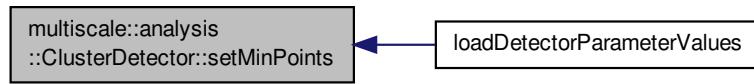
References *minPoints*, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Referenced by *loadDetectorParameterValues()*.

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.3.25 void ClusterDetector::updateClusterOriginDependentValues (Cluster & cluster, const vector< Point > & clusterConvexHull) [protected]

Update the cluster and compute the origin dependent values considering the convex hull.

The values which depend on the origin point are the distance of the cluster from the centre and the angle

Parameters

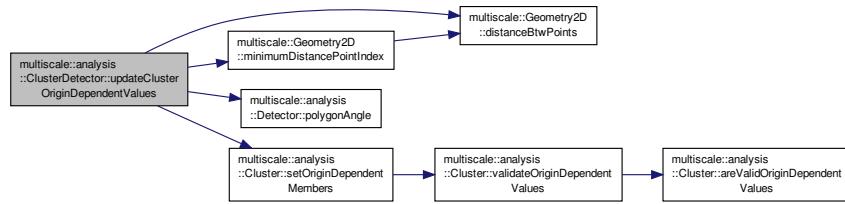
<i>cluster</i>	Cluster
<i>clusterConvex-Hull</i>	Convex hull of the cluster

Definition at line 139 of file ClusterDetector.cpp.

References multiscale::Geometry2D::distanceBtwPoints(), multiscale::Geometry2D::minimumDistancePointIndex(), multiscale::analysis::Detector::origin, multiscale::analysis::Detector::polygonAngle(), and multiscale::analysis::Cluster::setOriginDependentMembers().

Referenced by *analyseClustersOriginDependentValues()*.

Here is the call graph for this function:



Here is the caller graph for this function:



7.10.4 Member Data Documentation

7.10.4.1 double multiscale::analysis::ClusterDetector::avgPileUpDegree [protected]

Average pile up degree of all clusters

Definition at line 39 of file ClusterDetector.hpp.

Referenced by analyseClusters(), ClusterDetector(), and outputResultsToCsvFile().

7.10.4.2 double multiscale::analysis::ClusterDetector::clusterednessIndex [protected]

Index of clusteredness for all clusters

Definition at line 38 of file ClusterDetector.hpp.

Referenced by analyseClusters(), ClusterDetector(), and outputResultsToCsvFile().

7.10.4.3 vector<Cluster> multiscale::analysis::ClusterDetector::clusters [protected]

Clusters found in the image

Definition at line 49 of file ClusterDetector.hpp.

Referenced by clearPreviousDetectionResults(), getClusters(), outputResultsToCsvFile(), multiscale::analysis::SimulationClusterDetector::outputResultsToImage(), and processImageAndDetect().

7.10.4.4 double multiscale::analysis::ClusterDetector::entityPileupDegree [protected]

The pile up degree (intensity) of a grid position occupied by only

one entity

Definition at line 41 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), and multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition().

7.10.4.5 int multiscale::analysis::ClusterDetector::eps [protected]

DBSCAN algorithm parameter for specifying the maximum radius

of the neighbourhood

Definition at line 44 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), convertEpsValue(), createDetectorSpecificTrackbars(), initialiseDetectorSpecificFields(), and setEps().

7.10.4.6 int multiscale::analysis::ClusterDetector::minPoints [protected]

DBSCAN algorithm parameter for specifying the minimum number

of points in an eps-neighbourhood of that point

Definition at line 46 of file ClusterDetector.hpp.

Referenced by ClusterDetector(), createDetectorSpecificTrackbars(), getMinPoints(), getValidMinPointsValue(), initialiseDetectorSpecificFields(), and setMinPoints().

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

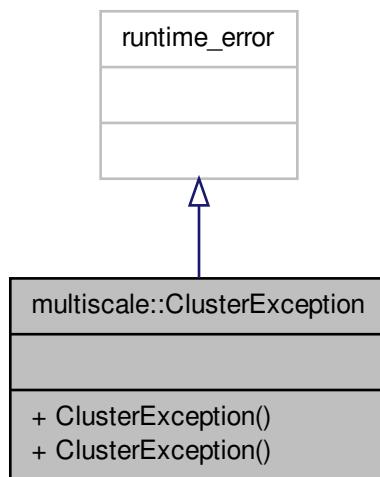
- include/multiscale/analysis/spatial/[ClusterDetector.hpp](#)
- src/multiscale/analysis/spatial/[ClusterDetector.cpp](#)

7.11 multiscale::ClusterException Class Reference

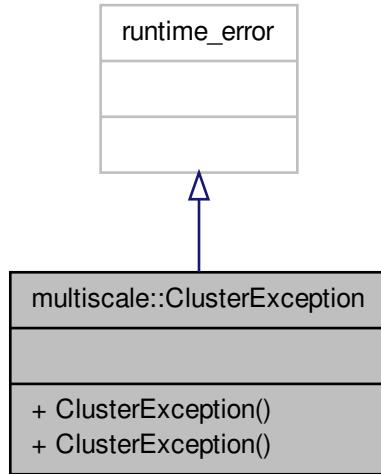
Exception class for the Cluster instances.

```
#include <ClusterException.hpp>
```

Inheritance diagram for multiscale::ClusterException:



Collaboration diagram for multiscale::ClusterException:



Public Member Functions

- [ClusterException \(const string &msg\)](#)
- [ClusterException \(const char *msg\)](#)

7.11.1 Detailed Description

Exception class for the Cluster instances.

Definition at line 13 of file ClusterException.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 multiscale::ClusterException::ClusterException (const string & msg) [inline]

Definition at line 17 of file ClusterException.hpp.

7.11.2.2 multiscale::ClusterException::ClusterException (const char * msg) [inline]

Definition at line 18 of file ClusterException.hpp.

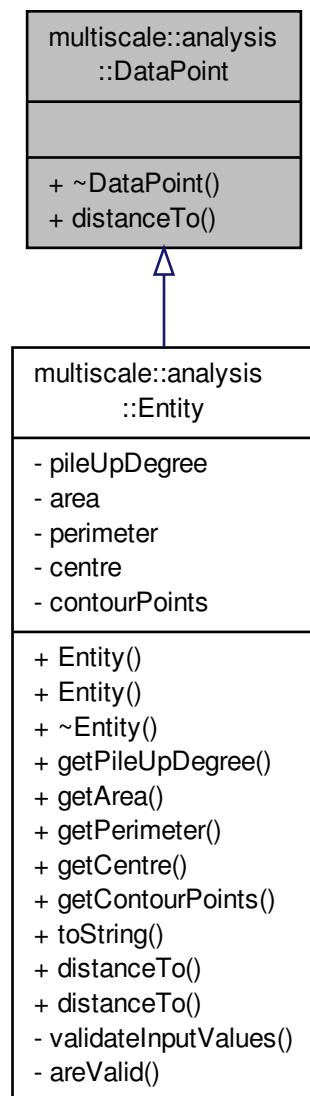
The documentation for this class was generated from the following file:

- include/multiscale/exception/ClusterException.hpp

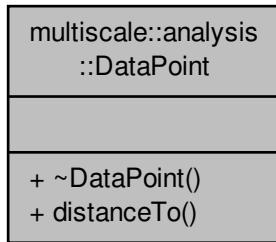
7.12 multiscale::analysis::DataPoint Class Reference

```
#include <DataPoint.hpp>
```

Inheritance diagram for multiscale::analysis::DataPoint:



Collaboration diagram for multiscale::analysis::DataPoint:



Public Member Functions

- virtual `~DataPoint ()`
- virtual double `distanceTo (shared_ptr< DataPoint > point)=0`
Compute the distance between this data point and another one.

7.12.1 Detailed Description

Definition at line 12 of file DataPoint.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 virtual multiscale::analysis::DataPoint::~DataPoint () [inline], [virtual]

Definition at line 16 of file DataPoint.hpp.

7.12.3 Member Function Documentation

7.12.3.1 virtual double multiscale::analysis::DataPoint::distanceTo (shared_ptr< DataPoint > point) [pure virtual]

Compute the distance between this data point and another one.

Parameters

<code>point</code>	Data point to which the distance is measured
--------------------	--

Implemented in [multiscale::analysis::Entity](#).

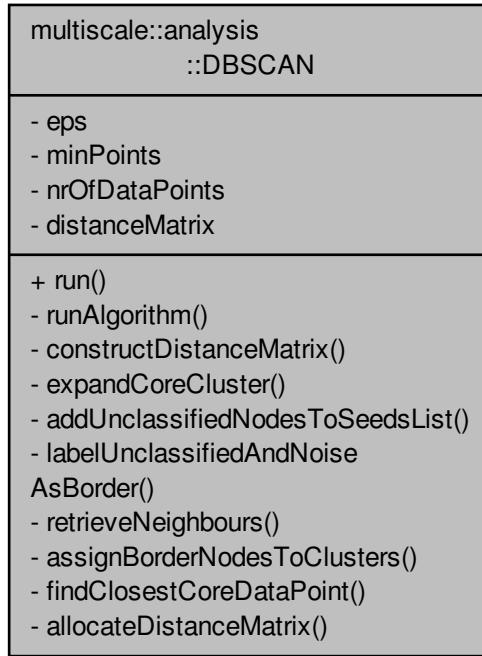
The documentation for this class was generated from the following file:

- include/multiscale/analysis/spatial/[DataPoint.hpp](#)

7.13 multiscale::analysis::DBSCAN Class Reference

```
#include <DBSCAN.hpp>
```

Collaboration diagram for multiscale::analysis::DBSCAN:



Static Public Member Functions

- static void **run** (const vector< shared_ptr< **DataPoint** >> &dataPoints, vector< int > &clusterIndexes, int &nrOfClusters, double **eps**, int **minPoints**)

Run the improved DBSCAN algorithm on the provided set of points.

Static Private Member Functions

- static void **runAlgorithm** (const vector< shared_ptr< **DataPoint** >> &dataPoints, vector< int > &clusterIndexes, int &nrOfClusters)

Run the improved DBSCAN algorithm on the provided set of points.
- static void **constructDistanceMatrix** (const vector< shared_ptr< **DataPoint** >> &dataPoints)

Construct the distance matrix between any two data points.
- static bool **expandCoreCluster** (vector< int > &clusterIndexes, int coreDataPointIndex, int clusterId)

Expand the cluster around the given core data point.
- static void **addUnclassifiedNodesToSeedsList** (const vector< int > &neighbours, const vector< int > &clusterIndexes, vector< int > &seeds)

Add all unclassified neighbour nodes to the seeds list.
- static void **labelUnclassifiedAndNoiseAsBorder** (const vector< int > &neighbours, vector< int > &clusterIndexes)

Label all unclassified and noise neighbour nodes as border nodes.

- static vector< int > [retrieveNeighbours](#) (int dataPointIndex)

Retrieve the list of neighbour indexes which are at a distance < eps far from the given data point.
- static void [assignBorderNodesToClusters](#) (vector< int > &clusterIndexes)

Assign the border nodes to the clusters to which the closest core objects belong.
- static int [findClosestCoreDataPoint](#) (const vector< int > &neighbours, int borderDataPointIndex, const vector< int > &clusterIndexes)

Find the closest core data point from the given set of neighbours to the given border data point.
- static void [allocateDistanceMatrix](#) ()

Allocate the distance matrix.

Static Private Attributes

- static double [eps](#) = 0
- static unsigned int [minPoints](#) = 0
- static unsigned int [nrOfDataPoints](#) = 0
- static vector< vector< double > > [distanceMatrix](#)

7.13.1 Detailed Description

Definition at line 20 of file DBSCAN.hpp.

7.13.2 Member Function Documentation

7.13.2.1 void DBSCAN::addUnclassifiedNodesToSeedsList (const vector< int > & *neighbours*, const vector< int > & *clusterIndexes*, vector< int > & *seeds*) [static], [private]

Add all unclassified neighbour nodes to the seeds list.

Parameters

<i>neighbours</i>	Neighbour nodes
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs
<i>seeds</i>	List of seeds (see DBSCAN algorithm)

Definition at line 85 of file DBSCAN.cpp.

References CLUSTERING_UNCLASSIFIED.

Referenced by [expandCoreCluster\(\)](#).

Here is the caller graph for this function:



7.13.2.2 void DBSCAN::allocateDistanceMatrix () [static], [private]

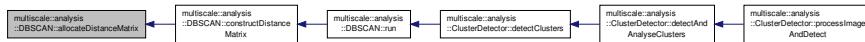
Allocate the distance matrix.

Definition at line 148 of file DBSCAN.cpp.

References [distanceMatrix](#), and [nrOfDataPoints](#).

Referenced by `constructDistanceMatrix()`.

Here is the caller graph for this function:



7.13.2.3 void DBSCAN::assignBorderNodesToClusters (`vector< int > & clusterIndexes`) [static], [private]

Assign the border nodes to the clusters to which the closest core objects belong.

Parameters

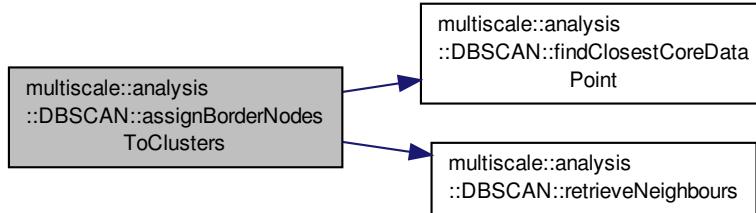
<code>clusterIndexes</code>	Indexes to which cluster each data point belongs
-----------------------------	--

Definition at line 117 of file DBSCAN.cpp.

References CLUSTERING_BORDER, `findClosestCoreDataPoint()`, `nrOfDataPoints`, and `retrieveNeighbours()`.

Referenced by `runAlgorithm()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.2.4 void DBSCAN::constructDistanceMatrix (`const vector< shared_ptr< DataPoint >> & dataPoints`) [static], [private]

Construct the distance matrix between any two data points.

Parameters

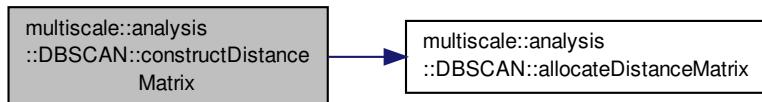
<code>dataPoints</code>	Data points
-------------------------	-------------

Definition at line 44 of file DBSCAN.cpp.

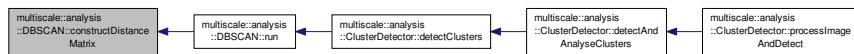
References allocateDistanceMatrix(), distanceMatrix, and nrOfDataPoints.

Referenced by run().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.2.5 bool DBSCAN::expandCoreCluster (vector< int > & clusterIndexes, int coreDataPointIndex, int clusterId) [static], [private]

Expand the cluster around the given core data point.

Parameters

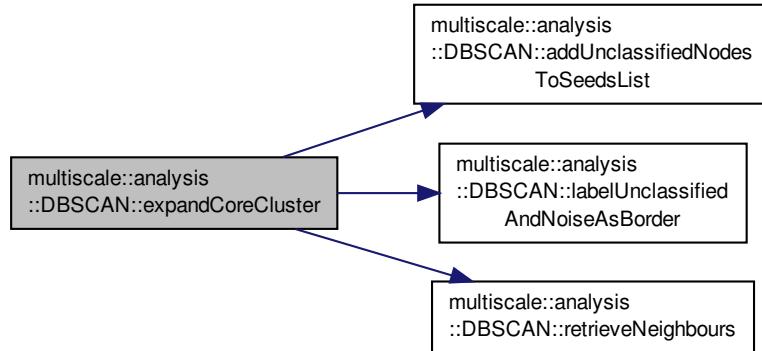
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs
<i>coreDataPoint- Index</i>	Core data point index
<i>clusterId</i>	Id of the cluster to which the core data point belongs

Definition at line 57 of file DBSCAN.cpp.

References addUnclassifiedNodesToSeedsList(), CLUSTERING_NOISE, labelUnclassifiedAndNoiseAsBorder(), minPoints, and retrieveNeighbours().

Referenced by runAlgorithm().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.2.6 int DBSCAN::findClosestCoreDataPoint (const vector< int > & neighbours, int borderDataPointIndex, const vector< int > & clusterIndexes) [static], [private]

Find the closest core data point from the given set of neighbours to the given border data point.

Parameters

<i>neighbours</i>	Set of neighbours
<i>borderDataPoint-Index</i>	Index of the border data point
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs

Definition at line 128 of file DBSCAN.cpp.

References distanceMatrix.

Referenced by assignBorderNodesToClusters().

Here is the caller graph for this function:



7.13.2.7 void DBSCAN::labelUnclassifiedAndNoiseAsBorder (const vector< int > & neighbours, vector< int > & clusterIndexes) [static], [private]

Label all unclassified and noise neighbour nodes as border nodes.

Parameters

<i>neighbours</i>	Neighbour nodes
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs

Definition at line 94 of file DBSCAN.cpp.

References CLUSTERING_BORDER, CLUSTERING_NOISE, and CLUSTERING_UNCLASSIFIED.

Referenced by expandCoreCluster().

Here is the caller graph for this function:

**7.13.2.8 vector< int > DBSCAN::retrieveNeighbours (int *dataPointIndex*) [static], [private]**

Retrieve the list of neighbour indexes which are at a distance < eps far from the given data point.

Parameters

<i>dataPointIndex</i>	Index of the data point for which the neighbours will be retrieved
-----------------------	--

Definition at line 103 of file DBSCAN.cpp.

References distanceMatrix, eps, and nrOfDataPoints.

Referenced by assignBorderNodesToClusters(), and expandCoreCluster().

Here is the caller graph for this function:

**7.13.2.9 void DBSCAN::run (const vector< shared_ptr< DataPoint >> & *dataPoints*, vector< int > & *clusterIndexes*, int & *nrOfClusters*, double *eps*, int *minPoints*) [static]**

Run the improved [DBSCAN](#) algorithm on the provided set of points.

The implementation of the improved [DBSCAN](#) algorithm is based on the paper: T. N. Tran, K. Drab, and M. Daszykowski, ‘Revised [DBSCAN](#) algorithm to cluster data with dense adjacent clusters’, Chemometrics and Intelligent Laboratory Systems, vol. 120, pp. 92–96, Jan. 2013.

Clusters start from index 1, because cluster 0 contains only noise data/points.

Parameters

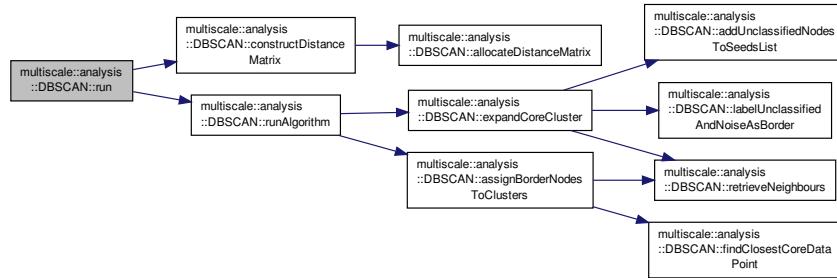
<i>dataPoints</i>	Collection of data points
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs
<i>nrOfClusters</i>	Total number of clusters
<i>eps</i>	Maximum distance between two neighbours
<i>minPoints</i>	Minimum number of points in one cluster

Definition at line 15 of file DBSCAN.cpp.

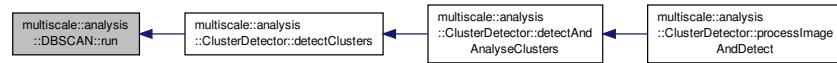
References constructDistanceMatrix(), eps, minPoints, nrOfDataPoints, and runAlgorithm().

Referenced by multiscale::analysis::ClusterDetector::detectClusters().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.2.10 void DBSCAN::runAlgorithm (const vector< shared_ptr< DataPoint >> & dataPoints, vector< int > & clusterIndexes, int & nrOfClusters) [static], [private]

Run the improved [DBSCAN](#) algorithm on the provided set of points.

The implementation of the improved [DBSCAN](#) algorithm is based on the paper: T. N. Tran, K. Drab, and M. Daszykowski, ‘Revised [DBSCAN](#) algorithm to cluster data with dense adjacent clusters’, Chemometrics and Intelligent Laboratory Systems, vol. 120, pp. 92–96, Jan. 2013.

Clusters start from index 1, because cluster 0 contains only noise data/points.

Parameters

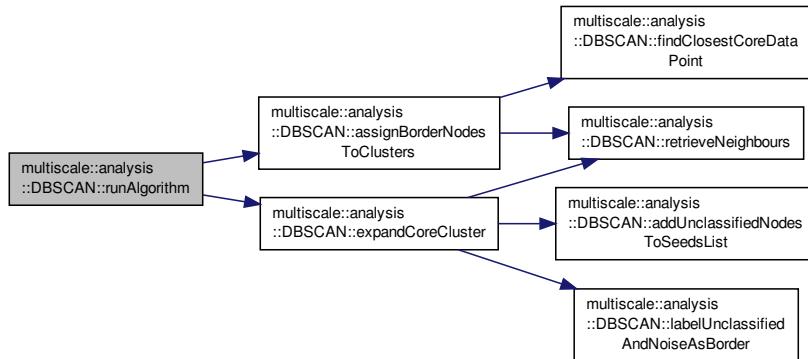
<i>dataPoints</i>	Collection of data points
<i>clusterIndexes</i>	Indexes to which cluster each data point belongs
<i>nrOfClusters</i>	Total number of clusters

Definition at line 26 of file DBSCAN.cpp.

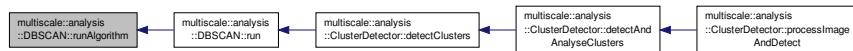
References assignBorderNodesToClusters(), CLUSTERING_UNCLASSIFIED, expandCoreCluster(), and nrOfDataPoints.

Referenced by run().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3 Member Data Documentation

7.13.3.1 `vector<vector< double > > DBSCAN::distanceMatrix` [static], [private]

The matrix containing the distances between any two data points

Definition at line 31 of file DBSCAN.hpp.

Referenced by `allocateDistanceMatrix()`, `constructDistanceMatrix()`, `findClosestCoreDataPoint()`, and `retrieveNeighbours()`.

7.13.3.2 `double DBSCAN::eps = 0` [static], [private]

DBSCAN algorithm parameter for specifying the maximum radius

of the neighbourhood

Definition at line 24 of file DBSCAN.hpp.

Referenced by `retrieveNeighbours()`, and `run()`.

7.13.3.3 `unsigned int DBSCAN::minPoints = 0` [static], [private]

DBSCAN algorithm parameter for specifying the minimum number

of points in an `eps`-neighbourhood of that point

Definition at line 26 of file DBSCAN.hpp.

Referenced by `expandCoreCluster()`, and `run()`.

7.13.3.4 `unsigned int DBSCAN::nrOfDataPoints = 0` [static], [private]

Number of data points in the data set

Definition at line 29 of file DBSCAN.hpp.

Referenced by `allocateDistanceMatrix()`, `assignBorderNodesToClusters()`, `constructDistanceMatrix()`, `retrieveNeighbours()`, `run()`, and `runAlgorithm()`.

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

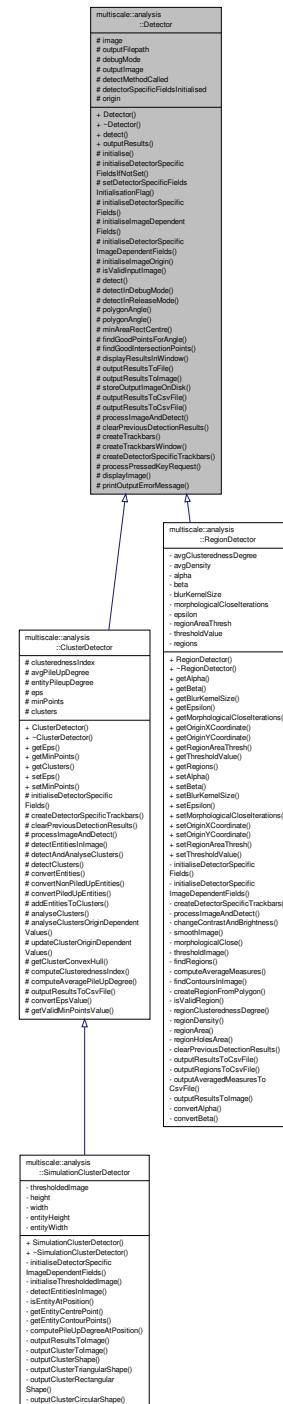
- include/multiscale/analysis/spatial/[DBSCAN.hpp](#)

- src/multiscale/analysis/spatial/[DBSCAN.cpp](#)

7.14 multiscale::analysis::Detector Class Reference

```
#include <Detector.hpp>
```

Inheritance diagram for multiscale::analysis::Detector:



Collaboration diagram for multiscale::analysis::Detector:

multiscale::analysis ::Detector
<pre># image # outputPath # debugMode # outputImage # detectMethodCalled # detectorSpecificFieldsInitialised # origin + Detector() + ~Detector() + detect() + outputResults() # initialise() # initialiseDetectorSpecific FieldsIfNotSet() # setDetectorSpecificFields InitialisationFlag() # initialiseDetectorSpecific Fields() # initialiseImageDependent Fields() # initialiseDetectorSpecific ImageDependentFields() # initialiseImageOrigin() # isValidateInputImage() # detect() # detectInDebugMode() # detectInReleaseMode() # polygonAngle() # polygonAngle() # minAreaRectCentre() # findGoodPointsForAngle() # findGoodIntersectionPoints() # displayResultsInWindow() # outputResultsToFile() # outputResultsToImage() # storeOutputImageOnDisk() # outputResultsToCsvFile() # outputResultsToCsvFile() # processImageAndDetect() # clearPreviousDetectionResults() # createTrackbars() # createTrackbarsWindow() # createDetectorSpecificTrackbars() # processPressedKeyRequest() # displayImage() # printOutputErrorMessage()</pre>

Public Member Functions

- **Detector** (bool `debugMode=false`)
- **virtual ~Detector ()**
- **void detect** (const Mat &`inputImage`)

Run the detection procedure on the given image.

- **void outputResults** (const string &`outputPath`)

Output the results to the given file.

Protected Member Functions

- void **initialise** ()

Initialisation function for the class.
- void **initialiseDetectorSpecificFieldsIfNotSet** ()

Initialisation of the detector specific values in case they were not set.
- void **setDetectorSpecificFieldsInitialisationFlag** (bool flag=true)

Set the detector specific fields initialisation flag to true.
- virtual void **initialiseDetectorSpecificFields** ()=0

Initialisation of the detector specific values.
- void **initialiseImageDependentFields** ()

Initialisation of the image dependent values.
- virtual void **initialiseDetectorSpecificImageDependentFields** ()=0

Initialisation of the detector specific image dependent values.
- void **initialiseImageOrigin** ()

Initialisation of the image origin.
- bool **isValidInputImage** (const Mat &inputImage)

Check if the image is valid.
- void **detect** ()

Run the detection procedure.
- void **detectInDebugMode** ()

Run the detection procedure when in debug mode.
- void **detectInReleaseMode** ()

Run the detection procedure when in release mode (i.e. non-debug mode)
- double **polygonAngle** (const vector< Point > &polygon, unsigned int closestPointIndex)

Compute the angle of the polygon.
- double **polygonAngle** (const vector< Point > &polygonConvexHull, const Point &closestPoint)

Compute the angle of the polygon.
- void **minAreaRectCentre** (const vector< Point > &polygon, Point ¢re)

Get the centre of the minimum area bounding rectangle.
- void **findGoodPointsForAngle** (const vector< Point > &polygonConvexHull, const Point &boundingRectCentre, const Point &closestPoint, vector< Point > &goodPointsForAngle)

Find the points for determining the angle of the polygon.
- void **findGoodIntersectionPoints** (const vector< Point > &polygonConvexHull, const Point &edgePointA, const Point &edgePointB, vector< Point > &goodPointsForAngle)

Find good intersection points for computing the angle of the polygon.
- void **displayResultsInWindow** ()

Display the results in a window.
- void **outputResultsToFile** ()

Output the results to file(s).
- virtual void **outputResultsToImage** ()=0

Output the results to an image.
- void **storeOutputImageOnDisk** ()

Store the image with the output results on disk.
- void **outputResultsToCsvFile** ()

Output the results to a file.
- virtual void **outputResultsToCsvFile** (ofstream &fout)=0

Output the results to a file using the provided output file stream.
- virtual void **processImageAndDetect** ()=0

Process the input image and detect objects/entities of interest.

- virtual void [clearPreviousDetectionResults \(\)=0](#)
Clear the results from the previous detection.
- void [createTrackbars \(\)](#)
Create the trackbars which allow the user to change the values of the parameters.
- void [createTrackbarsWindow \(\)](#)
Create the window in which the trackbars are placed.
- virtual void [createDetectorSpecificTrackbars \(\)=0](#)
Create the trackbars specific to the used detector.
- void [processPressedKeyRequest \(char &pressedKey\)](#)
Process the request of the user by pressing the key.
- void [displayImage \(const Mat &image, const string &windowName\)](#)
Display an image in a particular window.
- void [printOutputErrorMessage \(\)](#)
Print error message, because the detect method was not called before calling the output method.

Protected Attributes

- Mat [image](#)
- string [outputFilepath](#)
- bool [debugMode](#)
- Mat [outputImage](#)
- bool [detectMethodCalled](#)
- bool [detectorSpecificFieldsInitialised](#)
- Point [origin](#)

7.14.1 Detailed Description

Definition at line 29 of file Detector.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Detector::Detector (bool debugMode = false)

Definition at line 11 of file Detector.cpp.

7.14.2.2 Detector::~Detector () [virtual]

Definition at line 18 of file Detector.cpp.

7.14.3 Member Function Documentation

7.14.3.1 virtual void multiscale::analysis::Detector::clearPreviousDetectionResults () [protected], [pure virtual]

Clear the results from the previous detection.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

7.14.3.2 `virtual void multiscale::analysis::Detector::createDetectorSpecificTrackbars() [protected], [pure virtual]`

Create the trackbars specific to the used detector.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

7.14.3.3 `void Detector::createTrackbars() [protected]`

Create the trackbars which allow the user to change the values of the parameters.

Definition at line 189 of file `Detector.cpp`.

7.14.3.4 `void Detector::createTrackbarsWindow() [protected]`

Create the window in which the trackbars are placed.

Definition at line 194 of file `Detector.cpp`.

References `WIN_OUTPUT_IMAGE`.

7.14.3.5 `void Detector::detect(const Mat & inputImage)`

Run the detection procedure on the given image.

Parameters

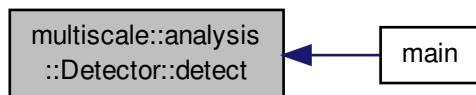
<code><i>inputImage</i></code>	The input image
--------------------------------	-----------------

Definition at line 23 of file `Detector.cpp`.

References `ERR_INVALID_IMAGE`.

Referenced by `main()`.

Here is the caller graph for this function:



7.14.3.6 `void Detector::detect() [protected]`

Run the detection procedure.

Definition at line 79 of file `Detector.cpp`.

7.14.3.7 `void Detector::detectInDebugMode() [protected]`

Run the detection procedure when in debug mode.

Definition at line 89 of file Detector.cpp.

References KEY_ESC.

7.14.3.8 void Detector::detectInReleaseMode() [protected]

Run the detection procedure when in release mode (i.e. non-debug mode)

Definition at line 104 of file Detector.cpp.

7.14.3.9 void Detector::displayImage(const Mat & image, const string & windowName) [protected]

Display an image in a particular window.

Parameters

<i>image</i>	The image
<i>windowName</i>	The name of the window

Definition at line 205 of file Detector.cpp.

7.14.3.10 void Detector::displayResultsInWindow() [protected]

Display the results in a window.

Definition at line 159 of file Detector.cpp.

References WIN_OUTPUT_IMAGE.

7.14.3.11 void Detector::findGoodIntersectionPoints(const vector< Point > & polygonConvexHull, const Point & edgePointA, const Point & edgePointB, vector< Point > & goodPointsForAngle) [protected]

Find good intersection points for computing the angle of the polygon.

Parameters

<i>polygonConvex-Hull</i>	The convex hull of the polygon
<i>edgePointA</i>	Point A on the edge
<i>edgePointB</i>	Point B on the edge
<i>goodPointsFor-Angle</i>	The "good" points for computing the angle

Definition at line 146 of file Detector.cpp.

References multiscale::Geometry2D::lineSegmentIntersection().

Here is the call graph for this function:



7.14.3.12 void Detector::findGoodPointsForAngle (const vector< Point > & *polygonConvexHull*, const Point & *boundingRectCentre*, const Point & *closestPoint*, vector< Point > & *goodPointsForAngle*) [protected]

Find the points for determining the angle of the polygon.

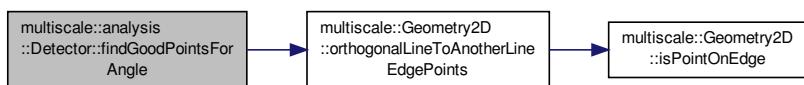
Parameters

<i>polygonConvex-Hull</i>	Convex hull of polygon
<i>boundingRect-Centre</i>	Centre of the rotated rectangle enclosing the polygon convex hull
<i>closestPoint</i>	Closest point to the origin from the set of points defining the polygon
<i>goodPointsFor-Angle</i>	The points which are relevant for computing the angle

Definition at line 134 of file Detector.cpp.

References multiscale::Geometry2D::orthogonalLineToAnotherLineEdgePoints().

Here is the call graph for this function:



7.14.3.13 void Detector::initialise () [protected]

Initialisation function for the class.

Definition at line 44 of file Detector.cpp.

7.14.3.14 virtual void multiscale::analysis::Detector::initialiseDetectorSpecificFields () [protected], [pure virtual]

Initialisation of the detector specific values.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

7.14.3.15 void Detector::initialiseDetectorSpecificFieldsIfNotSet () [protected]

Initialisation of the detector specific values in case they were not set.

Definition at line 49 of file Detector.cpp.

7.14.3.16 virtual void multiscale::analysis::Detector::initialiseDetectorSpecificImageDependentFields () [protected], [pure virtual]

Initialisation of the detector specific image dependent values.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::SimulationClusterDetector](#).

7.14.3.17 void Detector::initialiseImageDependentFields() [protected]

Initialisation of the image dependent values.

Definition at line 63 of file Detector.cpp.

7.14.3.18 void Detector::initialiseImageOrigin() [protected]

Definition at line 68 of file Detector.cpp.

7.14.3.19 bool Detector::isValidInputImage(const Mat & *inputImage*) [protected]

Check if the image is valid.

Check if the number of dimensions = 2, if the number of rows and number of columns is greater than one and if the image is of type CV_8UC1

Parameters

<i>inputImage</i>	The input image
-------------------	-----------------

Definition at line 75 of file Detector.cpp.

7.14.3.20 void Detector::minAreaRectCentre(const vector< Point > & *polygon*, Point & *centre*) [protected]

Get the centre of the minimum area bounding rectangle.

Parameters

<i>polygon</i>	The polygon
<i>centre</i>	The centre of the bounding rectangle

Definition at line 128 of file Detector.cpp.

7.14.3.21 void Detector::outputResults(const string & *outputfilepath*)

Output the results to the given file.

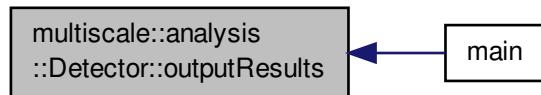
Parameters

<i>outputfilepath</i>	Path to the output file
-----------------------	-------------------------

Definition at line 34 of file Detector.cpp.

Referenced by main().

Here is the caller graph for this function:



7.14.3.22 void Detector::outputResultsToCsvFile () [protected]

Output the results to a file.

Definition at line 177 of file Detector.cpp.

References ERR_OUTPUT_FILE, and OUTPUT_EXTENSION.

7.14.3.23 virtual void multiscale::analysis::Detector::outputResultsToCsvFile (ostream & *fout*) [protected], [pure virtual]

Output the results to a file using the provided output file stream.

Parameters

<i>fout</i>	Output file stream
-------------	--------------------

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

7.14.3.24 void Detector::outputResultsToFile () [protected]

Output the results to file(s)

Definition at line 164 of file Detector.cpp.

7.14.3.25 virtual void multiscale::analysis::Detector::outputResultsToImage () [protected], [pure virtual]

Output the results to an image.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::SimulationClusterDetector](#).

7.14.3.26 double Detector::polygonAngle (const vector< Point > & *polygon*, unsigned int *closestPointIndex*) [protected]

Compute the angle of the polygon.

Compute the angle determined by the closest point to the origin and the points P1 and P2. These points are obtained from the intersection of the polygon with the line which is orthogonal to the line AB where:

- Point A is the polygon point closest to the origin;
- Point B is the centre point of the bounding rotated rectangle.

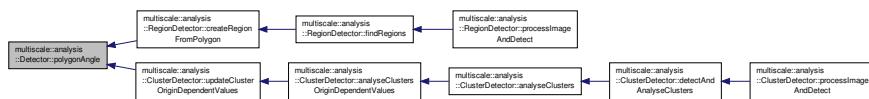
Parameters

<i>polygon</i>	Given polygon
<i>closestPoint-Index</i>	Index of the closest point to the origin from the set of points defining the polygon

Definition at line 109 of file Detector.cpp.

Referenced by multiscale::analysis::RegionDetector::createRegionFromPolygon(), and multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues().

Here is the caller graph for this function:



7.14.3.27 double Detector::polygonAngle (const vector< Point > & *polygonConvexHull*, const Point & *closestPoint*) [protected]

Compute the angle of the polygon.

Compute the angle determined by the closest point to the origin and the points P1 and P2. These points are obtained from the intersection of the convex hull with the line AB, determined by points A and B. Points A and B are the middle points of the sides of the rotated rectangle enclosing the polygon that are orthogonal to the line which is the nearest to the closestPoint.

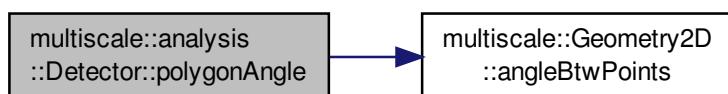
Parameters

<i>polygonConvex-Hull</i>	Convex hull of polygon
<i>closestPoint</i>	Closest point to the origin from the set of points defining the polygon

Definition at line 117 of file Detector.cpp.

References multiscale::Geometry2D::angleBtwPoints().

Here is the call graph for this function:



7.14.3.28 void Detector::printOutputErrorMessage () [protected]

Print error message, because the detect method was not called before calling the output method.

Definition at line 210 of file Detector.cpp.

References ERR_OUTPUT WITHOUT_DETECT.

7.14.3.29 virtual void multiscale::analysis::Detector::processImageAndDetect() [protected], [pure virtual]

Process the input image and detect objects/entities of interest.

Implemented in [multiscale::analysis::RegionDetector](#), and [multiscale::analysis::ClusterDetector](#).

7.14.3.30 void Detector::processPressedKeyRequest(char & pressedKey) [protected]

Process the request of the user by pressing the key.

Parameters

<code>pressedKey</code>	Key pressed by the user, if a key was pressed, or "-1", otherwise
-------------------------	---

Definition at line 199 of file `Detector.cpp`.

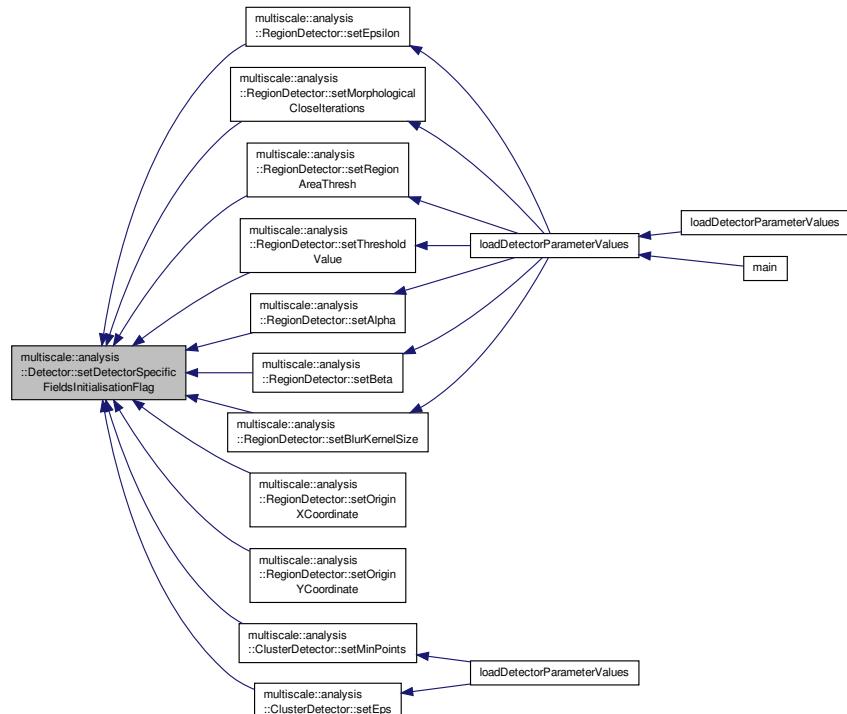
7.14.3.31 void Detector::setDetectorSpecificFieldsInitialisationFlag(bool flag = true) [protected]

Set the detector specific fields initialisation flag to true.

Definition at line 57 of file `Detector.cpp`.

Referenced by `multiscale::analysis::RegionDetector::setAlpha()`, `multiscale::analysis::RegionDetector::setBeta()`, `multiscale::analysis::RegionDetector::setBlurKernelSize()`, `multiscale::analysis::ClusterDetector::setEps()`, `multiscale::analysis::RegionDetector::setEpsilon()`, `multiscale::analysis::ClusterDetector::setMinPoints()`, `multiscale::analysis::RegionDetector::setMorphologicalCloseIterations()`, `multiscale::analysis::RegionDetector::setOriginXCoordinate()`, `multiscale::analysis::RegionDetector::setOriginYCoordinate()`, `multiscale::analysis::RegionDetector::setRegionAreaThresh()`, and `multiscale::analysis::RegionDetector::setThresholdValue()`.

Here is the caller graph for this function:



7.14.3.32 void Detector::storeOutputImageOnDisk() [protected]

Store the image with the output results on disk.

Definition at line 171 of file Detector.cpp.

References IMG_EXTENSION.

7.14.4 Member Data Documentation

7.14.4.1 bool multiscale::analysis::Detector::debugMode [protected]

Flag for indicating if debug mode is set

Definition at line 35 of file Detector.hpp.

7.14.4.2 bool multiscale::analysis::Detector::detectMethodCalled [protected]

Flag for indicating if the detect method was called

Definition at line 39 of file Detector.hpp.

7.14.4.3 bool multiscale::analysis::Detector::detectorSpecificFieldsInitialised [protected]

Flag for indicating if the parameters were

Definition at line 40 of file Detector.hpp.

7.14.4.4 Mat multiscale::analysis::Detector::image [protected]

Input image

Definition at line 33 of file Detector.hpp.

Referenced by multiscale::analysis::RegionDetector::changeContrastAndBrightness(), multiscale::analysis::SimulationClusterDetector::computePileUpDegreeAtPosition(), multiscale::analysis::SimulationClusterDetector::initialiseDetectorSpecificImageDependentFields(), multiscale::analysis::SimulationClusterDetector::initialiseThresholdedImage(), multiscale::analysis::SimulationClusterDetector::outputResultsToImage(), multiscale::analysis::RegionDetector::outputResultsToImage(), multiscale::analysis::RegionDetector::regionArea(), multiscale::analysis::RegionDetector::regionClusterednessDegree(), and multiscale::analysis::RegionDetector::regionDensity().

7.14.4.5 Point multiscale::analysis::Detector::origin [protected]

The point representing the origin

Definition at line 42 of file Detector.hpp.

Referenced by multiscale::analysis::RegionDetector::createRegionFromPolygon(), multiscale::analysis::RegionDetector::getOriginXCoordinate(), multiscale::analysis::RegionDetector::getOriginYCoordinate(), multiscale::analysis::RegionDetector::setOriginXCoordinate(), multiscale::analysis::RegionDetector::setOriginYCoordinate(), and multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues().

7.14.4.6 string multiscale::analysis::Detector::outputFilepath [protected]

Path of the output file

Definition at line 34 of file Detector.hpp.

7.14.4.7 Mat multiscale::analysis::Detector::outputImage [protected]

Image for displaying the results

Definition at line 37 of file `Detector.hpp`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputResultsToImage()`, and `multiscale::analysis::RegionDetector::outputResultsToImage()`.

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

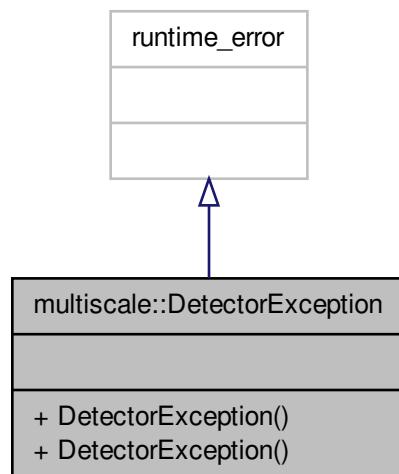
- `include/multiscale/analysis/spatial/Detector.hpp`
- `src/multiscale/analysis/spatial/Detector.cpp`

7.15 multiscale::DetectorException Class Reference

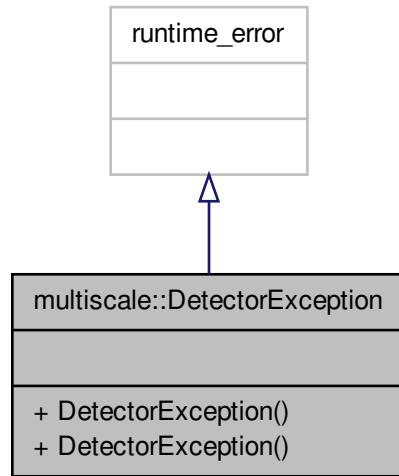
Exception class for the `Detector` class.

```
#include <DetectorException.hpp>
```

Inheritance diagram for `multiscale::DetectorException`:



Collaboration diagram for multiscale::DetectorException:



Public Member Functions

- [DetectorException \(const string &msg\)](#)
- [DetectorException \(const char *msg\)](#)

7.15.1 Detailed Description

Exception class for the Detector class.

Definition at line 13 of file `DetectorException.hpp`.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 multiscale::DetectorException::DetectorException (const string & msg) [inline]

Definition at line 17 of file `DetectorException.hpp`.

7.15.2.2 multiscale::DetectorException::DetectorException (const char * msg) [inline]

Definition at line 18 of file `DetectorException.hpp`.

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

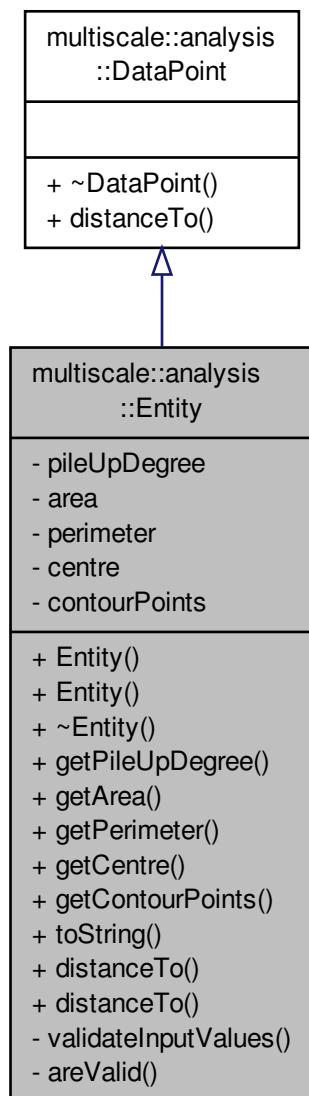
- [include/multiscale/exception/DetectorException.hpp](#)

7.16 multiscale::analysis::Entity Class Reference

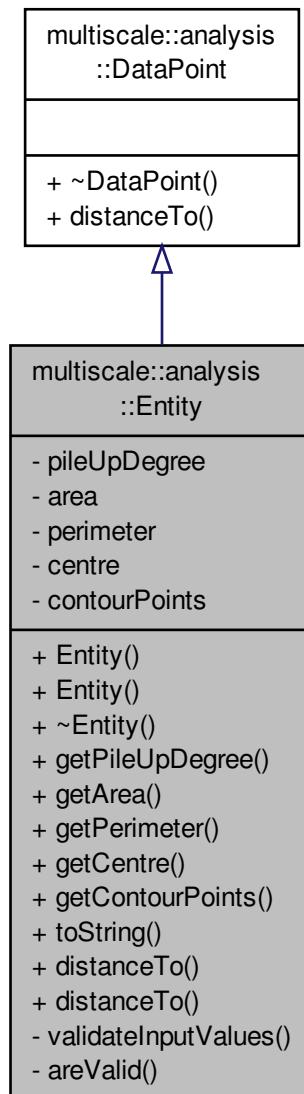
Class for representing an entity in an image (e.g. cell, organism etc.)

```
#include <Entity.hpp>
```

Inheritance diagram for multiscale::analysis::Entity:



Collaboration diagram for multiscale::analysis::Entity:



Public Member Functions

- `Entity` (unsigned int `pileUpDegree`, double `area`, double `perimeter`, const `Point2f ¢re`, const `vector<Point2f > &contourPoints`)
- `Entity` (const `Entity` &`entity`)
- `~Entity ()`
- unsigned int `getPileUpDegree () const`

Get the degree of pile up.
- double `getArea () const`

Get the area.
- double `getPerimeter () const`

Get the perimeter.

- `Point2f getCentre () const`
Get the point defining the centre of the entity.
- `vector< Point2f > getContourPoints () const`
Get the set of points defining the contour of the entity.
- `string toString ()`
Get a string representation of all the field values.
- `double distanceTo (shared_ptr< DataPoint > point) override`
Get the distance between this entity and another one.
- `double distanceTo (const Entity &entity)`
Get the distance between this entity and another one.

Private Member Functions

- `void validateInputValues (unsigned int pileUpDegree, double area, double perimeter, const Point2f ¢re, const vector< Point2f > &contourPoints)`
 - `bool areValid (unsigned int pileUpDegree, double area, double perimeter, const Point2f ¢re, const vector< Point2f > &contourPoints)`
- Check if the provided degree of pile up, area, centre and contour points are valid.*

Private Attributes

- `unsigned int pileUpDegree`
- `double area`
- `double perimeter`
- `Point2f centre`
- `vector< Point2f > contourPoints`

7.16.1 Detailed Description

Class for representing an entity in an image (e.g. cell, organism etc.)

Definition at line 24 of file Entity.hpp.

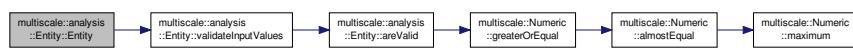
7.16.2 Constructor & Destructor Documentation

7.16.2.1 Entity::Entity (`unsigned int pileUpDegree, double area, double perimeter, const Point2f & centre, const vector< Point2f > & contourPoints`)

Definition at line 9 of file Entity.cpp.

References area, centre, contourPoints, perimeter, pileUpDegree, and validateInputValues().

Here is the call graph for this function:



7.16.2.2 Entity::Entity (const Entity & entity)

Definition at line 19 of file Entity.cpp.

References area, centre, contourPoints, perimeter, pileUpDegree, and validateInputValues().

Here is the call graph for this function:



7.16.2.3 Entity::~Entity ()

Definition at line 29 of file Entity.cpp.

7.16.3 Member Function Documentation

7.16.3.1 bool Entity::isValid (unsigned int pileUpDegree, double area, double perimeter, const Point2f & centre, const vector< Point2f > & contourPoints) [private]

Check if the provided degree of pile up, area, centre and contour points are valid.

Parameters

<i>pileUpDegree</i>	Degree of pile up
<i>area</i>	Area
<i>perimeter</i>	Perimeter
<i>centre</i>	Centre of the entity
<i>contourPoints</i>	Points defining the contour of the entity

Definition at line 74 of file Entity.cpp.

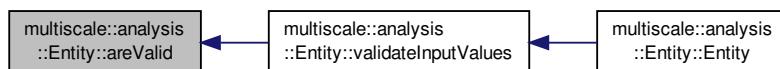
References multiscale::Numeric::greaterOrEqual().

Referenced by validateInputValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.3.2 double Entity::distanceTo (shared_ptr< DataPoint > *point*) [override], [virtual]

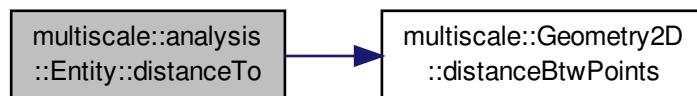
Get the distance between this entity and another one.

Implements [multiscale::analysis::DataPoint](#).

Definition at line 57 of file Entity.cpp.

References centre, and [multiscale::Geometry2D::distanceBtwPoints\(\)](#).

Here is the call graph for this function:



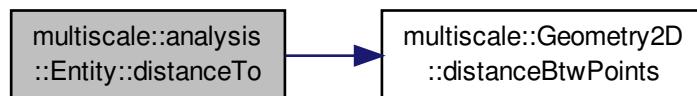
7.16.3.3 double Entity::distanceTo (const Entity & *entity*)

Get the distance between this entity and another one.

Definition at line 63 of file Entity.cpp.

References centre, and [multiscale::Geometry2D::distanceBtwPoints\(\)](#).

Here is the call graph for this function:



7.16.3.4 double Entity::getArea () const

Get the area.

Definition at line 35 of file Entity.cpp.

References area.

7.16.3.5 Point2f Entity::getCentre () const

Get the point defining the centre of the entity.

Definition at line 43 of file Entity.cpp.

References centre.

7.16.3.6 `vector< Point2f > Entity::getContourPoints() const`

Get the set of points defining the contour of the entity.

Definition at line 47 of file Entity.cpp.

References contourPoints.

7.16.3.7 `double Entity::getPerimeter() const`

Get the perimeter.

Definition at line 39 of file Entity.cpp.

References perimeter.

7.16.3.8 `unsigned int Entity::getPileUpDegree() const`

Get the degree of pile up.

Definition at line 31 of file Entity.cpp.

References pileUpDegree.

7.16.3.9 `string Entity::toString()`

Get a string representation of all the field values.

Definition at line 51 of file Entity.cpp.

References centre, OUTPUT_SEPARATOR, and pileUpDegree.

7.16.3.10 `void Entity::validateInputValues(unsigned int pileUpDegree, double area, double perimeter, const Point2f & centre, const vector< Point2f > & contourPoints) [private]`

Parameters

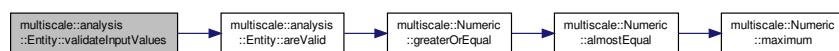
<code>pileUpDegree</code>	Degree of pile up
<code>area</code>	Area
<code>perimeter</code>	Perimeter
<code>centre</code>	Centre of the entity
<code>contourPoints</code>	Points defining the contour of the entity

Definition at line 67 of file Entity.cpp.

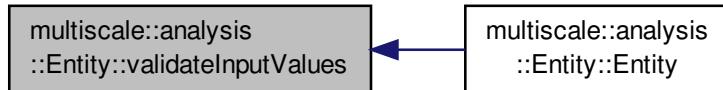
References isValid(), and ERR_INPUT.

Referenced by Entity().

Here is the call graph for this function:



Here is the caller graph for this function:



7.16.4 Member Data Documentation

7.16.4.1 double multiscale::analysis::Entity::area [private]

Area of the entity

Definition at line 29 of file Entity.hpp.

Referenced by Entity(), and getArea().

7.16.4.2 Point2f multiscale::analysis::Entity::centre [private]

Point defining the centre of the entity

Definition at line 32 of file Entity.hpp.

Referenced by distanceTo(), Entity(), getCentre(), and toString().

7.16.4.3 vector<Point2f> multiscale::analysis::Entity::contourPoints [private]

Set of points defining the contour of the entity

Definition at line 33 of file Entity.hpp.

Referenced by Entity(), and getContourPoints().

7.16.4.4 double multiscale::analysis::Entity::perimeter [private]

Perimeter of the entity

Definition at line 30 of file Entity.hpp.

Referenced by Entity(), and getPerimeter().

7.16.4.5 unsigned int multiscale::analysis::Entity::pileUpDegree [private]

Degree of pile up (relevant only if entities can pile up onto each other)

Definition at line 28 of file Entity.hpp.

Referenced by Entity(), getPileUpDegree(), and toString().

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

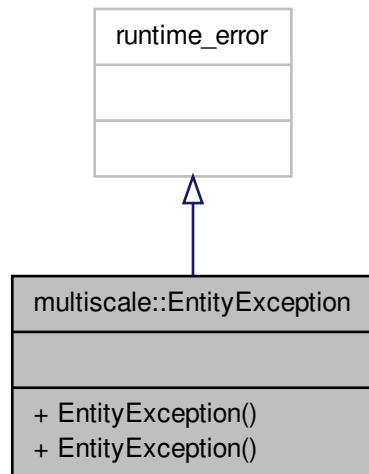
- include/multiscale/analysis/spatial/[Entity.hpp](#)
- src/multiscale/analysis/spatial/[Entity.cpp](#)

7.17 multiscale::EntityException Class Reference

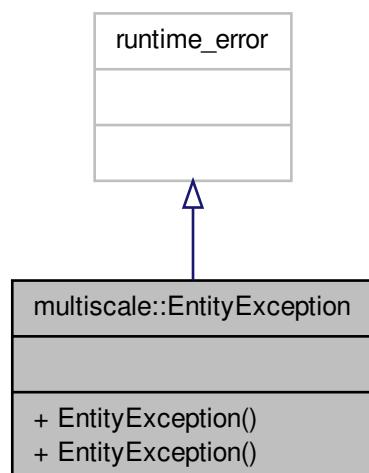
Exception class for the Entity instances.

```
#include <EntityException.hpp>
```

Inheritance diagram for multiscale::EntityException:



Collaboration diagram for multiscale::EntityException:



Public Member Functions

- [EntityException](#) (const string &msg)
- [EntityException](#) (const char *msg)

7.17.1 Detailed Description

Exception class for the Entity instances.

Definition at line 13 of file EntityException.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 multiscale::EntityException::EntityException (const string & msg) [inline]

Definition at line 17 of file EntityException.hpp.

7.17.2.2 multiscale::EntityException::EntityException (const char * msg) [inline]

Definition at line 18 of file EntityException.hpp.

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

- include/multiscale/exception/[EntityException.hpp](#)

7.18 multiscale::Geometry2D Class Reference

Two-dimensional geometric operations.

```
#include <Geometry2D.hpp>
```

Collaboration diagram for multiscale::Geometry2D:

multiscale::Geometry2D
+ isBetweenCoordinates() + angleOfLineWrtOxAxis() + isAngleBetween() + isOppositeAngleBetween() + isAngleBetweenNonReflex() + isOppositeAngleBetweenNonReflex() + oppositeAngle() + slopeOfLine() + distanceBtwPoints() + distanceFromPointToLine() + middlePoint() + orthogonalLineToAnotherLine EdgePoints() + areOnTheSameSideOfLine() + lineEquationDeterminedBy Points() + areIdenticalLines() + areIdenticalLines() + lineIntersection() + lineIntersection() + lineIntersection() + lineSegmentIntersection() + lineCircleIntersection() + lineSegmentCircleIntersection() + angleBtwPoints() + findPointsOnEdge() + minimumDistancePointIndex() + areaOfTriangle() + isPointOnLineSegment() + areEqualPoints() + areCollinear() - isPointOnEdge() - isBetweenCoordinates() - translate() - inverseTranslate() - lineCircleTwoIntersection Points() - lineCircleOneIntersection Point()

Public Member Functions

- template<typename T , typename U >
bool **isBetweenCoordinates** (T c, U c1, U c2)

Static Public Member Functions

- static double `angleOfLineWrtOxAxis` (const Point2f &a, const Point2f &b)

Get the angle of the line measured from the Ox axis in counterclockwise direction.
- static bool `isAngleBetween` (double angle1, double angle2, double angle3)

Check if angle1 lies between angles 2 and 3.
- static bool `isOppositeAngleBetween` (double angle1, double angle2, double angle3)

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between angles 2 and 3.
- static bool `isAngleBetweenNonReflex` (double angle1, double angle2, double angle3)

Check if angle1 lies between non reflex angle determined by angles 2 and 3.
- static bool `isOppositeAngleBetweenNonReflex` (double angle1, double angle2, double angle3)

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between non reflex angle determined by angles 2 and 3.
- static double `oppositeAngle` (double angle)

Return the angle opposite to the given angle.
- static bool `slopeOfLine` (const Point2f &a, const Point2f &b, double &slope)

Compute the slope of the line defined by points "a" and "b".
- static double `distanceBtwPoints` (const Point2f &a, const Point2f &b)

Compute the distance between two points.
- static double `distanceFromPointToLine` (const Point2f &a, const Point2f &linePointB, const Point2f &linePointC)

Compute the distance from a point "a" to a line specified by two points "B" and "C".
- static Point2f `middlePoint` (const Point2f &a, const Point2f &b)

Get the point in the middle of the segment determined by points "a" and "b".
- static void `orthogonalLineToAnotherLineEdgePoints` (const Point &a1, const Point &b1, Point &a2, Point &b2, int nrOfRows, int nrOfCols)

Find the points which are on the edge and on the line orthogonal to the line defined by 2 given points.
- static bool `areOnTheSameSideOfLine` (const Point2f &p1, const Point2f &p2, const Point2f &a, const Point2f &b)

Check if p1 and p2 are on the same side of the line determined by points a and b.
- static void `lineEquationDeterminedByPoints` (const Point2f &p, const Point2f &q, double &a, double &b, double &c)

Get the values of "a", "b" and "c" of the line equation $ax + by + c = 0$ knowing that point "p" and "q" are on the line.
- static bool `areIdenticalLines` (double a1, double b1, double c1, double a2, double b2, double c2)

Check if two lines are identical.
- static bool `areIdenticalLines` (const Point2f &a1, const Point2f &b1, const Point2f &a2, const Point2f &b2)

Check if two lines are identical.
- static bool `lineIntersection` (const Point2f &a1, const Point2f &b1, const Point2f &a2, const Point2f &b2, Point2f &intersection)

Determine the intersection point of two lines, if this point exists.
- static bool `lineIntersection` (const Point &a1, const Point &b1, const Point &a2, const Point &b2, Point &intersection)

Determine the intersection point of two lines, if this point exists.
- static bool `lineIntersection` (double a1, double b1, double c1, double a2, double b2, double c2, Point2f &intersection)

Determine the intersection point of two lines, if this point exists.
- static bool `lineSegmentIntersection` (const Point &a1, const Point &b1, const Point &a2, const Point &b2, Point &intersection)

Determine the intersection point of two line segments, if this point exists.
- static bool `lineCircleIntersection` (Point2f a, Point2f b, const Point2f &circleOrigin, double radius, vector<Point2f > &intersectionPoints)

Determine if a line and a circle intersect and return the intersection points if they exist.

- static bool `lineSegmentCircleIntersection` (const Point2f &a, const Point2f &b, const Point2f &circleOrigin, double radius, vector< Point2f > &intersectionPoints)

Determine if a line segment and a circle intersect and return the intersection points if they exist.
- static double `angleBtwPoints` (const Point2f &a, const Point2f &b, const Point2f &c)

Compute the angle between three points.
- static vector< Point2f > `findPointsOnEdge` (const vector< Point2f > &points, unsigned int nrOfRows, unsigned int nrOfCols)

Find the subset of points from the given set of points which lie on the edge.
- static unsigned int `minimumDistancePointIndex` (const vector< Point > &points, const Point2f &origin)

Get the index of the point which is the closest to the origin.
- static double `areaOfTriangle` (const Point2f &a, const Point2f &b, const Point2f &c)

Compute the area of a triangle defined by three points.
- static bool `isPointOnLineSegment` (const Point2f &point, const Point2f &lineSegmentStart, const Point2f &lineSegmentEnd)

Check if one point lies between two other points.
- static bool `areEqualPoints` (const Point2f &point1, const Point2f &point2)

Check if points point1 and point2 are equal or not.
- static bool `areCollinear` (const Point2f &point1, const Point2f &point2, const Point2f &point3)

Check if the three points are collinear.

Static Private Member Functions

- static bool `isPointOnEdge` (const Point2f &p, int nrOfRows, int nrOfCols)

Check if the given point is on the edge.
- template<typename T, typename U>
 static bool `isBetweenCoordinates` (T c, U c1, U c2)

Check if the coordinate c lies between c1 and c2.
- static void `translate` (Point2f &point, const Point2f &translation)

Translate a point by the given values.
- static void `inverseTranslate` (Point2f &point, const Point2f &translation)

Inverse translate a point by the given values.
- static void `lineCircleTwoIntersectionPoints` (const Point2f &circleOrigin, double A, double B, double C, double delta, vector< Point2f > &intersectionPoints)

Treat the case when the line and circle intersect in two points.
- static void `lineCircleOneIntersectionPoint` (const Point2f &circleOrigin, double A, double B, double C, double delta, vector< Point2f > &intersectionPoints)

Treat the case when the line and circle intersect in one point.

7.18.1 Detailed Description

Two-dimensional geometric operations.

Definition at line 19 of file Geometry2D.hpp.

7.18.2 Member Function Documentation

7.18.2.1 double Geometry2D::angleBtwPoints (const Point2f & a, const Point2f & b, const Point2f & c) [static]

Compute the angle between three points.

Compute the angle between the lines determined by points A, B and B, C

Parameters

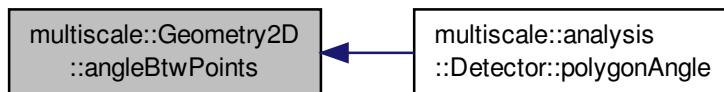
<i>a</i>	Point2f <i>a</i>
<i>b</i>	Point2f <i>b</i>
<i>c</i>	Point2f <i>c</i>

Definition at line 307 of file Geometry2D.cpp.

References PI.

Referenced by multiscale::analysis::Detector::polygonAngle().

Here is the caller graph for this function:



7.18.2.2 double Geometry2D::angleOfLineWrtOxAxis (const Point2f & *a*, const Point2f & *b*) [static]

Get the angle of the line measured from the Ox axis in counterclockwise direction.

The line is specified by points "a" and "b". The value of the angle is expressed in degrees.

Parameters

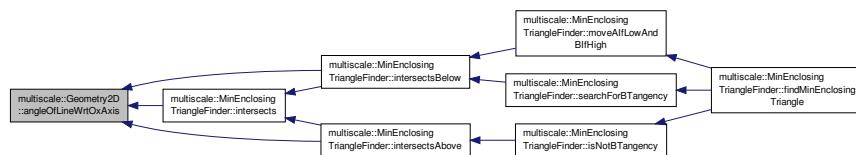
<i>a</i>	Point2f <i>a</i>
<i>b</i>	Point2f <i>b</i>

Definition at line 9 of file Geometry2D.cpp.

References PI.

Referenced by multiscale::MinEnclosingTriangleFinder::intersects(), multiscale::MinEnclosingTriangleFinder::intersectsAbove(), and multiscale::MinEnclosingTriangleFinder::intersectsBelow().

Here is the caller graph for this function:



7.18.2.3 double Geometry2D::areaOfTriangle (const Point2f & *a*, const Point2f & *b*, const Point2f & *c*) [static]

Compute the area of a triangle defined by three points.

The area is computed using the determinant method. An example is presented at <http://demonstrations.-wolfram.com/TheAreaOfATriangleUsingADeterminant/> (Last access: 10.07.2013)

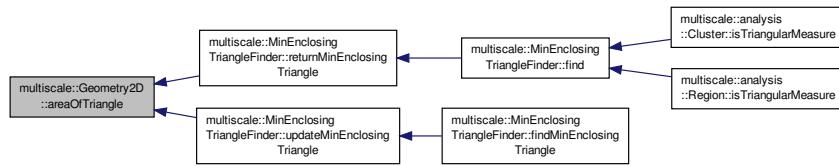
Parameters

<i>a</i>	Point2f a
<i>b</i>	Point2f b
<i>c</i>	Point2f c

Definition at line 352 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::returnMinEnclosingTriangle(), and multiscale::MinEnclosingTriangleFinder::updateMinEnclosingTriangle().

Here is the caller graph for this function:



7.18.2.4 bool Geometry2D::areCollinear (const Point2f & *point1*, const Point2f & *point2*, const Point2f & *point3*) [static]

Check if the three points are collinear.

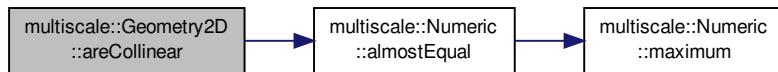
Parameters

<i>point1</i>	Point 1
<i>point2</i>	Point 2
<i>point3</i>	Point 3

Definition at line 374 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Here is the call graph for this function:



7.18.2.5 bool Geometry2D::areEqualPoints (const Point2f & *point1*, const Point2f & *point2*) [static]

Check if points point1 and point2 are equal or not.

Parameters

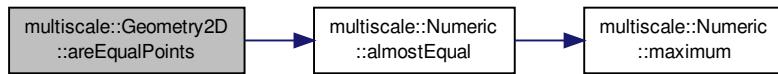
<i>point1</i>	One point
<i>point2</i>	The other point

Definition at line 370 of file Geometry2D.cpp.

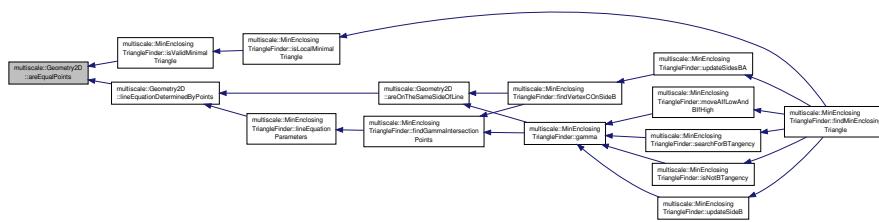
References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and lineEquationDeterminedByPoints().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.6 bool Geometry2D::areIdenticalLines (double a1, double b1, double c1, double a2, double b2, double c2) [static]

Check if two lines are identical.

Lines are be specified in the following form: $A_1x + B_1x = C_1$ $A_2x + B_2x = C_2$

If $(A_1/A_2) == (B_1/B_2) == (C_1/C_2)$, then the lines are identical else they are not

Parameters

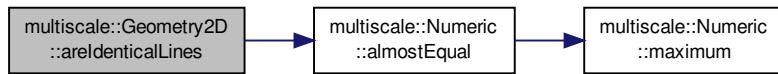
<i>a1</i>	A1
<i>b1</i>	B1
<i>c1</i>	C1
<i>a2</i>	A2
<i>b2</i>	B2
<i>c2</i>	C2

Definition at line 161 of file Geometry2D.cpp.

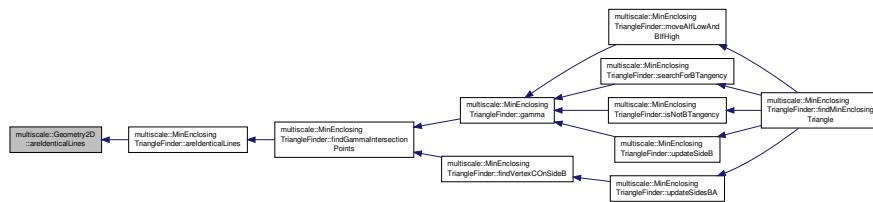
References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::areIdenticalLines().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.7 bool Geometry2D::areIdenticalLines (const Point2f & a1, const Point2f & b1, const Point2f & a2, const Point2f & b2) [static]

Check if two lines are identical.

The lines are specified by a pair of points each. If they are identical, then the function returns true, else it returns false.

Lines can be specified in the following form: $A_1x + B_1x = C_1$ $A_2x + B_2x = C_2$

If $(A_1/A_2) == (B_1/B_2) == (C_1/C_2)$, then the lines are identical else they are not

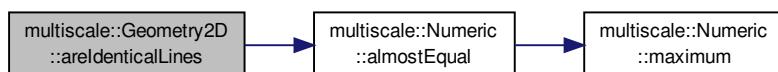
Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line

Definition at line 172 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Here is the call graph for this function:



7.18.2.8 `bool Geometry2D::areOnTheSameSideOfLine (const Point2f & p1, const Point2f & p2, const Point2f & a, const Point2f & b) [static]`

Check if p1 and p2 are on the same side of the line determined by points a and b.

Parameters

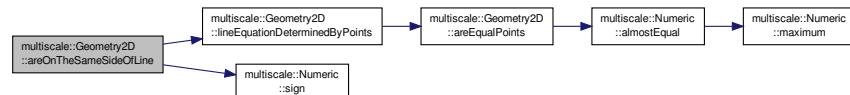
<i>p1</i>	Point p1
<i>p2</i>	Point p2
<i>a</i>	First point for determining line
<i>b</i>	Second point for determining line

Definition at line 141 of file Geometry2D.cpp.

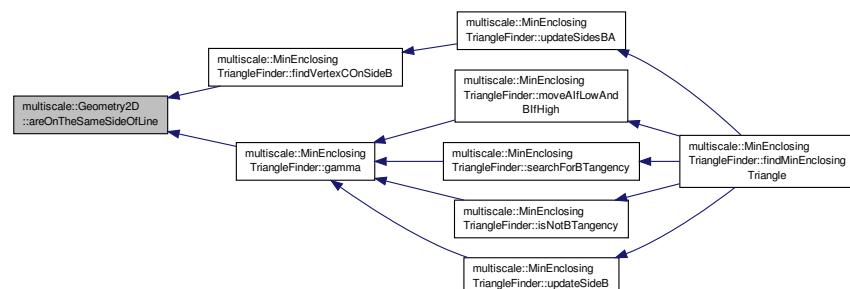
References `lineEquationDeterminedByPoints()`, and `multiscale::Numeric::sign()`.

Referenced by `multiscale::MinEnclosingTriangleFinder::findVertexCOnSideB()`, and `multiscale::MinEnclosingTriangleFinder::gamma()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.9 `double Geometry2D::distanceBtwPoints (const Point2f & a, const Point2f & b) [static]`

Compute the distance between two points.

Compute the Euclidean distance between two points

Parameters

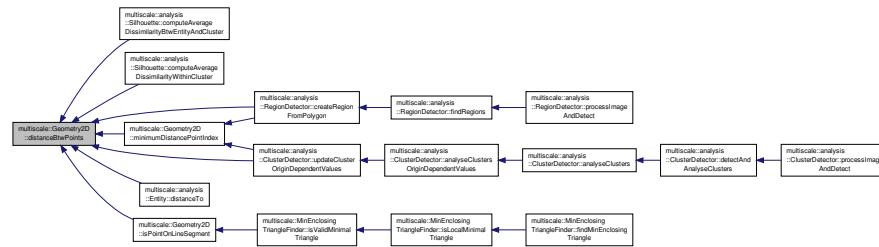
<i>a</i>	Point2f a
<i>b</i>	Point2f b

Definition at line 69 of file Geometry2D.cpp.

Referenced by `multiscale::analysis::Silhouette::computeAverageDissimilarityBtwEntityAndCluster()`, `multiscale::analysis::Silhouette::computeAverageDissimilarityWithinCluster()`, `multiscale::analysis::RegionDetector::create-`

RegionFromPolygon(), multiscale::analysis::Entity::distanceTo(), isPointOnLineSegment(), minimumDistancePointIndex(), and multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues().

Here is the caller graph for this function:



7.18.2.10 double Geometry2D::distanceFromPointToLine (const Point2f & a, const Point2f & linePointB, const Point2f & linePointC) [static]

Compute the distance from a point "a" to a line specified by two points "B" and "C".

Formula used:

$$| (x_c - x_b) * (y_b - y_a) - (x_b - x_a) * (y_c - y_b) |$$

$$d = \sqrt{((x_c - x_b)^2 + (y_c - y_b)^2)}$$

Reference: <http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>

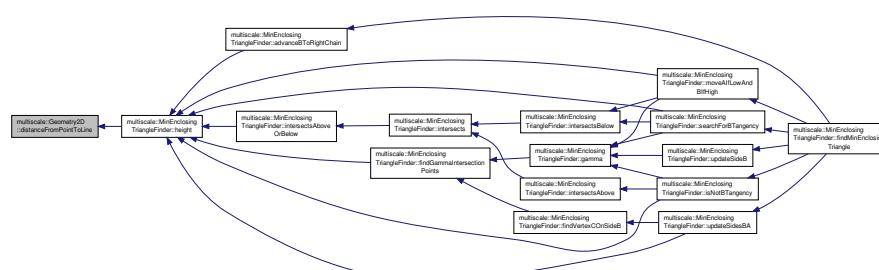
Parameters

<i>a</i>	Point2f from which the distance is measures
<i>linePointB</i>	One of the points determining the line
<i>linePointC</i>	One of the points determining the line

Definition at line 76 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::height().

Here is the caller graph for this function:



7.18.2.11 vector< Point2f > Geometry2D::findPointsOnEdge (const vector< Point2f > & points, unsigned int nrOfRows, unsigned int nrOfCols) [static]

Find the subset of points from the given set of points which lie on the edge.

A point "p" is considered to be on the edge if: $((p.x == 1) \&\& (p.y > 1) \&\& (p.y < \text{nrOfCols}))$ OR $((p.x == \text{nrOfRows}) \&\& (p.y > 1) \&\& (p.y < \text{nrOfCols}))$ OR $((p.y == 1) \&\& (p.x > 1) \&\& (p.x < \text{nrOfRows}))$ OR $((p.y == \text{nrOfCols}) \&\& (p.x > 1) \&\& (p.x < \text{nrOfRows}))$

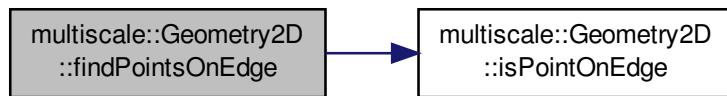
Parameters

<i>points</i>	The set of points
<i>nrOfRows</i>	The number of rows
<i>nrOfCols</i>	The number of columns

Definition at line 319 of file Geometry2D.cpp.

References `isPointOnEdge()`.

Here is the call graph for this function:



7.18.2.12 void Geometry2D::inverseTranslate (Point2f & point, const Point2f & translation) [static], [private]

Inverse translate a point by the given values.

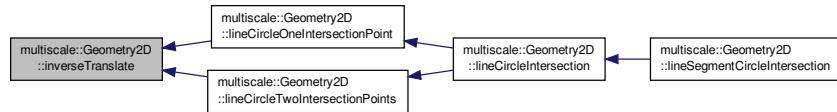
Parameters

<i>point</i>	The point
<i>translation</i>	Translation values

Definition at line 400 of file Geometry2D.cpp.

Referenced by `lineCircleOneIntersectionPoint()`, and `lineCircleTwoIntersectionPoints()`.

Here is the caller graph for this function:



7.18.2.13 bool Geometry2D::isAngleBetween (double angle1, double angle2, double angle3) [static]

Check if angle1 lies between angles 2 and 3.

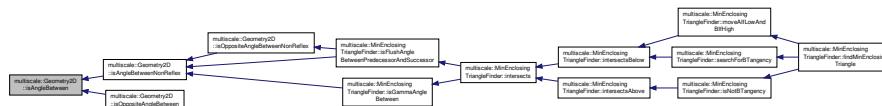
Parameters

<i>angle1</i>	The angle which lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 19 of file Geometry2D.cpp.

Referenced by `isAngleBetweenNonReflex()`, and `isOppositeAngleBetween()`.

Here is the caller graph for this function:

7.18.2.14 bool `Geometry2D::isAngleBetweenNonReflex (double angle1, double angle2, double angle3) [static]`

Check if angle1 lies between non reflex angle determined by angles 2 and 3.

Parameters

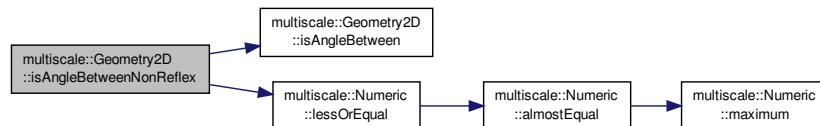
<i>angle1</i>	The angle which lies between angle2 and angle3 or not
<i>angle2</i>	One of the boundary angles
<i>angle3</i>	The other boundary angle

Definition at line 33 of file Geometry2D.cpp.

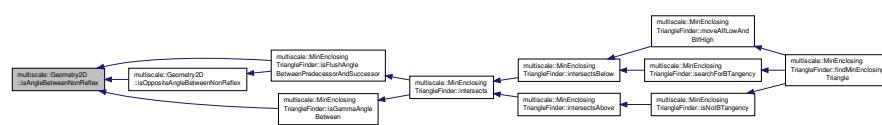
References `isAngleBetween()`, and `multiscale::Numeric::lessOrEqual()`.

Referenced by `multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor()`, `multiscale::MinEnclosingTriangleFinder::isGammaAngleBetween()`, and `isOppositeAngleBetweenNonReflex()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.15 `template<typename T, typename U> bool multiscale::Geometry2D::isBetweenCoordinates (T c, U c1, U c2)`

Definition at line 391 of file Geometry2D.cpp.

7.18.2.16 `template<typename T, typename U> static bool multiscale::Geometry2D::isBetweenCoordinates (T c, U c1, U c2) [static], [private]`

Check if the coordinate c lies between c1 and c2.

Parameters

<code>c</code>	Coordinate c
<code>c1</code>	Coordinate c1
<code>c2</code>	Coordinate c2

7.18.2.17 `bool Geometry2D::isOppositeAngleBetween (double angle1, double angle2, double angle3) [static]`

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between angles 2 and 3.

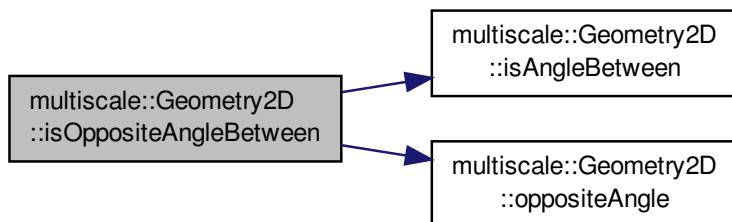
Parameters

<code>angle1</code>	The angle for which the opposite angle lies between angle2 and angle3 or not
<code>angle2</code>	One of the boundary angles
<code>angle3</code>	The other boundary angle

Definition at line 27 of file Geometry2D.cpp.

References `isAngleBetween()`, and `oppositeAngle()`.

Here is the call graph for this function:



7.18.2.18 `bool Geometry2D::isOppositeAngleBetweenNonReflex (double angle1, double angle2, double angle3) [static]`

Check if the opposite of angle1, ((angle1 + 180) % 360), lies between non reflex angle determined by angles 2 and 3.

Parameters

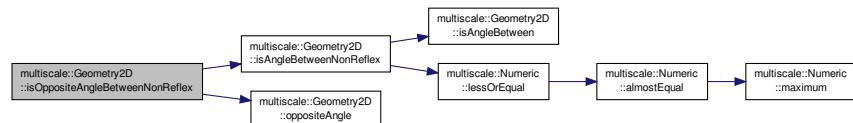
<code>angle1</code>	The angle which lies between angle2 and angle3 or not
<code>angle2</code>	One of the boundary angles
<code>angle3</code>	The other boundary angle

Definition at line 45 of file Geometry2D.cpp.

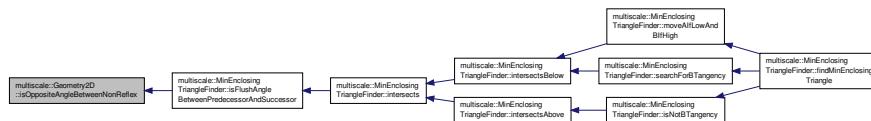
References `isAngleBetweenNonReflex()`, and `oppositeAngle()`.

Referenced by `multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.19 bool Geometry2D::isPointOnEdge (const Point2f & p, int nrOfRows, int nrOfCols) [static], [private]

Check if the given point is on the edge.

A point "p" is considered to be on the edge if: $((p.x == 1) \&\& (p.y > 1) \&\& (p.y < nrOfCols))$ OR $((p.x == nrOfRows) \&\& (p.y > 1) \&\& (p.y < nrOfCols))$ OR $((p.y == 1) \&\& (p.x > 1) \&\& (p.x < nrOfRows))$ OR $((p.y == nrOfCols) \&\& (p.x > 1) \&\& (p.x < nrOfRows))$

Parameters

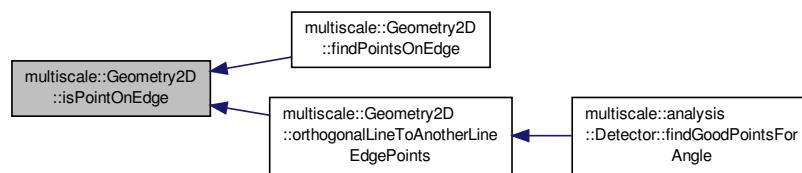
<code>p</code>	Point2f p
<code>nrOfRows</code>	The number of rows
<code>nrOfCols</code>	The number of columns

Definition at line 381 of file Geometry2D.cpp.

References `MATRIX_START_INDEX`.

Referenced by `findPointsOnEdge()`, and `orthogonalLineToAnotherLineEdgePoints()`.

Here is the caller graph for this function:



7.18.2.20 `bool Geometry2D::isPointOnLineSegment (const Point2f & point, const Point2f & lineSegmentStart, const Point2f & lineSegmentEnd) [static]`

Check if one point lies between two other points.

Parameters

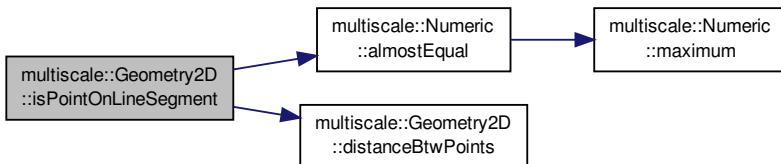
<i>point</i>	Point lying possibly outside the line segment
<i>lineSegment- Start</i>	First point determining the line segment
<i>lineSegmentEnd</i>	Second point determining the line segment

Definition at line 361 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual(), and distanceBtwPoints().

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.21 `bool Geometry2D::lineCircleIntersection (Point2f a, Point2f b, const Point2f & circleOrigin, double radius, vector< Point2f > & intersectionPoints) [static]`

Determine if a line and a circle intersect and return the intersection points if they exist.

We translate all the points such that the circle origin coincides with the origin of the coordinate system. When returning the results, the intersection points are inverse translated.

Parameters

<i>a</i>	First point for determining the line
<i>b</i>	Second point for determining the line
<i>circleOrigin</i>	Origin of the circle
<i>radius</i>	Radius of the circle
<i>intersection- Points</i>	The intersection points between the circle and the line

< Two intersection points

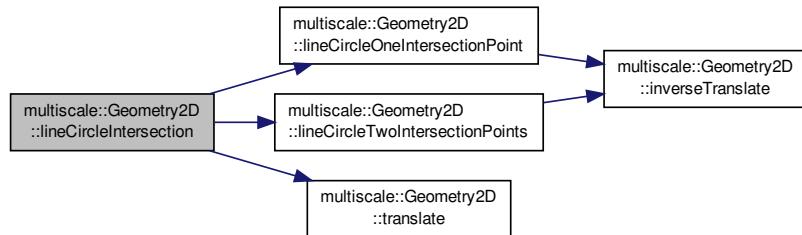
< One intersection point

Definition at line 259 of file Geometry2D.cpp.

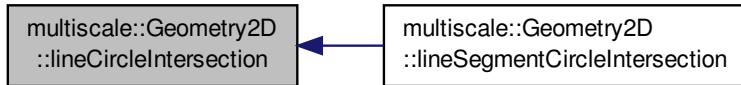
References lineCircleOneIntersectionPoint(), lineCircleTwoIntersectionPoints(), and translate().

Referenced by lineSegmentCircleIntersection().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.22 void Geometry2D::lineCircleOneIntersectionPoint (const Point2f & circleOrigin, double A, double B, double C, double delta, vector< Point2f > & intersectionPoints) [static], [private]

Treat the case when the line and circle intersect in one point.

Parameters

<i>circleOrigin</i>	Origin of the circle
<i>A</i>	$y_2 - y_1$
<i>B</i>	$x_1 - x_2$
<i>C</i>	$A \cdot x_1 + B \cdot y_1$
<i>delta</i>	$(4 * B^2 * C^2) - (4 * (A^2 + B^2) * (C^2 - (R^2 * A^2)))$
<i>intersectionPoints</i>	Intersection points

Definition at line 423 of file Geometry2D.cpp.

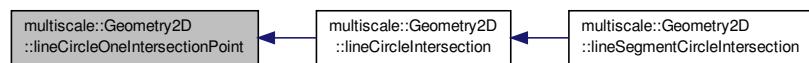
References inverseTranslate().

Referenced by lineCircleIntersection().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.23 void Geometry2D::lineCircleTwoIntersectionPoints (const Point2f & circleOrigin, double A, double B, double C, double delta, vector< Point2f > & intersectionPoints) [static], [private]

Treat the case when the line and circle intersect in two points.

Parameters

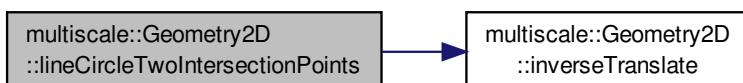
<i>circleOrigin</i>	Origin of the circle
<i>A</i>	$y_2 - y_1$
<i>B</i>	$x_1 - x_2$
<i>C</i>	$A \cdot x_1 + B \cdot y_1$
<i>delta</i>	$(4 * B^2 * C^2) - (4 * (A^2 + B^2)) * (C^2 - (R^2 * A^2))$
<i>intersection- Points</i>	Intersection points

Definition at line 405 of file Geometry2D.cpp.

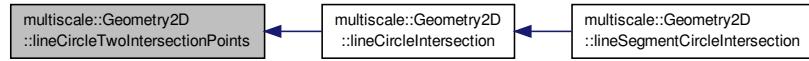
References inverseTranslate().

Referenced by lineCircleIntersection().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.24 void Geometry2D::lineEquationDeterminedByPoints (const Point2f & p, const Point2f & q, double & a, double & b, double & c) [static]

Get the values of "a", "b" and "c" of the line equation $ax + by + c = 0$ knowing that point "p" and "q" are on the line.

$$a = q.y - p.y \quad b = p.x - q.x \quad c = -(p.x * a) - (p.y * b)$$

Parameters

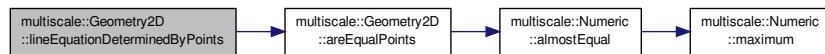
<i>p</i>	Point2f p
<i>q</i>	Point2f q
<i>a</i>	Parameter "a" from the line equation
<i>b</i>	Parameter "b" from the line equation
<i>c</i>	Parameter "c" from the line equation

Definition at line 153 of file Geometry2D.cpp.

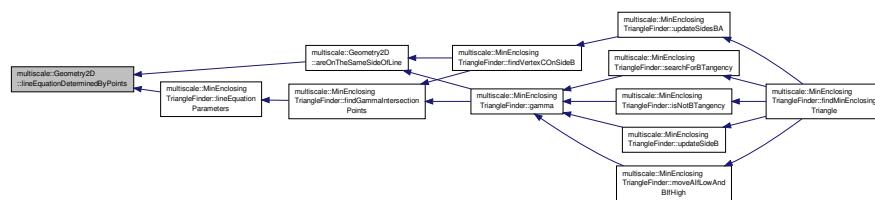
References areEqualPoints().

Referenced by areOnTheSameSideOfLine(), and multiscale::MinEnclosingTriangleFinder::lineEquationParameters().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.25 bool Geometry2D::lineIntersection (const Point2f & a1, const Point2f & b1, const Point2f & a2, const Point2f & b2, Point2f & intersection) [static]

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified by a pair of points each. If they intersect, then the function returns true, else it returns false.

Lines can be specified in the following form: $A_1x + B_1x = C_1$ $A_2x + B_2x = C_2$

If $\det (= A_1B_2 - A_2B_1) == 0$, then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with $P(x, y)$ where: $x = (C_1B_2 - C_2B_1) / (\det)$ $y = (C_2A_1 - C_1A_2) / (\det)$

Parameters

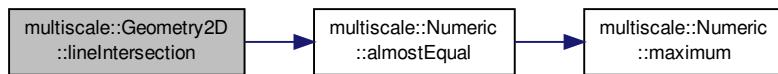
<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 212 of file Geometry2D.cpp.

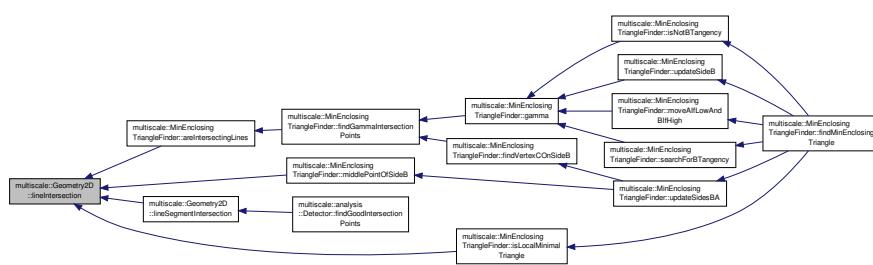
References multiscale::Numeric::almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::areIntersectingLines(), multiscale::MinEnclosingTriangleFinder::isLocalMinimalTriangle(), lineSegmentIntersection(), and multiscale::MinEnclosingTriangleFinder::middlePointOfSideB().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.26 bool Geometry2D::lineIntersection (const Point & *a1*, const Point & *b1*, const Point & *a2*, const Point & *b2*, Point & *intersection*) [static]

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified by a pair of points each. If they intersect, then the function returns true, else it returns false.

Lines can be specified in the following form: $A_1x + B_1x = C_1$ $A_2x + B_2x = C_2$

If $\det (= A_1xB_2 - A_2xB_1) == 0$, then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with $P(x, y)$ where: $x = (C_1xB_2 - C_2xB_1) / (\det)$ $y = (C_2xA_1 - C_1xA_2) / (\det)$

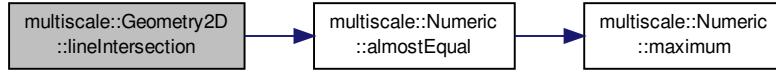
Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 191 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Here is the call graph for this function:



7.18.2.27 `bool Geometry2D::lineIntersection (double a1, double b1, double c1, double a2, double b2, double c2, Point2f & intersection) [static]`

Determine the intersection point of two lines, if this point exists.

Two lines intersect if they are not parallel (Parallel lines intersect at +/- infinity, but we do not consider this case here).

The lines are specified in the following form: $A_1x + B_1x = C_1$ $A_2x + B_2x = C_2$

If $\det (= A_1xB_2 - A_2xB_1) == 0$, then lines are parallel else they intersect

If they intersect, then let us denote the intersection point with $P(x, y)$ where: $x = (C_1xB_2 - C_2xB_1) / (\det)$ $y = (C_2xA_1 - C_1xA_2) / (\det)$

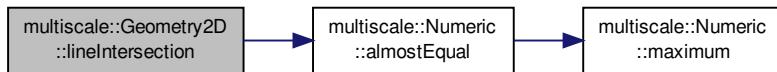
Parameters

<i>a1</i>	A1
<i>b1</i>	B1
<i>c1</i>	C1
<i>a2</i>	A2
<i>b2</i>	B2
<i>c2</i>	C2
<i>intersection</i>	The intersection point, if this point exists

Definition at line 233 of file Geometry2D.cpp.

References multiscale::Numeric::almostEqual().

Here is the call graph for this function:



7.18.2.28 `bool Geometry2D::lineSegmentCircleIntersection (const Point2f & a, const Point2f & b, const Point2f & circleOrigin, double radius, vector< Point2f > & intersectionPoints) [static]`

Determine if a line segment and a circle intersect and return the intersection points if they exist.

We translate all the points such that the circle origin coincides with the origin of the coordinate system. When returning the results, the intersection points are inverse translated.

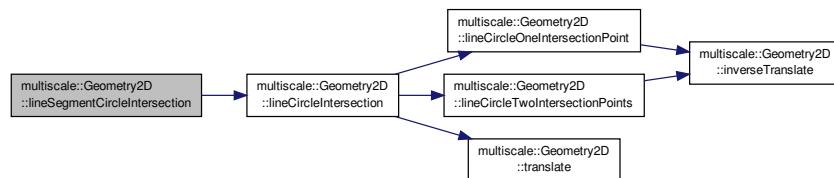
Parameters

<i>a</i>	First point for determining the line
<i>b</i>	Second point for determining the line
<i>circleOrigin</i>	Origin of the circle
<i>radius</i>	Radius of the circle
<i>intersection- Points</i>	The intersection points between the circle and the line

Definition at line 288 of file Geometry2D.cpp.

References `lineCircleIntersection()`.

Here is the call graph for this function:



7.18.2.29 `bool Geometry2D::lineSegmentIntersection (const Point & a1, const Point & b1, const Point & a2, const Point & b2, Point & intersection) [static]`

Determine the intersection point of two line segments, if this point exists.

Find the intersection point of the lines, if this point exists. Let us assume that this point exists and let us denote it with $P(x, y)$. Then, in order for the point to be the intersection of the segments and not of the lines, we have to verify the following conditions:

1. $\min(a1.x, b1.x) \leq x \leq \max(a1.x, b1.x) - x$ coordinate is valid for first line segment
2. $\min(a2.x, b2.x) \leq x \leq \max(a2.x, b2.x) - x$ coordinate is valid for second line segment

3. $\min(a1.y, b1.y) \leq y \leq \max(a1.y, b1.y)$ – y coordinate is valid for first line segment
4. $\min(a2.y, b2.y) \leq y \leq \max(a2.y, b2.y)$ – y coordinate is valid for second line segment

Parameters

<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>intersection</i>	The intersection point, if this point exists

Definition at line 246 of file Geometry2D.cpp.

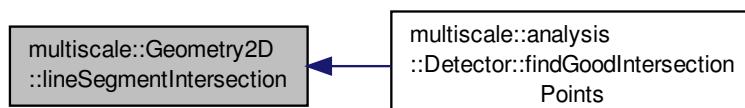
References lineIntersection().

Referenced by multiscale::analysis::Detector::findGoodIntersectionPoints().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.30 Point2f Geometry2D::middlePoint (const Point2f & a, const Point2f & b) [static]

Get the point in the middle of the segment determined by points "a" and "b".

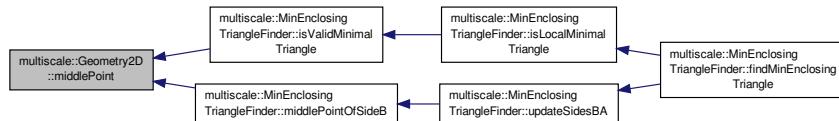
Parameters

<i>a</i>	Point2f a
<i>b</i>	Point2f b

Definition at line 88 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and multiscale::MinEnclosingTriangleFinder::middlePointOfSideB().

Here is the caller graph for this function:



7.18.2.31 `unsigned int Geometry2D::minimumDistancePointIndex (const vector< Point > & points, const Point2f & origin) [static]`

Get the index of the point which is the closest to the origin.

Get the index of the point P from the given set of points, such that for any point A from the set of points $\text{dist}(A, \text{origin}) \geq \text{dist}(P, \text{origin})$.

Parameters

<code>points</code>	The set of points
<code>origin</code>	The origin

Definition at line 333 of file Geometry2D.cpp.

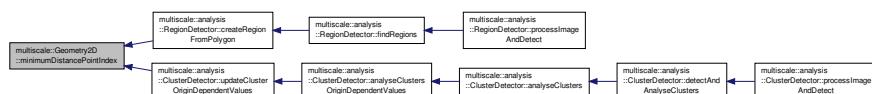
References `distanceBtwPoints()`.

Referenced by `multiscale::analysis::RegionDetector::createRegionFromPolygon()`, and `multiscale::analysis::ClusterDetector::updateClusterOriginDependentValues()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.32 `double Geometry2D::oppositeAngle (double angle) [static]`

Return the angle opposite to the given angle.

`if (angle < 180) then return (angle + 180); else return (angle - 180); endif`

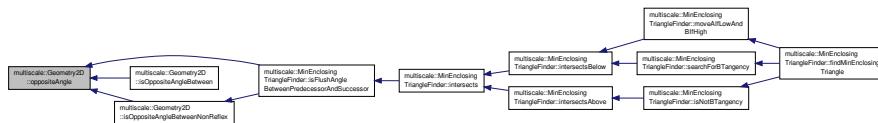
Parameters

<i>angle</i>	Angle
--------------	-------

Definition at line 51 of file Geometry2D.cpp.

Referenced by multiscale::MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor(), isOppositeAngleBetween(), and isOppositeAngleBetweenNonReflex().

Here is the caller graph for this function:



7.18.2.33 void Geometry2D::orthogonalLineToAnotherLineEdgePoints (const Point & a1, const Point & b1, Point & a2, Point & b2, int nrOfRows, int nrOfCols) [static]

Find the points which are on the edge and on the line orthogonal to the line defined by 2 given points.

Parameters

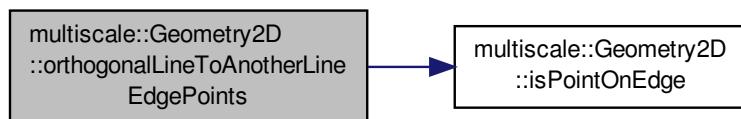
<i>a1</i>	First point for determining the first line
<i>b1</i>	Second point for determining the first line
<i>a2</i>	First point for determining the second line
<i>b2</i>	Second point for determining the second line
<i>nrOfRows</i>	Maximum number of rows in the considered matrix
<i>nrOfCols</i>	Maximum number of columns in the considered matrix

Definition at line 95 of file Geometry2D.cpp.

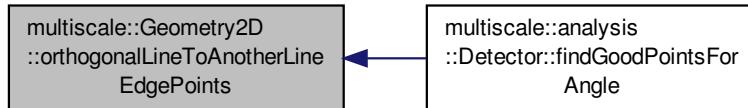
References isPointOnEdge().

Referenced by multiscale::analysis::Detector::findGoodPointsForAngle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.2.34 bool Geometry2D::slopeOfLine (const Point2f & a, const Point2f & b, double & slope) [static]

Compute the slope of the line defined by points "a" and "b".

Returns true if the slope of the line can be computed and false otherwise.

Parameters

<i>a</i>	Point2f a
<i>b</i>	Point2f b
<i>slope</i>	Slope of the line if it is different from (+/-)infinity

Definition at line 56 of file Geometry2D.cpp.

7.18.2.35 void Geometry2D::translate (Point2f & point, const Point2f & translation) [static], [private]

Translate a point by the given values.

Parameters

<i>point</i>	The point
<i>translation</i>	Translation values

Definition at line 395 of file Geometry2D.cpp.

Referenced by lineCircleIntersection().

Here is the caller graph for this function:



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

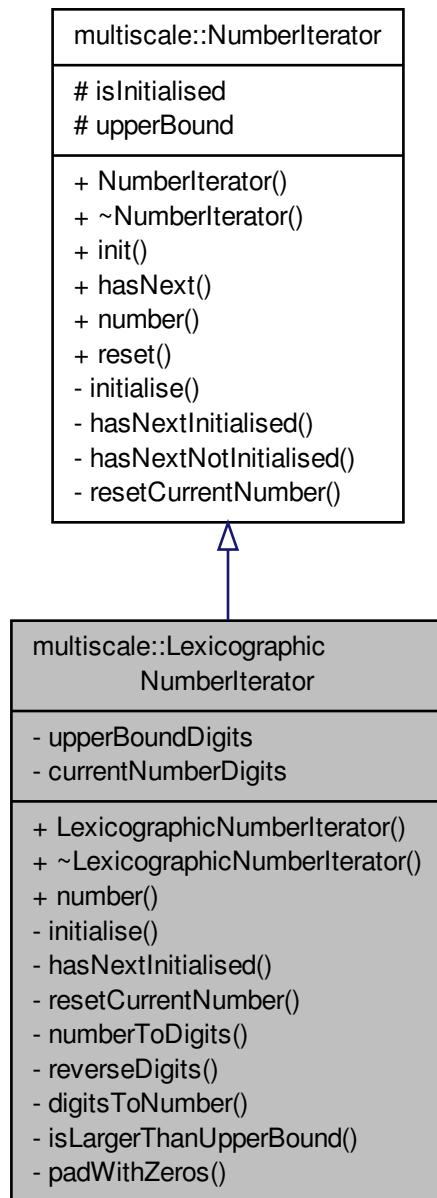
- include/multiscale/util/[Geometry2D.hpp](#)
- src/multiscale/util/[Geometry2D.cpp](#)

7.19 multiscale::LexicographicNumberIterator Class Reference

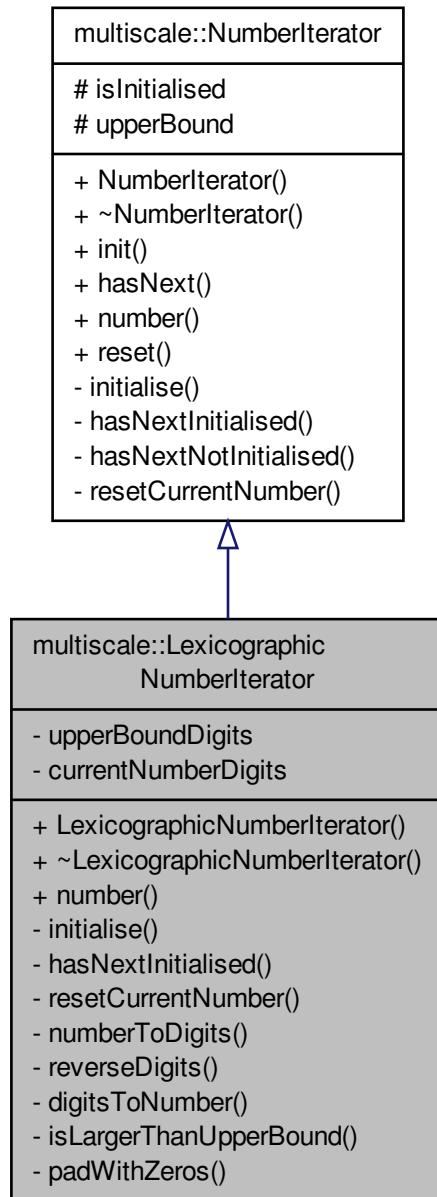
Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "—" .

```
#include <LexicographicNumberIterator.hpp>
```

Inheritance diagram for multiscale::LexicographicNumberIterator:



Collaboration diagram for multiscale::LexicographicNumberIterator:



Public Member Functions

- `LexicographicNumberIterator (unsigned int upperBound)`
- `~LexicographicNumberIterator ()`
- `unsigned int number ()`

Get the number pointed by the iterator.

Private Member Functions

- void [initialise \(\)](#)
Initialise the vectors of digits.
- bool [hasNextInitialised \(\)](#)
Check if there is a next number when in initialised state.
- void [resetCurrentNumber \(\)](#)
Reset the digits of the current number to the initial value.
- void [numberToDigits \(unsigned int number, vector< unsigned char > &digits\)](#)
Convert the number to a vector of digits.
- void [reverseDigits \(vector< unsigned char > &digits\)](#)
Reverse the order of the digits.
- unsigned int [digitsToNumber \(vector< unsigned char > &digits\)](#)
Convert the vector of digits to the number they represent.
- bool [isLargerThanUpperBound \(unsigned char lastDigit\)](#)
Check if the current number with the provided last digit is greater than the upper bound.
- void [padWithZeros \(\)](#)
Pad the current number with zeros.

Private Attributes

- vector< unsigned char > [upperBoundDigits](#)
- vector< unsigned char > [currentNumberDigits](#)

Additional Inherited Members

7.19.1 Detailed Description

Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an "—" .

Definition at line 14 of file LexicographicNumberIterator.hpp.

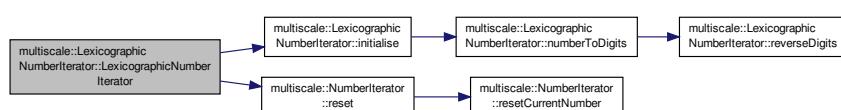
7.19.2 Constructor & Destructor Documentation

7.19.2.1 LexicographicNumberIterator::LexicographicNumberIterator (unsigned int *upperBound*)

Definition at line 6 of file LexicographicNumberIterator.cpp.

References [initialise\(\)](#), and [multiscale::NumberIterator::reset\(\)](#).

Here is the call graph for this function:



7.19.2.2 LexicographicNumberIterator::~LexicographicNumberIterator()

Definition at line 11 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and upperBoundDigits.

7.19.3 Member Function Documentation

7.19.3.1 unsigned int LexicographicNumberIterator::digitsToNumber(vector< unsigned char > & digits) [private]

Convert the vector of digits to the number they represent.

Parameters

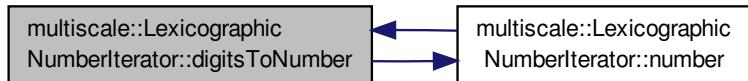
<i>digits</i>	The digits
---------------	------------

Definition at line 74 of file LexicographicNumberIterator.cpp.

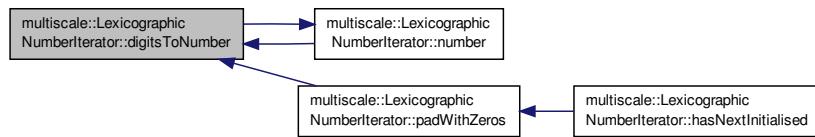
References number().

Referenced by number(), and padWithZeros().

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.3.2 bool LexicographicNumberIterator::hasNextInitialised() [private], [virtual]

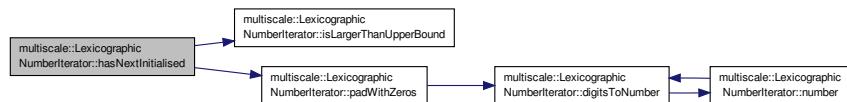
Check if there is a next number when in initialised state.

Implements [multiscale::NumberIterator](#).

Definition at line 26 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, isLargerThanUpperBound(), and padWithZeros().

Here is the call graph for this function:



7.19.3.3 void LexicographicNumberIterator::initialise() [private], [virtual]

Initialise the vectors of digits.

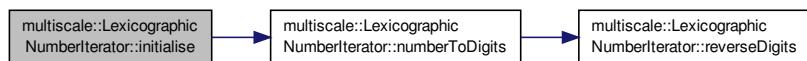
Implements [multiscale::NumberIterator](#).

Definition at line 20 of file LexicographicNumberIterator.cpp.

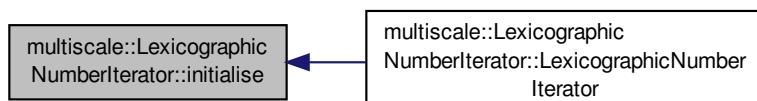
References `currentNumberDigits`, `numberToDigits()`, `multiscale::NumberIterator::upperBound`, and `upperBound-Digits`.

Referenced by `LexicographicNumberIterator()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.3.4 bool LexicographicNumberIterator::isLargerThanUpperBound(unsigned char lastDigit) [private]

Check if the current number with the provided last digit is greater than the upper bound.

Check if the current number is greater than the upper bound when replacing the last digit of the current number with the provided digit

Parameters

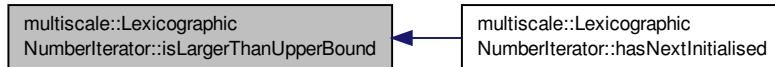
<code>lastDigit</code>	The last digit
------------------------	----------------

Definition at line 86 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and upperBoundDigits.

Referenced by hasNextInitialised().

Here is the caller graph for this function:



7.19.3.5 unsigned int LexicographicNumberIterator::number () [virtual]

Get the number pointed by the iterator.

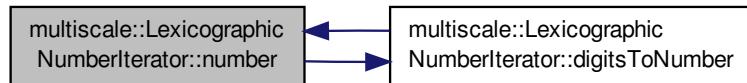
Implements [multiscale::NumberIterator](#).

Definition at line 16 of file LexicographicNumberIterator.cpp.

References currentNumberDigits, and digitsToNumber().

Referenced by digitsToNumber().

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.3.6 void LexicographicNumberIterator::numberToDigits (unsigned int *number*, vector< unsigned char > & *digits*) [private]

Convert the number to a vector of digits.

Parameters

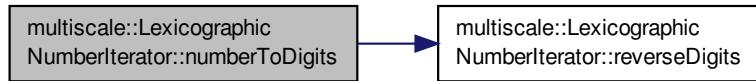
<i>number</i>	The number
<i>digits</i>	The digits of the number

Definition at line 53 of file LexicographicNumberIterator.cpp.

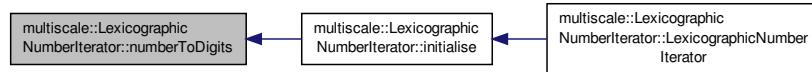
References `reverseDigits()`.

Referenced by `initialise()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.3.7 void LexicographicNumberIterator::padWithZeros () [private]

Pad the current number with zeros.

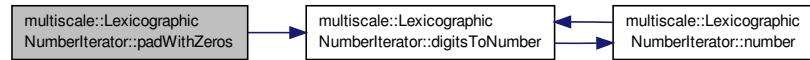
Pad the current number with the maximum number of zeros such that it does not become larger than the upper bound

Definition at line 107 of file `LexicographicNumberIterator.cpp`.

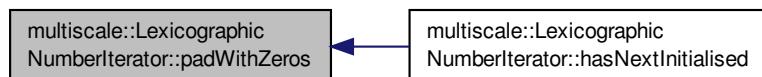
References `currentNumberDigits`, `digitsToNumber()`, `multiscale::NumberIterator::upperBound`, and `upperBoundDigits`.

Referenced by `hasNextInitialised()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.19.3.8 void LexicographicNumberIterator::resetCurrentNumber() [private], [virtual]

Reset the digits of the current number to the initial value.

Implements [multiscale::NumberIterator](#).

Definition at line 42 of file LexicographicNumberIterator.cpp.

References `currentNumberDigits`, and `upperBoundDigits`.

7.19.3.9 void LexicographicNumberIterator::reverseDigits(vector<unsigned char> & digits) [private]

Reverse the order of the digits.

Reverse the order of the digits such that the first one is swapped with the last one, the second one is swapped with the last but one and so on.

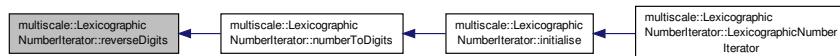
Parameters

<code>digits</code>	The digits
---------------------	------------

Definition at line 63 of file LexicographicNumberIterator.cpp.

Referenced by `numberToDigits()`.

Here is the caller graph for this function:



7.19.4 Member Data Documentation

7.19.4.1 vector<unsigned char> multiscale::LexicographicNumberIterator::currentNumberDigits [private]

The digits of the number to which the iterator points

Definition at line 19 of file LexicographicNumberIterator.hpp.

Referenced by `hasNextInitialised()`, `initialise()`, `isLargerThanUpperBound()`, `number()`, `padWithZeros()`, `resetCurrentNumber()`, and `~LexicographicNumberIterator()`.

7.19.4.2 vector<unsigned char> multiscale::LexicographicNumberIterator::upperBoundDigits [private]

The digits of the upper bound

Definition at line 18 of file LexicographicNumberIterator.hpp.

Referenced by `initialise()`, `isLargerThanUpperBound()`, `padWithZeros()`, `resetCurrentNumber()`, and `~LexicographicNumberIterator()`.

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

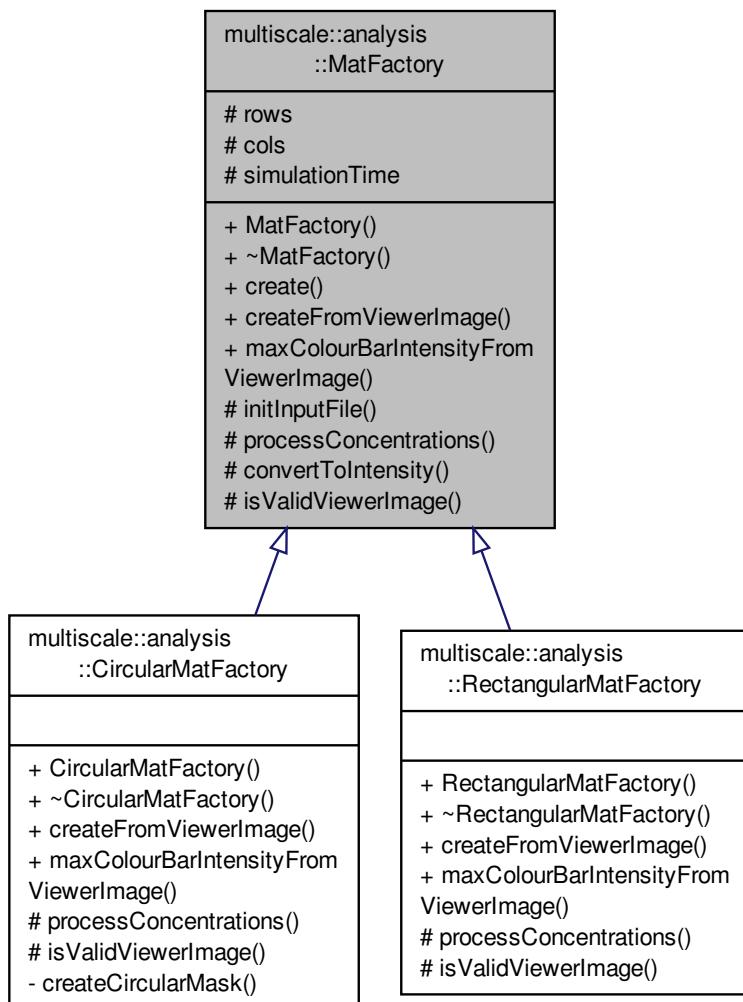
- include/multiscale/util/iterator/[LexicographicNumberIterator.hpp](#)
- src/multiscale/util/iterator/[LexicographicNumberIterator.cpp](#)

7.20 multiscale::analysis::MatFactory Class Reference

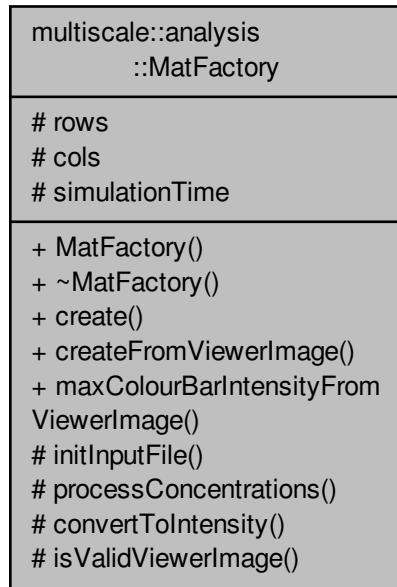
Class for creating a Mat object.

```
#include <MatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::MatFactory:



Collaboration diagram for multiscale::analysis::MatFactory:



Public Member Functions

- `MatFactory ()`
- virtual `~MatFactory ()`
- Mat `create (const string &inputFile)`

Create a Mat object from the input file.
- virtual Mat `createFromViewerImage (const string &inputFile)=0`

Create a Mat object from the image file obtained from Rectangular/CircularGeometryViewer.
- virtual double `maxColourBarIntensityFromViewerImage (const string &inputFile)=0`

Get the maximum grayscale intensity of the colour bar in the image.

Protected Member Functions

- void `initInputFile (ifstream &fin, const string &inputFile)`

Initialise the input file.
- virtual unsigned char * `processConcentrations (ifstream &fin)=0`

Process concentrations from file.
- unsigned char `convertToIntensity (double concentration)`

Convert concentration to intensity.
- virtual bool `isValidViewerImage (const Mat &image)=0`

Check if the image generated by the viewer has the required resolution.

Protected Attributes

- int `rows`
- int `cols`
- double `simulationTime`

7.20.1 Detailed Description

Class for creating a Mat object.

Definition at line 20 of file MatFactory.hpp.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 MatFactory::MatFactory()

Definition at line 9 of file MatFactory.cpp.

7.20.2.2 MatFactory::~MatFactory() [virtual]

Definition at line 11 of file MatFactory.cpp.

7.20.3 Member Function Documentation

7.20.3.1 unsigned char MatFactory::convertToIntensity(double concentration) [protected]

Convert concentration to intensity.

Convert the concentration (real value between 0 and 1) to intensity (integer value between 0 and 255)

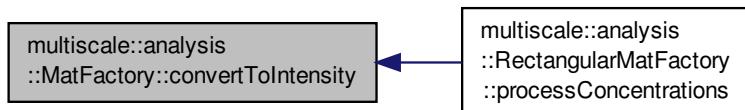
Parameters

<code>concentration</code>	A value between 0 and 1
----------------------------	-------------------------

Definition at line 43 of file MatFactory.cpp.

Referenced by multiscale::analysis::RectangularMatFactory::processConcentrations().

Here is the caller graph for this function:



7.20.3.2 Mat MatFactory::create(const string & inputFile)

Create a Mat object from the input file.

Create the Mat instance from the values given in the input file

FORMAT OF INPUT FILE:

- 1st line contains two positive integers and a real value: nr_rows, nr_cols and simulation_time
- 2nd - (nr_rows + 1)th lines contain the concentrations of the positions in the grid

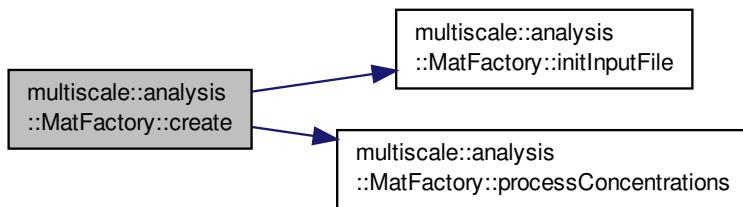
Parameters

<i>inputFile</i>	The path to the input file
------------------	----------------------------

Definition at line 13 of file MatFactory.cpp.

References cols, ERR_IN_EXTRA_DATA, initInputFile(), processConcentrations(), and rows.

Here is the call graph for this function:



7.20.3.3 virtual Mat multiscale::analysis::MatFactory::createFromViewerImage (const string & *inputFile*) [pure virtual]

Create a Mat object from the image file obtained from Rectangular/CircularGeometryViewer.

Create the Mat instance from the given image file

Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

Implemented in [multiscale::analysis::CircularMatFactory](#), and [multiscale::analysis::RectangularMatFactory](#).

7.20.3.4 void MatFactory::initInputFile (ifstream & *fin*, const string & *inputFile*) [protected]

Initialise the input file.

Initialise the input file. Open an input file stream to the given input file path.

Parameters

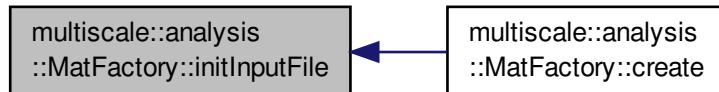
<i>fin</i>	An input stream for reading data from the input file
<i>inputFile</i>	The path to the input file

Definition at line 33 of file MatFactory.cpp.

References cols, ERR_INPUT_OPEN, rows, and simulationTime.

Referenced by `create()`.

Here is the caller graph for this function:



7.20.3.5 virtual bool multiscale::analysis::MatFactory::isValidViewerImage (const Mat & *image*) [protected], [pure virtual]

Check if the image generated by the viewer has the required resolution.

Parameters

<i>image</i>	Image generated by the viewer
--------------	-------------------------------

Implemented in [multiscale::analysis::RectangularMatFactory](#), and [multiscale::analysis::CircularMatFactory](#).

7.20.3.6 virtual double multiscale::analysis::MatFactory::maxColourBarIntensityFromViewerImage (const string & *inputFile*) [pure virtual]

Get the maximum grayscale intensity of the colour bar in the image.

Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

Implemented in [multiscale::analysis::CircularMatFactory](#), and [multiscale::analysis::RectangularMatFactory](#).

7.20.3.7 virtual unsigned char* multiscale::analysis::MatFactory::processConcentrations (ifstream & *fin*) [protected], [pure virtual]

Process concentrations from file.

Process the concentrations from the file. This method will be implemented only by subclasses of this abstract class

Implemented in [multiscale::analysis::RectangularMatFactory](#), and [multiscale::analysis::CircularMatFactory](#).

Referenced by `create()`.

Here is the caller graph for this function:



7.20.4 Member Data Documentation

7.20.4.1 int multiscale::analysis::MatFactory::cols [protected]

Number of columns in the Mat object

Definition at line 25 of file MatFactory.hpp.

Referenced by create(), initInputFile(), and multiscale::analysis::RectangularMatFactory::processConcentrations().

7.20.4.2 int multiscale::analysis::MatFactory::rows [protected]

Number of rows in the Mat object

Definition at line 24 of file MatFactory.hpp.

Referenced by create(), initInputFile(), and multiscale::analysis::RectangularMatFactory::processConcentrations().

7.20.4.3 double multiscale::analysis::MatFactory::simulationTime [protected]

Simulation time read from the input file

Definition at line 26 of file MatFactory.hpp.

Referenced by initInputFile().

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

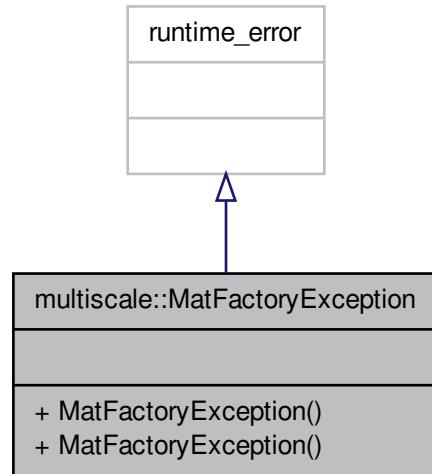
- include/multiscale/analysis/spatial/[MatFactory.hpp](#)
- src/multiscale/analysis/spatial/[MatFactory.cpp](#)

7.21 multiscale::MatFactoryException Class Reference

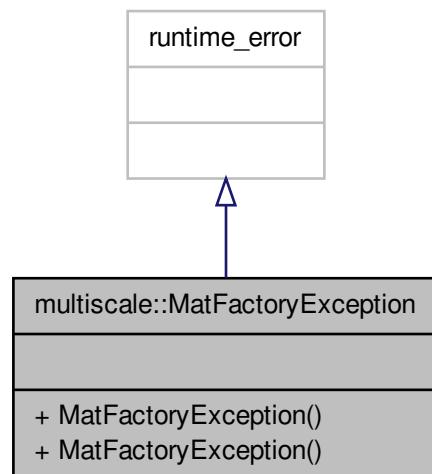
Exception class for the MatFactory class.

```
#include <MatFactoryException.hpp>
```

Inheritance diagram for multiscale::MatFactoryException:



Collaboration diagram for multiscale::MatFactoryException:



Public Member Functions

- [MatFactoryException](#) (const string &msg)
- [MatFactoryException](#) (const char *msg)

7.21.1 Detailed Description

Exception class for the MatFactory class.

Definition at line 13 of file MatFactoryException.hpp.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 multiscale::MatFactoryException::MatFactoryException (const string & msg) [inline]

Definition at line 17 of file MatFactoryException.hpp.

7.21.2.2 multiscale::MatFactoryException::MatFactoryException (const char * msg) [inline]

Definition at line 18 of file MatFactoryException.hpp.

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

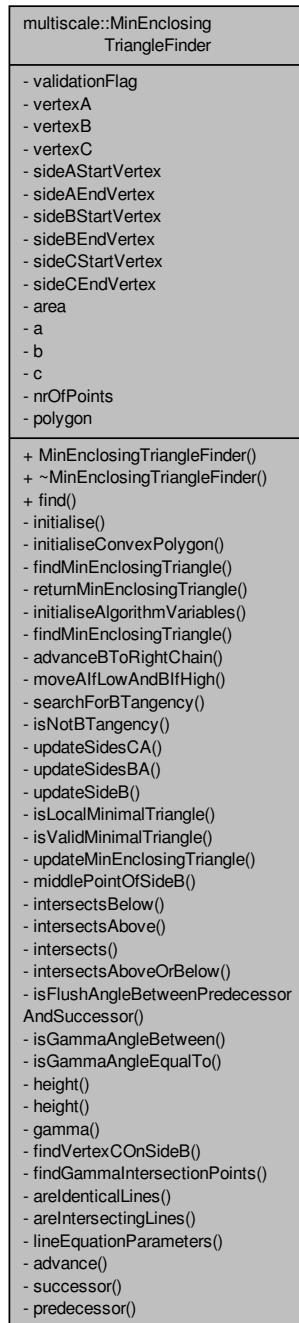
- [include/multiscale/exception/MatFactoryException.hpp](#)

7.22 multiscale::MinEnclosingTriangleFinder Class Reference

Class for computing the minimum area enclosing triangle for a given polygon.

```
#include <MinEnclosingTriangleFinder.hpp>
```

Collaboration diagram for multiscale::MinEnclosingTriangleFinder:



Public Member Functions

- [MinEnclosingTriangleFinder \(\)](#)
- [~MinEnclosingTriangleFinder \(\)](#)
- double [find](#) (const vector< Point2f > &points, vector< Point2f > &minEnclosingTriangle)

Find the minimum area enclosing triangle for the given 2D point set.

Private Member Functions

- void **initialise** (const vector< Point2f > &points, vector< Point2f > &minEnclosingTriangle)

Initialisation function for the class.
- void **initialiseConvexPolygon** (const vector< Point2f > &points)

Initialise polygon as the convex hull of the given set of points.
- double **findMinEnclosingTriangle** (const vector< Point2f > &points, vector< Point2f > &minEnclosingTriangle)

Find the minimum area enclosing triangle for the given polygon.
- double **returnMinEnclosingTriangle** (const vector< Point2f > &points, vector< Point2f > &minEnclosingTriangle)

Return the minimum area enclosing triangle in case the given polygon has at most three points.
- void **initialiseAlgorithmVariables** ()

Initialisation of the algorithm variables.
- void **findMinEnclosingTriangle** (vector< Point2f > &minEnclosingTriangle, double &minEnclosingTriangleArea)

Find the minimum area enclosing triangle for the given polygon.
- void **advanceBToRightChain** ()

Advance b to the right chain.
- void **moveAIfLowAndBIfHigh** ()

Move "a" if it is low and "b" if it is high.
- void **searchForBTangency** ()

Search for the tangency of side B.
- bool **isNotBTangency** ()

Check if tangency for side B was not obtained.
- void **updateSidesCA** ()

Update sides A and C.
- void **updateSidesBA** ()

Update sides B and possibly A if tangency for side B was not obtained.
- void **updateSideB** ()

Set side B if tangency for side B was obtained.
- bool **isLocalMinimalTriangle** ()

Update the triangle vertices after all sides were set and check if a local minimal triangle was found or not.
- bool **isValidMinimalTriangle** ()

Check if the found minimal triangle is valid.
- void **updateMinEnclosingTriangle** (vector< Point2f > &minEnclosingTriangle, double &minEnclosingTriangleArea)

Update the current minimum area enclosing triangle if the newly obtained one has a smaller area.
- bool **middlePointOfSideB** (Point2f &middlePointOfSideB)

Return the middle point of side B.
- bool **intersectsBelow** (const Point2f &gammaPoint, unsigned int polygonPointIndex)

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon below the point polygon[polygonPointIndex].
- bool **intersectsAbove** (const Point2f &gammaPoint, unsigned int polygonPointIndex)

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon above the point polygon[polygonPointIndex].
- unsigned int **intersects** (double angleOfGammaAndPoint, unsigned int polygonPointIndex)

Check if/where the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon.
- unsigned int **intersectsAboveOrBelow** (unsigned int successorOrPredecessorIndex, unsigned int pointIndex)

If (gamma(x) x) intersects P between successorOrPredecessorIndex and pointIndex is it above/below?

- bool `isFlushAngleBetweenPredecessorAndSuccessor` (double &angleFlushEdge, double anglePredecessor, double angleSuccessor)

Check if the angle of the flush edge or its opposite angle lie between the angle of the predecessor and successor.
- bool `isGammaAngleBetween` (double &gammaAngle, double angle1, double angle2)

Check if the angle of the line (gamma(p) p) or its opposite angle lie between angle1 and angle2.
- bool `isGammaAngleEqualTo` (double &gammaAngle, double angle)

Check if the angle of the line (gamma(p) p) or its opposite angle is equal to the given angle.
- double `height` (unsigned int polygonPointIndex)

Compute the height of the point specified by the given index.
- double `height` (const Point2f &polygonPoint)

Compute the height of the point.
- bool `gamma` (unsigned int polygonPointIndex, Point2f &gammaPoint)

Find gamma for a given point "p" specified by its index.
- Point2f `findVertexCOnSideB` ()

*Find vertex C which lies on side B at a distance = 2 * height(a-1) from side C.*
- bool `findGammaIntersectionPoints` (unsigned int polygonPointIndex, const Point2f &side1StartVertex, const Point2f &side1EndVertex, const Point2f &side2StartVertex, const Point2f &side2EndVertex, Point2f &intersectionPoint1, Point2f &intersectionPoint2)

Find the intersection points to compute gamma(point)
- bool `areIdenticalLines` (const vector< double > &side1Params, const vector< double > &side2Params, double sideCEExtraParam)

Check if the given lines are identical or not.
- bool `areIntersectingLines` (const vector< double > &side1Params, const vector< double > &side2Params, double sideCEExtraParam, Point2f &intersectionPoint1, Point2f &intersectionPoint2)

Check if the given lines intersect or not. If the lines intersect find their intersection points.
- vector< double > `lineEquationParameters` (const Point2f &p, const Point2f &q)

Get the line equation parameters "a", "b" and "c" for the line determined by points "p" and "q".
- void `advance` (unsigned int &index)

Advance the given index with one position.
- unsigned int `successor` (unsigned int index)

Return the successor of the provided point index.
- unsigned int `predecessor` (unsigned int index)

Return the predecessor of the provided point index.

Private Attributes

- unsigned int `validationFlag`
- Point2f `vertexA`
- Point2f `vertexB`
- Point2f `vertexC`
- Point2f `sideAStartVertex`
- Point2f `sideAEndVertex`
- Point2f `sideBStartVertex`
- Point2f `sideBEndVertex`
- Point2f `sideCStartVertex`
- Point2f `sideCEndVertex`
- double `area`
- unsigned int `a`
- unsigned int `b`
- unsigned int `c`
- unsigned int `nrOfPoints`
- vector< Point2f > `polygon`

7.22.1 Detailed Description

Class for computing the minimum area enclosing triangle for a given polygon.

This implementation has a linear complexity ($\theta(n)$) with respect to the number of points defining the convex polygon and is based on the algorithm described in the following paper:

J. O'Rourke, A. Aggarwal, S. Maddila, and M. Baldwin, 'An optimal algorithm for finding minimal enclosing triangles', Journal of Algorithms, vol. 7, no. 2, pp. 258–269, Jun. 1986.

Definition at line 36 of file MinEnclosingTriangleFinder.hpp.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 MinEnclosingTriangleFinder::MinEnclosingTriangleFinder ()

Definition at line 12 of file MinEnclosingTriangleFinder.cpp.

7.22.2.2 MinEnclosingTriangleFinder::~MinEnclosingTriangleFinder ()

Definition at line 14 of file MinEnclosingTriangleFinder.cpp.

7.22.3 Member Function Documentation

7.22.3.1 void MinEnclosingTriangleFinder::advance (unsigned int & index) [private]

Advance the given index with one position.

Parameters

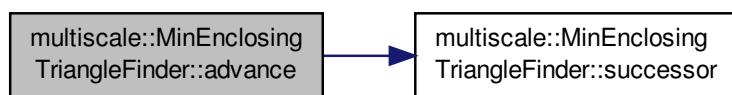
<i>index</i>	Index of the point
--------------	--------------------

Definition at line 395 of file MinEnclosingTriangleFinder.cpp.

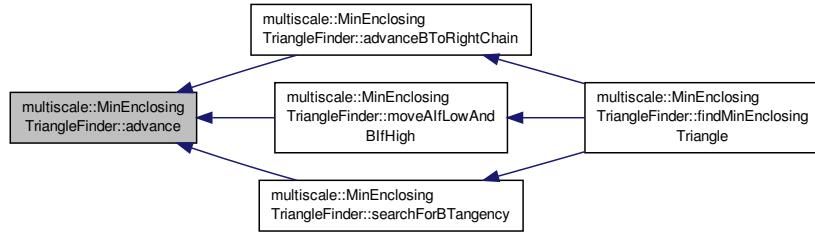
References successor().

Referenced by advanceBToRightChain(), moveAIfLowAndBIfHigh(), and searchForBTangency().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.2 void MinEnclosingTriangleFinder::advanceBToRightChain() [private]

Advance b to the right chain.

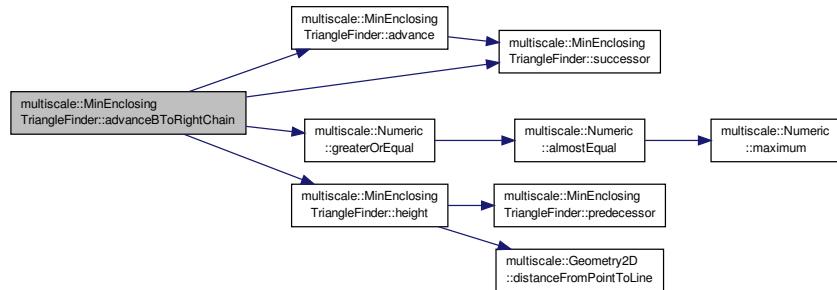
See paper for more details

Definition at line 92 of file MinEnclosingTriangleFinder.cpp.

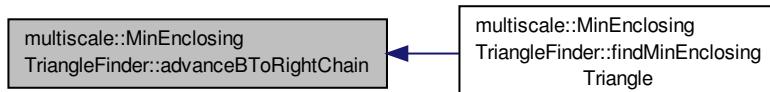
References advance(), b, multiscale::Numeric::greaterOrEqual(), height(), and successor().

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.3 bool MinEnclosingTriangleFinder::areIdenticalLines (const vector< double > & side1Params, const vector< double > & side2Params, double sideCExtraParam) [private]

Check if the given lines are identical or not.

The lines are specified as: $ax + by + c = 0$ OR $ax + by + c (+/-) \text{sideCExtraParam} = 0$

Parameters

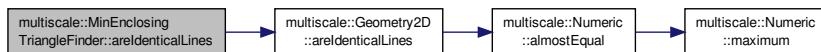
<i>side1Params</i>	Vector containing the values of a, b and c for side 1
<i>side2Params</i>	Vector containing the values of a, b and c for side 2
<i>sideCExtraParam</i>	Extra parameter for the flush edge C

Definition at line 360 of file MinEnclosingTriangleFinder.cpp.

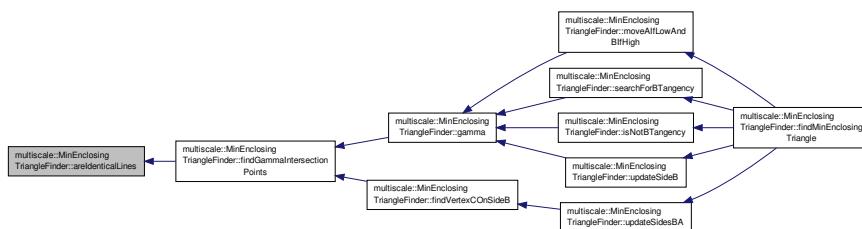
References multiscale::Geometry2D::areIdenticalLines().

Referenced by findGammalIntersectionPoints().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.4 bool MinEnclosingTriangleFinder::areIntersectingLines (const vector< double > & side1Params, const vector< double > & side2Params, double sideCExtraParam, Point2f & intersectionPoint1, Point2f & intersectionPoint2) [private]

Check if the given lines intersect or not. If the lines intersect find their intersection points.

The lines are specified as: $ax + by + c = 0$ OR $ax + by + c (+/-) \text{sideCExtraParam} = 0$

Parameters

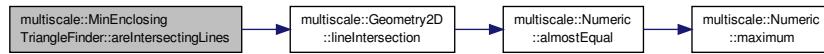
<i>side1Params</i>	Vector containing the values of a, b and c for side 1
<i>side2Params</i>	Vector containing the values of a, b and c for side 2
<i>sideCExtraParam</i>	Extra parameter for the flush edge C
<i>intersectionPoint1</i>	The first intersection point, if it exists
<i>intersectionPoint2</i>	The second intersection point, if it exists

Definition at line 370 of file MinEnclosingTriangleFinder.cpp.

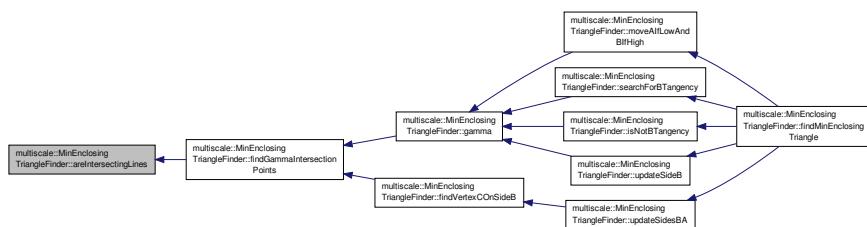
References multiscale::Geometry2D::lineIntersection().

Referenced by findGammaIntersectionPoints().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.5 double MinEnclosingTriangleFinder::find (const vector< Point2f > & points, vector< Point2f > & minEnclosingTriangle)

Find the minimum area enclosing triangle for the given 2D point set.

Precondition: Number of points in the set is at least 1.

Parameters

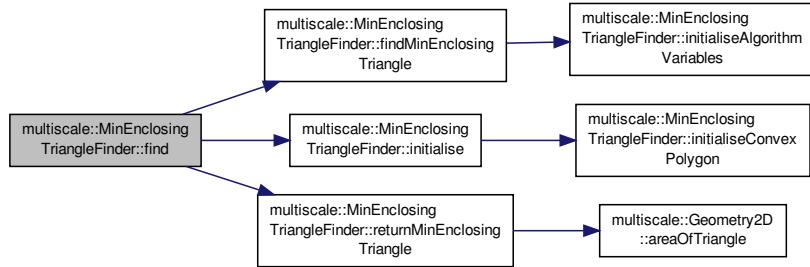
<i>points</i>	Set of points
<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 16 of file MinEnclosingTriangleFinder.cpp.

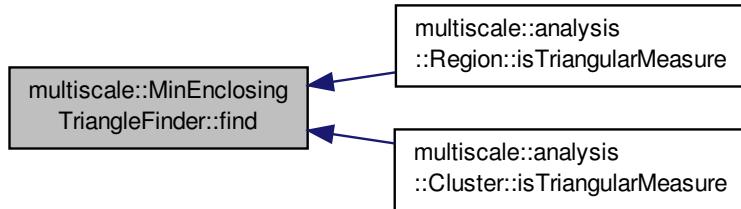
References findMinEnclosingTriangle(), initialise(), polygon, and returnMinEnclosingTriangle().

Referenced by multiscale::analysis::Region::isTriangularMeasure(), and multiscale::analysis::Cluster::isTriangularMeasure().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.6 `bool MinEnclosingTriangleFinder::findGammaIntersectionPoints (unsigned int polygonPointIndex, const Point2f & side1StartVertex, const Point2f & side1EndVertex, const Point2f & side2StartVertex, const Point2f & side2EndVertex, Point2f & intersectionPoint1, Point2f & intersectionPoint2) [private]`

Find the intersection points to compute gamma(point)

Parameters

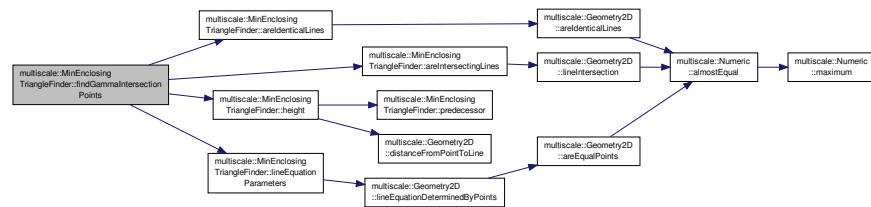
<i>polygonPoint-Index</i>	Index of the polygon point for which the distance is known
<i>side1StartVertex</i>	Start vertex for side 1
<i>side1EndVertex</i>	End vertex for side 1
<i>side2StartVertex</i>	Start vertex for side 2
<i>side2EndVertex</i>	End vertex for side 2
<i>intersection-Point1</i>	First intersection point between one pair of lines
<i>intersection-Point2</i>	Second intersection point between another pair of lines

Definition at line 337 of file MinEnclosingTriangleFinder.cpp.

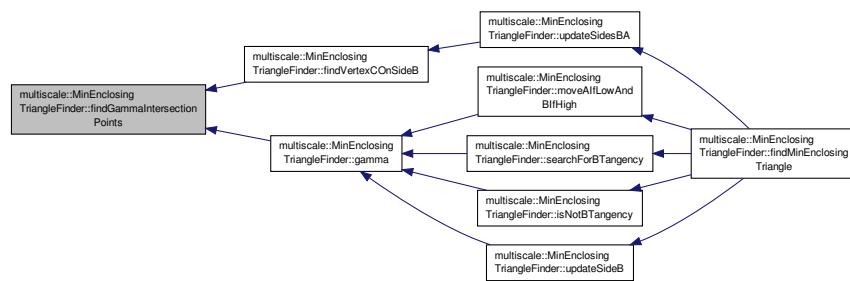
References areIdenticalLines(), areIntersectingLines(), height(), and lineEquationParameters().

Referenced by findVertexCOnSideB(), and gamma().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.7 double MinEnclosingTriangleFinder::findMinEnclosingTriangle (const vector< Point2f > & polygon, vector< Point2f > & minEnclosingTriangle) [private]

Find the minimum area enclosing triangle for the given polygon.

Parameters

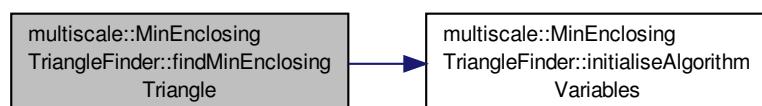
<i>polygon</i>	Polygon of points for which the minimum area enclosing triangle will be found
<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 41 of file MinEnclosingTriangleFinder.cpp.

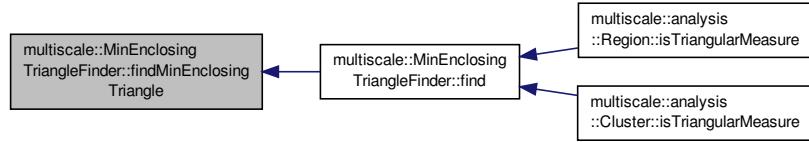
References initialiseAlgorithmVariables().

Referenced by find().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.8 void MinEnclosingTriangleFinder::findMinEnclosingTriangle (vector< Point2f > & minEnclosingTriangle, double & minEnclosingTriangleArea) [private]

Find the minimum area enclosing triangle for the given polygon.

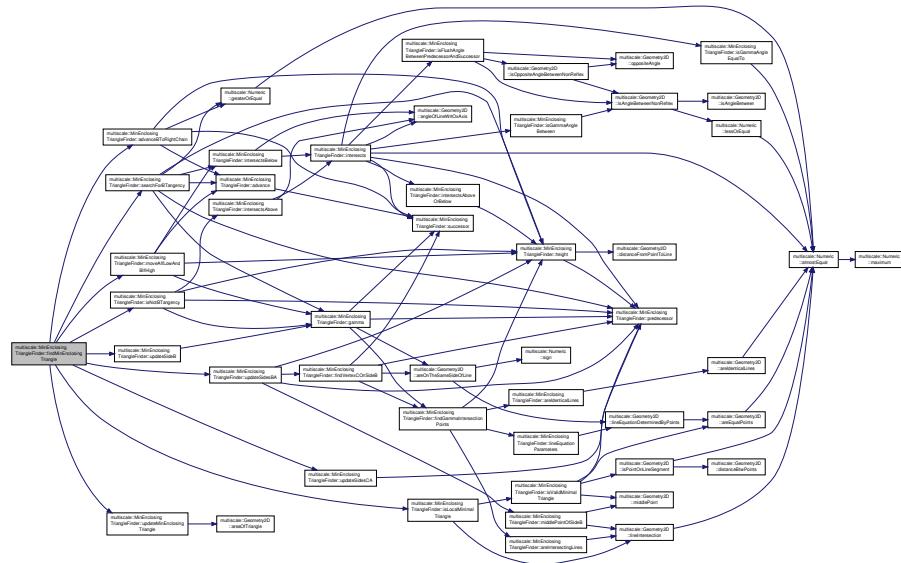
Parameters

<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon
<i>minEnclosingTriangleArea</i>	Area of the minimum area enclosing triangle

Definition at line 71 of file MinEnclosingTriangleFinder.cpp.

References advanceBToRightChain(), c, isLocalMinimalTriangle(), isNotBTangency(), moveAIfLowAndBIfHigh(), nrOfPoints, searchForBTangency(), updateMinEnclosingTriangle(), updateSideB(), updateSidesBA(), and updateSidesCA().

Here is the call graph for this function:



7.22.3.9 Point2f MinEnclosingTriangleFinder::findVertexCOnSideB () [private]

Find vertex C which lies on side B at a distance = $2 * \text{height}(a-1)$ from side C.

Considering that line (x y) is a line parallel to (c c-1) and that the distance between the lines is equal to $2 * \text{height}(a-1)$, we can have two possible (x y) lines.

Therefore, we will compute two intersection points between the lines (x y) and (b b-1) and take the point which is closest to point polygon[b].

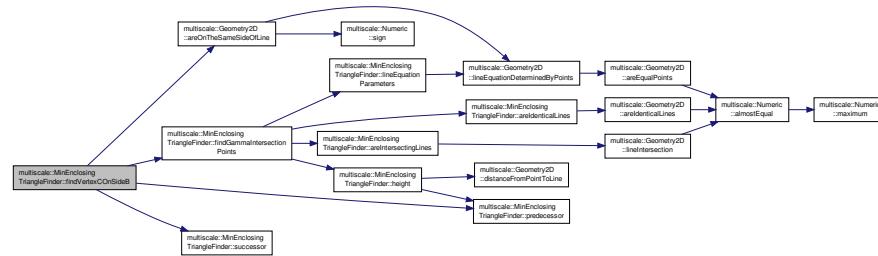
See paper and formula for distance from point to a line for more details

Definition at line 320 of file MinEnclosingTriangleFinder.cpp.

References a, multiscale::Geometry2D::areOnTheSameSideOfLine(), c, ERR_VERTEX_C_ON_SIDE_B, findGammaIntersectionPoints(), polygon, predecessor(), sideBEndVertex, sideBStartVertex, sideCEndVertex, sideCStartVertex, and successor().

Referenced by updateSidesBA().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.10 bool MinEnclosingTriangleFinder::gamma (unsigned int *polygonPointIndex*, Point2f & *gammaPoint*) [private]

Find gamma for a given point "p" specified by its index.

The function returns true if gamma exists i.e. if lines (a a-1) and (x y) intersect and false otherwise. In case the two lines intersect in point intersectionPoint, gamma is computed.

Considering that line (x y) is a line parallel to (c c-1) and that the distance between the lines is equal to $2 * \text{height}(p)$, we can have two possible (x y) lines.

Therefore, we will compute two intersection points between the lines (x y) and (a a-1) and take the point which is closest to point polygon[a].

See paper and formula for distance from point to a line for more details

Parameters

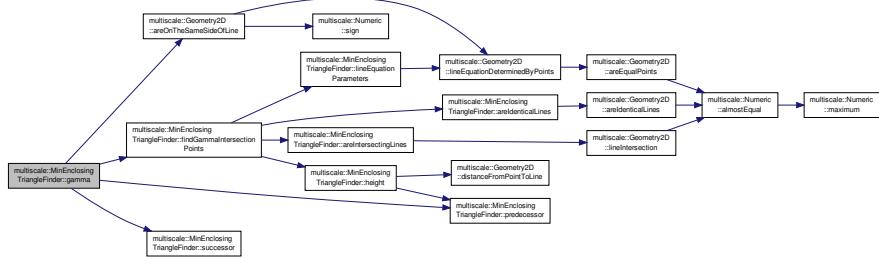
<i>polygonPointIndex</i>	Index of the polygon point
<i>gammaPoint</i>	Point2f $\text{gamma}(\text{polygon}[\text{polygonPointIndex}])$

Definition at line 301 of file MinEnclosingTriangleFinder.cpp.

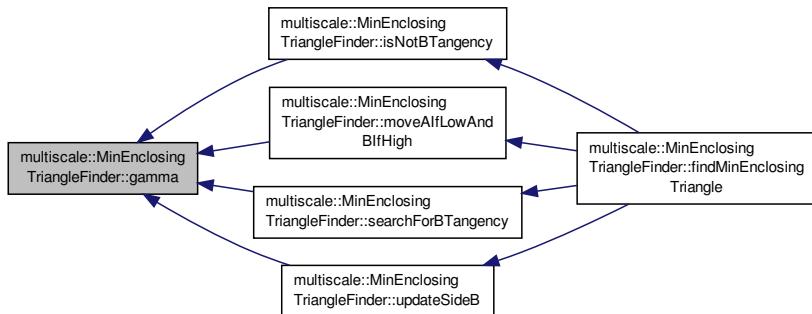
References a, multiscale::Geometry2D::areOnTheSameSideOfLine(), c, findGammaIntersectionPoints(), polygon, predecessor(), and successor().

Referenced by isNotBTangency(), moveAIfLowAndBIfHigh(), searchForBTangency(), and updateSideB().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.11 double MinEnclosingTriangleFinder::height (unsigned int *polygonPointIndex*) [private]

Compute the height of the point specified by the given index.

See paper for more details

Parameters

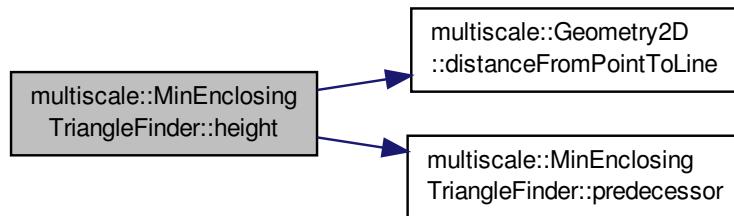
<i>polygonPointIndex</i>	Index of the polygon point
--------------------------	----------------------------

Definition at line 285 of file MinEnclosingTriangleFinder.cpp.

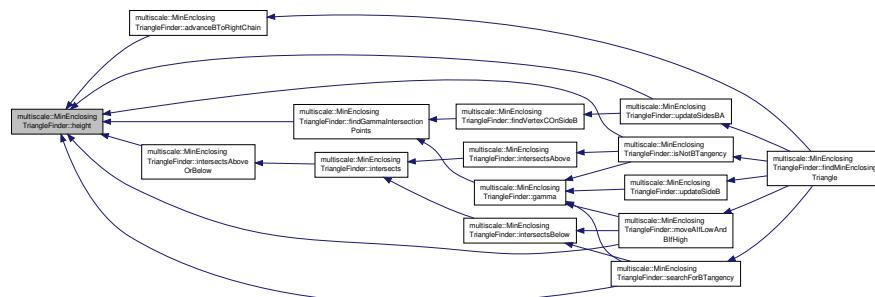
References c, multiscale::Geometry2D::distanceFromPointToLine(), polygon, and predecessor().

Referenced by advanceBToRightChain(), findGammaIntersectionPoints(), intersectsAboveOrBelow(), isNotBTangency(), moveAIfLowAndBIfHigh(), searchForBTangency(), and updateSidesBA().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.12 double MinEnclosingTriangleFinder::height (const Point2f & polygonPoint) [private]

Compute the height of the point.

See paper for more details

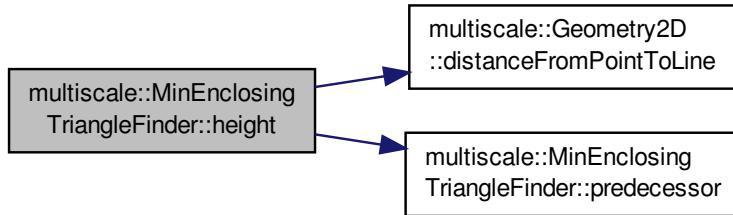
Parameters

<i>polygonPoint</i>	Polygon point
---------------------	---------------

Definition at line 294 of file MinEnclosingTriangleFinder.cpp.

References c, multiscale::Geometry2D::distanceFromPointToLine(), polygon, and predecessor().

Here is the call graph for this function:



7.22.3.13 void MinEnclosingTriangleFinder::initialise (const vector< Point2f > & *points*, vector< Point2f > & *minEnclosingTriangle*) [private]

Initialisation function for the class.

Initialise the polygon and other class' fields.

Parameters

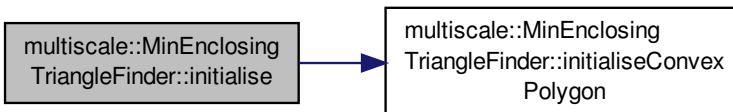
<i>points</i>	Set of points
<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 28 of file MinEnclosingTriangleFinder.cpp.

References initialiseConvexPolygon().

Referenced by find().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.14 void MinEnclosingTriangleFinder::initialiseAlgorithmVariables () [private]

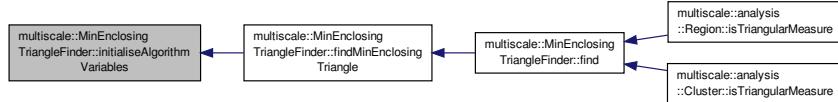
Initialisation of the algorithm variables.

Definition at line 63 of file MinEnclosingTriangleFinder.cpp.

References a, b, c, nrOfPoints, and polygon.

Referenced by findMinEnclosingTriangle().

Here is the caller graph for this function:



7.22.3.15 void MinEnclosingTriangleFinder::initialiseConvexPolygon (const vector< Point2f > & points) [private]

Initialise polygon as the convex hull of the given set of points.

Parameters

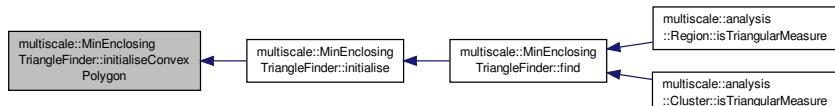
<i>points</i>	Set of points
---------------	---------------

Definition at line 35 of file MinEnclosingTriangleFinder.cpp.

References CONVEX_HULL_CLOCKWISE, and polygon.

Referenced by initialise().

Here is the caller graph for this function:



7.22.3.16 `unsigned int MinEnclosingTriangleFinder::intersects (double angleOfGammaAndPoint, unsigned int polygonPointIndex) [private]`

Check if/where the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon.

Parameters

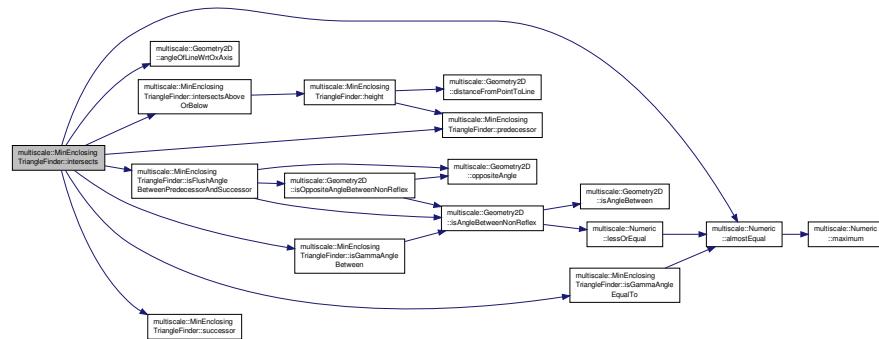
<code>angleOfGamma-AndPoint</code>	Angle between gammaPoint and polygon[polygonPointIndex]
<code>polygonPoint-Index</code>	Index of the polygon point which is considered when determining the line

Definition at line 232 of file MinEnclosingTriangleFinder.cpp.

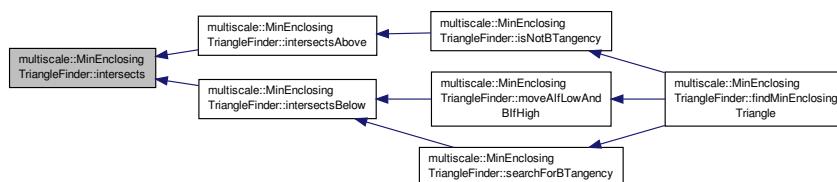
References multiscale::Numeric::almostEqual(), multiscale::Geometry2D::angleOfLineWrtOxAxis(), c, INTERSECTS_BELOW, INTERSECTS_CRITICAL, intersectsAboveOrBelow(), isFlushAngleBetweenPredecessorAndSuccessor(), isGammaAngleBetween(), isGammaAngleEqualTo(), polygon, predecessor(), and successor().

Referenced by intersectsAbove(), and intersectsBelow().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.17 `bool MinEnclosingTriangleFinder::intersectsAbove (const Point2f & gammaPoint, unsigned int polygonPointIndex) [private]`

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon above the point polygon[polygonPointIndex].

Parameters

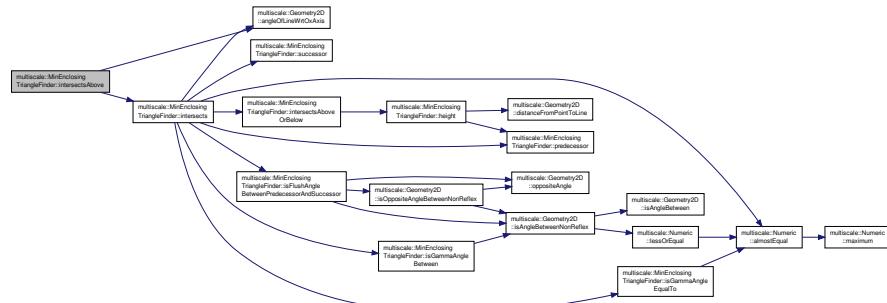
<code>gammaPoint</code>	Gamma(p)
<code>polygonPoint-Index</code>	Index of the polygon point which is considered when determining the line

Definition at line 226 of file MinEnclosingTriangleFinder.cpp.

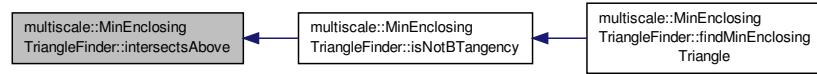
References multiscale::Geometry2D::angleOfLineWrtOxAxis(), intersects(), INTERSECTS_ABOVE, and polygon.

Referenced by `isNotBTangency()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.18 unsigned int MinEnclosingTriangleFinder::intersectsAboveOrBelow (unsigned int successorOrPredecessorIndex, unsigned int pointIndex) [private]

If (gamma(x) x) intersects P between successorOrPredecessorIndex and pointIndex is it above/below?

Parameters

<i>successorOrPredecessorIndex</i>	Index of the successor or predecessor
<i>pointIndex</i>	Index of the point x in the polygon

Definition at line 256 of file MinEnclosingTriangleFinder.cpp.

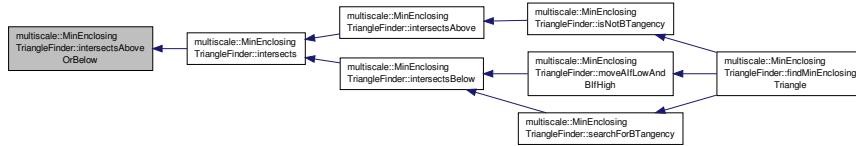
References height(), INTERSECTS ABOVE, and INTERSECTS BELOW.

Referenced by intersects().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.19 bool MinEnclosingTriangleFinder::intersectsBelow (const Point2f & gammaPoint, unsigned int polygonPointIndex) [private]

Check if the line determined by gammaPoint and polygon[polygonPointIndex] intersects the polygon below the point polygon[polygonPointIndex].

Parameters

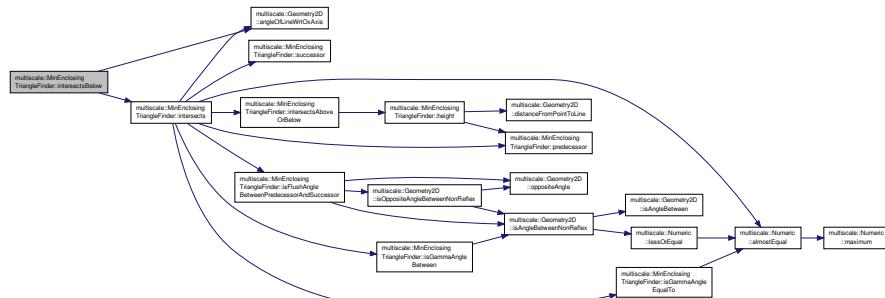
<i>gammaPoint</i>	Gamma(p)
<i>polygonPoint-Index</i>	Index of the polygon point which is considered when determining the line

Definition at line 220 of file MinEnclosingTriangleFinder.cpp.

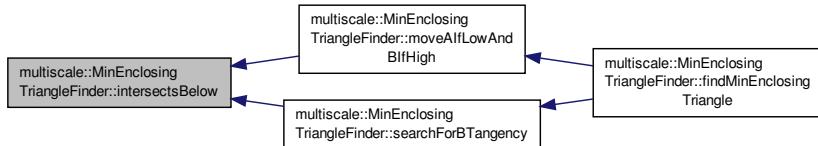
References multiscale::Geometry2D::angleOfLineWrtOxAxis(), intersects(), INTERSECTS_BELOW, and polygon.

Referenced by moveAlfLowAndBIfHigh(), and searchForBTangency().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.20 bool MinEnclosingTriangleFinder::isFlushAngleBetweenPredecessorAndSuccessor (double & angleFlushEdge, double anglePredecessor, double angleSuccessor) [private]

Check if the angle of the flush edge or its opposite angle lie between the angle of the predecessor and successor.

Parameters

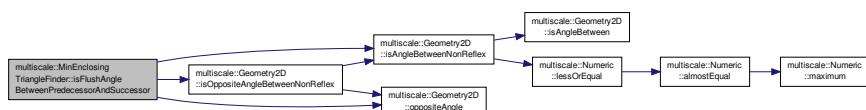
<i>angleFlushEdge</i>	Angle of the flush edge
<i>anglePredecessor</i>	Angle of the predecessor
<i>angleSuccessor</i>	Angle of the successor

Definition at line 264 of file MinEnclosingTriangleFinder.cpp.

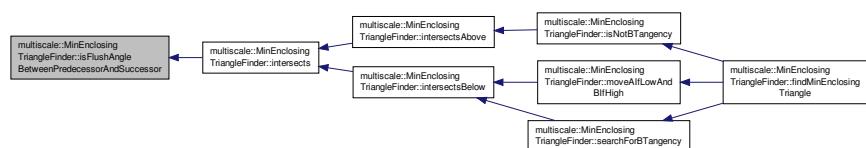
References multiscale::Geometry2D::isAngleBetweenNonReflex(), multiscale::Geometry2D::isOppositeAngleBetweenNonReflex(), and multiscale::Geometry2D::oppositeAngle().

Referenced by intersects().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.21 bool MinEnclosingTriangleFinder::isGammaAngleBetween (double & gammaAngle, double angle1, double angle2) [private]

Check if the angle of the line (gamma(p) p) or its opposite angle lie between angle1 and angle2.

Parameters

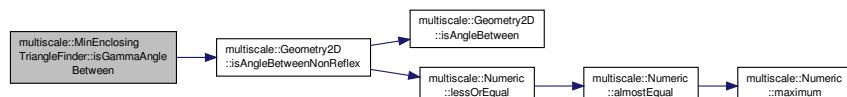
<i>gammaAngle</i>	Angle of the line ($\text{gamma}(p)$ p)
<i>angle1</i>	One of the boundary angles
<i>angle2</i>	Another boundary angle

Definition at line 277 of file MinEnclosingTriangleFinder.cpp.

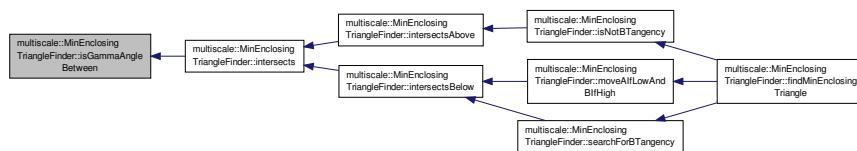
References multiscale::Geometry2D::isAngleBetweenNonReflex().

Referenced by intersects().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.22 bool MinEnclosingTriangleFinder::isGammaAngleEqualTo (double & *gammaAngle*, double *angle*) [private]

Check if the angle of the line ($\text{gamma}(p)$ p) or its opposite angle is equal to the given angle.

Parameters

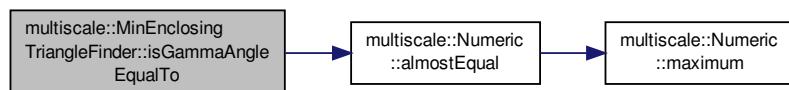
<i>gammaAngle</i>	Angle of the line ($\text{gamma}(p)$ p)
<i>angle</i>	Angle to compare against

Definition at line 281 of file MinEnclosingTriangleFinder.cpp.

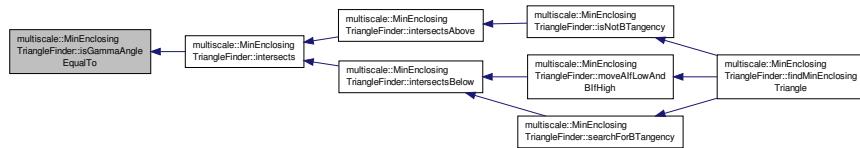
References multiscale::Numeric::almostEqual().

Referenced by intersects().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.23 bool MinEnclosingTriangleFinder::isLocalMinimalTriangle() [private]

Update the triangle vertices after all sides were set and check if a local minimal triangle was found or not.

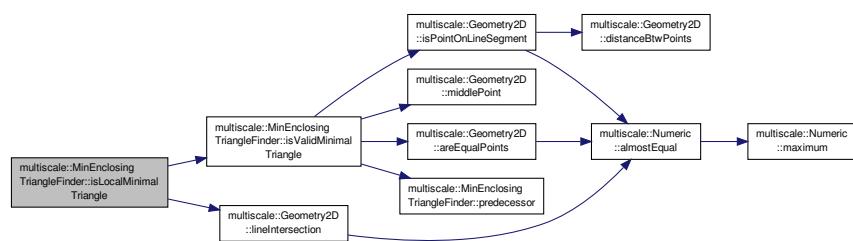
See paper for more details

Definition at line 165 of file MinEnclosingTriangleFinder.cpp.

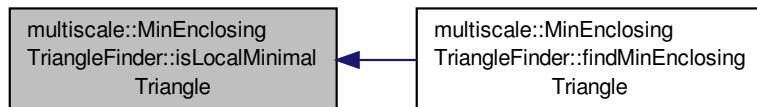
References isValidMinimalTriangle(), multiscale::Geometry2D::lineIntersection(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex, sideCEndVertex, sideCStartVertex, vertexA, vertexB, and vertexC.

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.24 bool MinEnclosingTriangleFinder::isNotBTangency() [private]

Check if tangency for side B was not obtained.

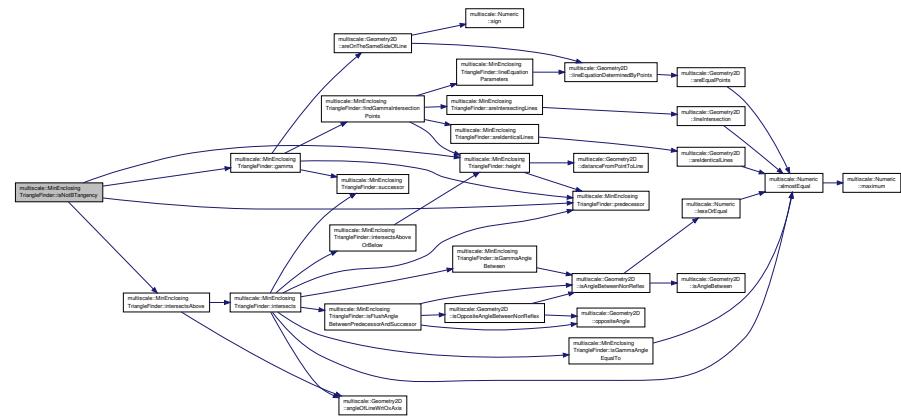
See paper for more details

Definition at line 119 of file MinEnclosingTriangleFinder.cpp.

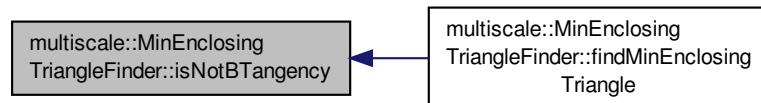
References a, b, gamma(), height(), intersectsAbove(), and predecessor().

Referenced by `findMinEnclosingTriangle()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.25 bool MinEnclosingTriangleFinder::isValidMinimalTriangle() [private]

Check if the found minimal triangle is valid.

This means that all midpoints of the triangle should touch the polygon.

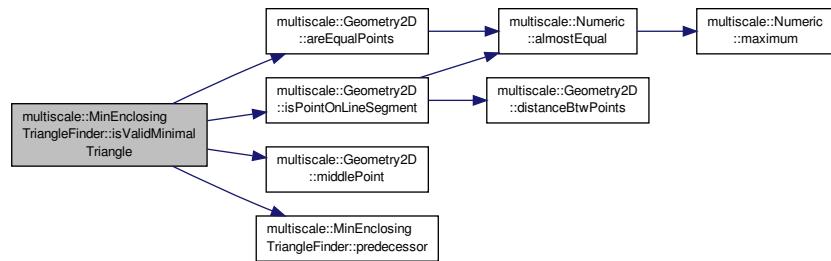
See paper for more details

Definition at line 175 of file MinEnclosingTriangleFinder.cpp.

References a, multiscale::Geometry2D::areEqualPoints(), b, multiscale::Geometry2D::isPointOnLineSegment(), multiscale::Geometry2D::middlePoint(), polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideBEnd-Vertex, sideBStartVertex, sideCEndVertex, sideCStartVertex, VALIDATION_SIDE_A_TANGENT, VALIDATION_-SIDE_B_TANGENT, validationFlag, vertexA, vertexB, and vertexC.

Referenced by `isLocalMinimalTriangle()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.26 `vector< double > MinEnclosingTriangleFinder::lineEquationParameters (const Point2f & p, const Point2f & q) [private]`

Get the line equation parameters "a", "b" and "c" for the line determined by points "p" and "q".

The equation of the line is considered in the general form: $ax + by + c = 0$

Parameters

<code>p</code>	One point for defining the equation of the line
<code>q</code>	Second point for defining the equation of the line

Definition at line 382 of file MinEnclosingTriangleFinder.cpp.

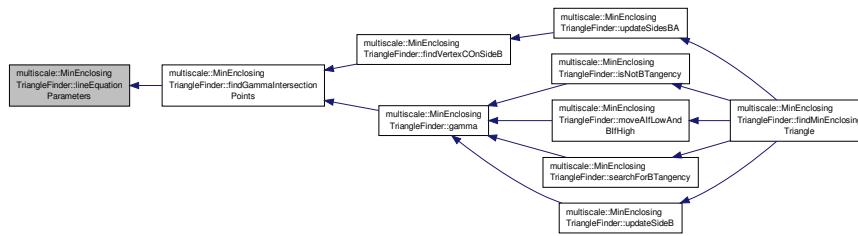
References a, b, c, and multiscale::Geometry2D::lineEquationDeterminedByPoints().

Referenced by findGammalIntersectionPoints().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.27 bool MinEnclosingTriangleFinder::middlePointOfSideB (Point2f & middlePointOfSideB) [private]

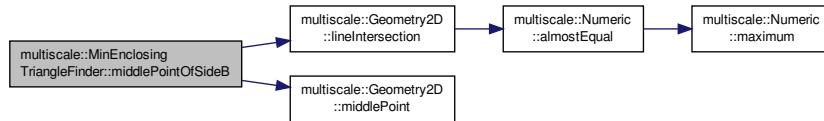
Return the middle point of side B.

Definition at line 207 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::lineIntersection(), multiscale::Geometry2D::middlePoint(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex, sideCEndVertex, sideCStartVertex, vertexA, and vertexC.

Referenced by updateSidesBA().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.28 void MinEnclosingTriangleFinder::moveAIfLowAndBIfHigh () [private]

Move "a" if it is low and "b" if it is high.

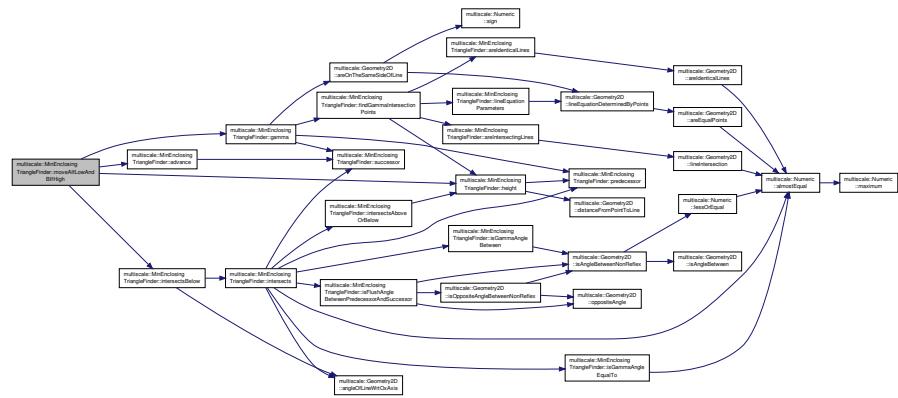
See paper for more details

Definition at line 98 of file MinEnclosingTriangleFinder.cpp.

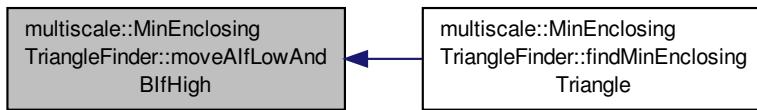
References a, advance(), b, gamma(), height(), and intersectsBelow().

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.29 `unsigned int MinEnclosingTriangleFinder::predecessor (unsigned int index) [private]`

Return the predecessor of the provided point index.

The predecessor of the first polygon point is the last polygon point (circular referencing)

Parameters

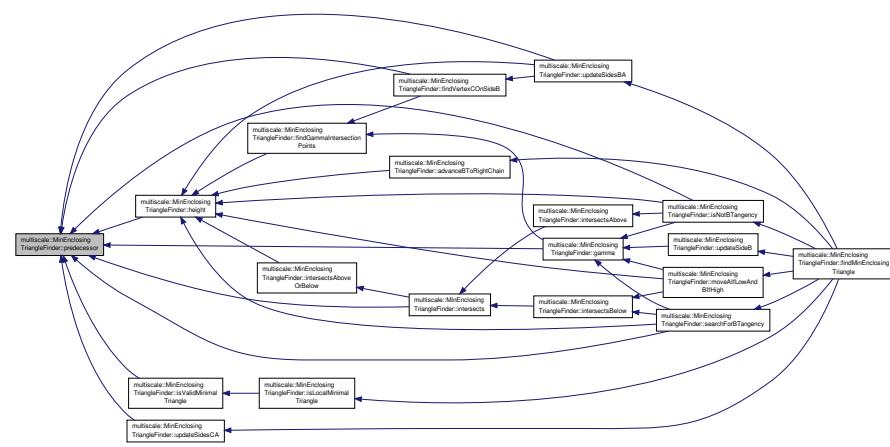
<i>index</i>	Index of the point
--------------	--------------------

Definition at line 403 of file MinEnclosingTriangleFinder.cpp.

References nrOfPoints.

Referenced by `findVertexCOnSideB()`, `gamma()`, `height()`, `intersects()`, `isNotBTangency()`, `isValidMinimalTriangle()`, `searchForBTangency()`, `updateSidesBA()`, and `updateSidesCA()`.

Here is the caller graph for this function:



7.22.3.30 double MinEnclosingTriangleFinder::returnMinEnclosingTriangle (const vector< Point2f > & *polygon*, vector< Point2f > & *minEnclosingTriangle*) [private]

Return the minimum area enclosing triangle in case the given polygon has at most three points.

Parameters

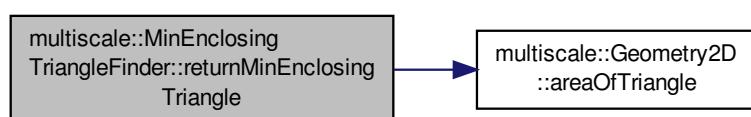
<i>polygon</i>	Polygon of points for which the minimum area enclosing triangle will be found
<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon

Definition at line 52 of file MinEnclosingTriangleFinder.cpp.

References multiscale::Geometry2D::areaOfTriangle().

Referenced by find().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.31 void MinEnclosingTriangleFinder::searchForBTangency() [private]

Search for the tangency of side B.

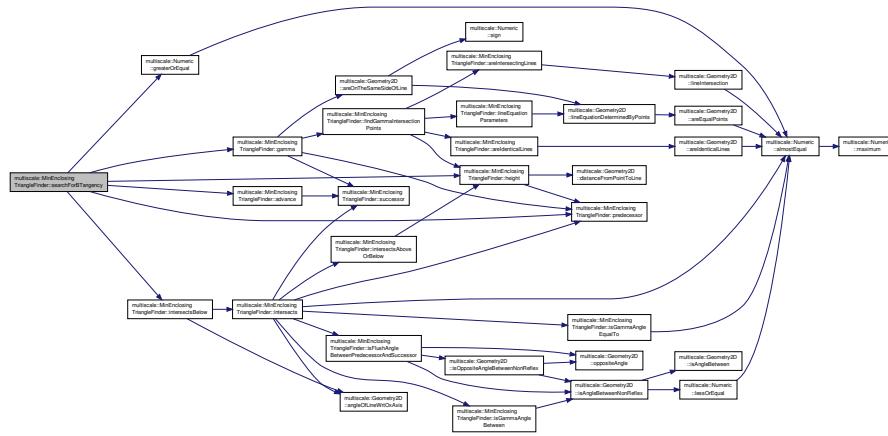
See paper for more details

Definition at line 110 of file MinEnclosingTriangleFinder.cpp.

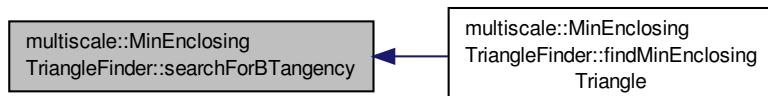
References a, advance(), b, gamma(), multiscale::Numeric::greaterOrEqual(), height(), intersectsBelow(), and predecessor().

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.32 unsigned int MinEnclosingTriangleFinder::successor (unsigned int *index*) [private]

Return the successor of the provided point index.

The successor of the last polygon point is the first polygon point (circular referencing)

Parameters

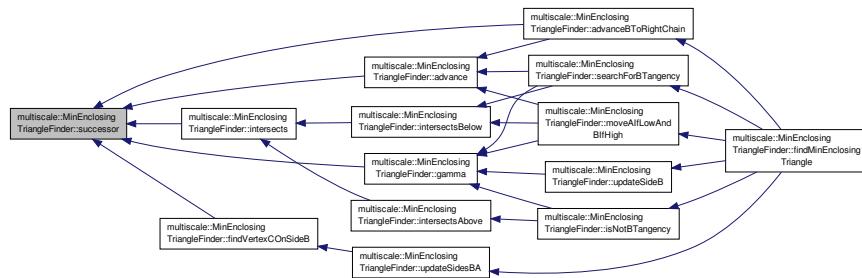
<i>index</i>	Index of the point
--------------	--------------------

Definition at line 399 of file MinEnclosingTriangleFinder.cpp.

References nrOfPoints.

Referenced by advance(), advanceBToRightChain(), findVertexCOnSideB(), gamma(), and intersects().

Here is the caller graph for this function:



7.22.3.33 void MinEnclosingTriangleFinder::updateMinEnclosingTriangle (vector< Point2f > & *minEnclosingTriangle*, double & *minEnclosingTriangleArea*) [private]

Update the current minimum area enclosing triangle if the newly obtained one has a smaller area.

Parameters

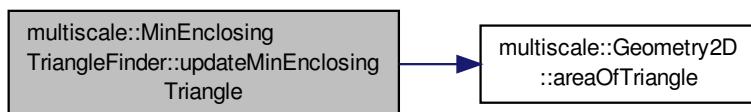
<i>minEnclosingTriangle</i>	Minimum area triangle enclosing the given polygon
<i>minEnclosingTriangleArea</i>	Area of the minimum area triangle enclosing the given polygon

Definition at line 193 of file MinEnclosingTriangleFinder.cpp.

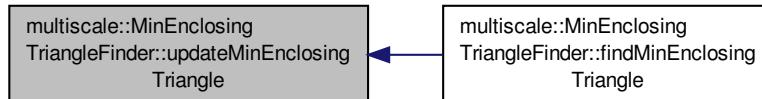
References area, multiscale::Geometry2D::areaOfTriangle(), vertexA, vertexB, and vertexC.

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.34 void MinEnclosingTriangleFinder::updateSideB() [private]

Set side B if tangency for side B was obtained.

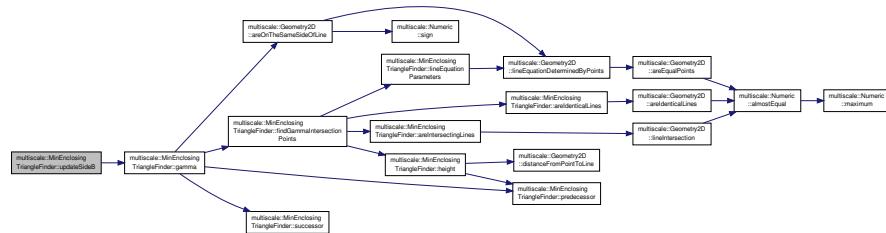
See paper for more details

Definition at line 155 of file MinEnclosingTriangleFinder.cpp.

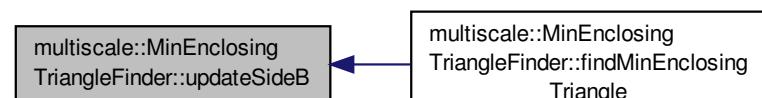
References b, ERR_SIDE_B_GAMMA, gamma(), polygon, sideBEndVertex, sideBStartVertex, VALIDATION_SIDE_B_TANGENT, and validationFlag.

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.35 void MinEnclosingTriangleFinder::updateSidesBA() [private]

Update sides B and possibly A if tangency for side B was not obtained.

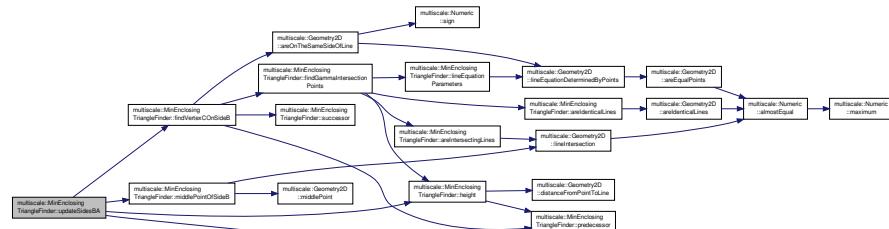
See paper for more details

Definition at line 137 of file MinEnclosingTriangleFinder.cpp.

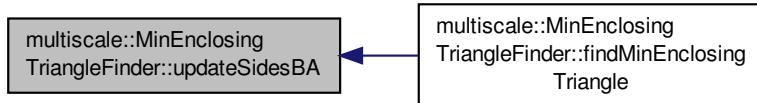
References a, b, findVertexCOnSideB(), height(), middlePointOfSideB(), polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideBEndVertex, sideBStartVertex, VALIDATION_SIDE_A_TANGENT, VALIDATION_SIDES_F-LUSH, and validationFlag.

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.3.36 void MinEnclosingTriangleFinder::updateSidesCA() [private]

Update sides A and C.

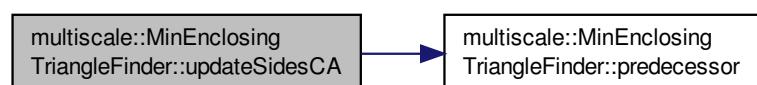
Side C will have as start and end vertices the polygon points "c" and "c-1" Side A will have as start and end vertices the polygon points "a" and "a-1"

Definition at line 129 of file MinEnclosingTriangleFinder.cpp.

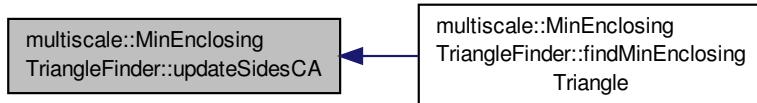
References a, c, polygon, predecessor(), sideAEndVertex, sideAStartVertex, sideCEndVertex, and sideCStartVertex.

Referenced by findMinEnclosingTriangle().

Here is the call graph for this function:



Here is the caller graph for this function:



7.22.4 Member Data Documentation

7.22.4.1 unsigned int multiscale::MinEnclosingTriangleFinder::a [private]

Index of point "a"; see paper for more details

Definition at line 61 of file MinEnclosingTriangleFinder.hpp.

Referenced by findVertexCOnSideB(), gamma(), initialiseAlgorithmVariables(), isNotBTangency(), isValidMinimalTriangle(), lineEquationParameters(), moveAIfLowAndBIfHigh(), searchForBTangency(), updateSidesBA(), and updateSidesCA().

7.22.4.2 double multiscale::MinEnclosingTriangleFinder::area [private]

Area of the current considered enclosing triangle

Definition at line 59 of file MinEnclosingTriangleFinder.hpp.

Referenced by updateMinEnclosingTriangle().

7.22.4.3 unsigned int multiscale::MinEnclosingTriangleFinder::b [private]

Index of point "b"; see paper for more details

Definition at line 62 of file MinEnclosingTriangleFinder.hpp.

Referenced by advanceBToRightChain(), initialiseAlgorithmVariables(), isNotBTangency(), isValidMinimalTriangle(), lineEquationParameters(), moveAIfLowAndBIfHigh(), searchForBTangency(), updateSideB(), and updateSidesBA().

7.22.4.4 unsigned int multiscale::MinEnclosingTriangleFinder::c [private]

Index of point "c"; see paper for more details

Definition at line 63 of file MinEnclosingTriangleFinder.hpp.

Referenced by findMinEnclosingTriangle(), findVertexCOnSideB(), gamma(), height(), initialiseAlgorithmVariables(), intersects(), lineEquationParameters(), and updateSidesCA().

7.22.4.5 unsigned int multiscale::MinEnclosingTriangleFinder::nrOfPoints [private]

Number of points defining the polygon

Definition at line 65 of file MinEnclosingTriangleFinder.hpp.

Referenced by findMinEnclosingTriangle(), initialiseAlgorithmVariables(), predecessor(), and successor().

7.22.4.6 `vector<Point2f> multiscale::MinEnclosingTriangleFinder::polygon` [private]

Polygon for which the minimum area enclosing triangle is computed

Definition at line 67 of file MinEnclosingTriangleFinder.hpp.

Referenced by `find()`, `findVertexCOnSideB()`, `gamma()`, `height()`, `initialiseAlgorithmVariables()`, `initialiseConvexPolygon()`, `intersects()`, `intersectsAbove()`, `intersectsBelow()`, `isValidMinimalTriangle()`, `updateSideB()`, `updateSidesBA()`, and `updateSidesCA()`.

7.22.4.7 `Point2f multiscale::MinEnclosingTriangleFinder::sideAEndVertex` [private]

Ending vertex for side A of triangle

Definition at line 51 of file MinEnclosingTriangleFinder.hpp.

Referenced by `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, `updateSidesBA()`, and `updateSidesCA()`.

7.22.4.8 `Point2f multiscale::MinEnclosingTriangleFinder::sideAStartVertex` [private]

Starting vertex for side A of triangle

Definition at line 50 of file MinEnclosingTriangleFinder.hpp.

Referenced by `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, `updateSidesBA()`, and `updateSidesCA()`.

7.22.4.9 `Point2f multiscale::MinEnclosingTriangleFinder::sideBEndVertex` [private]

Ending vertex for side B of triangle

Definition at line 54 of file MinEnclosingTriangleFinder.hpp.

Referenced by `findVertexCOnSideB()`, `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, `updateSideB()`, and `updateSidesBA()`.

7.22.4.10 `Point2f multiscale::MinEnclosingTriangleFinder::sideBStartVertex` [private]

Starting vertex for side B of triangle

Definition at line 53 of file MinEnclosingTriangleFinder.hpp.

Referenced by `findVertexCOnSideB()`, `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, `updateSideB()`, and `updateSidesBA()`.

7.22.4.11 `Point2f multiscale::MinEnclosingTriangleFinder::sideCEndVertex` [private]

Ending vertex for side C of triangle

Definition at line 57 of file MinEnclosingTriangleFinder.hpp.

Referenced by `findVertexCOnSideB()`, `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, and `updateSidesCA()`.

7.22.4.12 `Point2f multiscale::MinEnclosingTriangleFinder::sideCStartVertex` [private]

Starting vertex for side C of triangle

Definition at line 56 of file MinEnclosingTriangleFinder.hpp.

Referenced by `findVertexCOnSideB()`, `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, and `updateSidesCA()`.

7.22.4.13 unsigned int multiscale::MinEnclosingTriangleFinder::validationFlag [private]

Validation flag can take the following values:

- `VALIDATION_SIDE_A_TANGENT`;
- `VALIDATION_SIDE_B_TANGENT`;
- `VALIDATION_SIDES_FLUSH`.

Definition at line 40 of file `MinEnclosingTriangleFinder.hpp`.

Referenced by `isValidMinimalTriangle()`, `updateSideB()`, and `updateSidesBA()`.

7.22.4.14 Point2f multiscale::MinEnclosingTriangleFinder::vertexA [private]

Vertex A of the current considered enclosing triangle

Definition at line 46 of file `MinEnclosingTriangleFinder.hpp`.

Referenced by `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, and `updateMinEnclosingTriangle()`.

7.22.4.15 Point2f multiscale::MinEnclosingTriangleFinder::vertexB [private]

Vertex B of the current considered enclosing triangle

Definition at line 47 of file `MinEnclosingTriangleFinder.hpp`.

Referenced by `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, and `updateMinEnclosingTriangle()`.

7.22.4.16 Point2f multiscale::MinEnclosingTriangleFinder::vertexC [private]

Vertex C of the current considered enclosing triangle

Definition at line 48 of file `MinEnclosingTriangleFinder.hpp`.

Referenced by `isLocalMinimalTriangle()`, `isValidMinimalTriangle()`, `middlePointOfSideB()`, and `updateMinEnclosingTriangle()`.

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

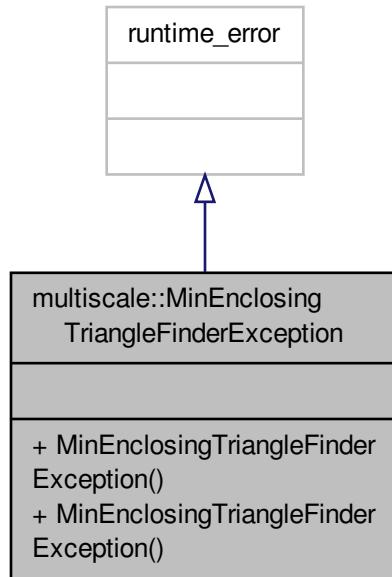
- `include/multiscale/util/MinEnclosingTriangleFinder.hpp`
- `src/multiscale/util/MinEnclosingTriangleFinder.cpp`

7.23 multiscale::MinEnclosingTriangleFinderException Class Reference

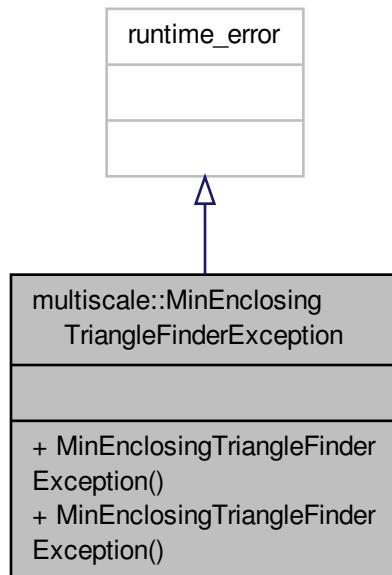
Exception class for the minimum area enclosing triangle module.

```
#include <MinEnclosingTriangleFinderException.hpp>
```

Inheritance diagram for multiscale::MinEnclosingTriangleFinderException:



Collaboration diagram for multiscale::MinEnclosingTriangleFinderException:



Public Member Functions

- [MinEnclosingTriangleFinderException \(const string &msg\)](#)
- [MinEnclosingTriangleFinderException \(const char *msg\)](#)

7.23.1 Detailed Description

Exception class for the minimum area enclosing triangle module.

Definition at line 13 of file MinEnclosingTriangleFinderException.hpp.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 multiscale::MinEnclosingTriangleFinderException::MinEnclosingTriangleFinderException (const string & msg) [inline]

Definition at line 17 of file MinEnclosingTriangleFinderException.hpp.

7.23.2.2 multiscale::MinEnclosingTriangleFinderException::MinEnclosingTriangleFinderException (const char * msg) [inline]

Definition at line 18 of file MinEnclosingTriangleFinderException.hpp.

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

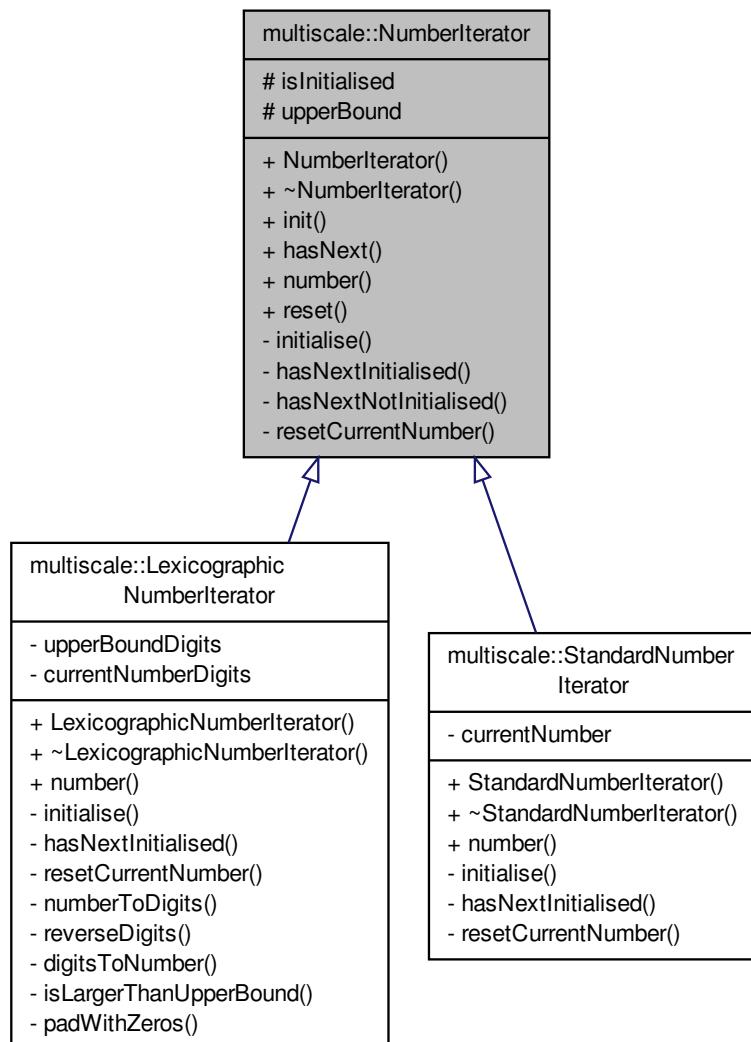
- [include/multiscale/exception/MinEnclosingTriangleFinderException.hpp](#)

7.24 multiscale::NumberIterator Class Reference

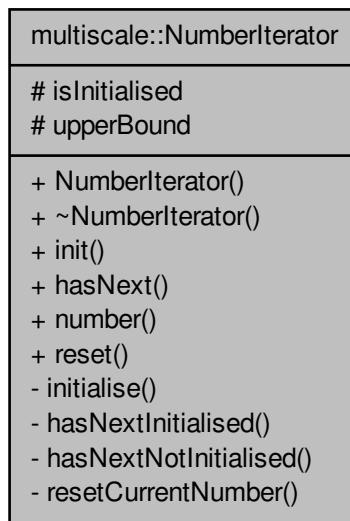
Abstract class representing a number iterator.

```
#include <NumberIterator.hpp>
```

Inheritance diagram for multiscale::NumberIterator:



Collaboration diagram for multiscale::NumberIterator:



Public Member Functions

- `NumberIterator (unsigned int upperBound)`
- virtual `~NumberIterator ()`
- void `init (unsigned int upperBound)`

Initialise the iterator considering the given upper bound.
- bool `hasNext ()`

Check if there is a next number.
- virtual unsigned int `number ()=0`

Get the number pointed by the iterator.
- void `reset ()`

Reset the iterator.

Protected Attributes

- bool `isInitialised`
- unsigned int `upperBound`

Private Member Functions

- virtual void `initialise ()=0`

Initialisation of the members of the class.
- virtual bool `hasNextInitialised ()=0`

Check if there is a next number when in initialised state.
- bool `hasNextNotInitialised ()`

Check if there is a next number when in not initialised state.
- virtual void `resetCurrentNumber ()=0`

Reset the current number to its initial value.

7.24.1 Detailed Description

Abstract class representing a number iterator.

Definition at line 7 of file NumberIterator.hpp.

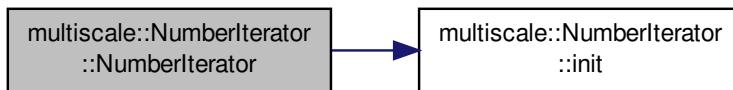
7.24.2 Constructor & Destructor Documentation

7.24.2.1 NumberIterator::NumberIterator (*unsigned int upperBound*)

Definition at line 6 of file NumberIterator.cpp.

References init().

Here is the call graph for this function:



7.24.2.2 virtual multiscale::NumberIterator::~NumberIterator () [inline], [virtual]

Definition at line 17 of file NumberIterator.hpp.

7.24.3 Member Function Documentation

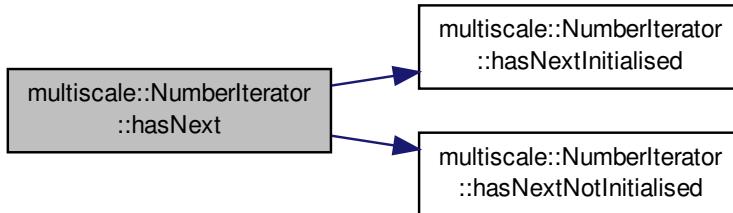
7.24.3.1 bool NumberIterator::hasNext ()

Check if there is a next number.

Definition at line 14 of file NumberIterator.cpp.

References hasNextInitialised(), hasNextNotInitialised(), and isInitialised.

Here is the call graph for this function:



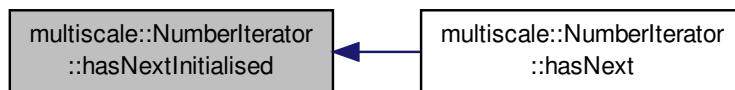
7.24.3.2 virtual bool multiscale::NumberIterator::hasNextInitialised() [private], [pure virtual]

Check if there is a next number when in initialised state.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

Referenced by [hasNext\(\)](#).

Here is the caller graph for this function:



7.24.3.3 bool NumberIterator::hasNextNotInitialised() [private]

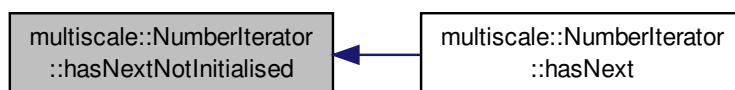
Check if there is a next number when in not initialised state.

Definition at line 28 of file [NumberIterator.cpp](#).

References [isInitialised](#).

Referenced by [hasNext\(\)](#).

Here is the caller graph for this function:

7.24.3.4 void NumberIterator::init(unsigned int *upperBound*)

Initialise the iterator considering the given upper bound.

Parameters

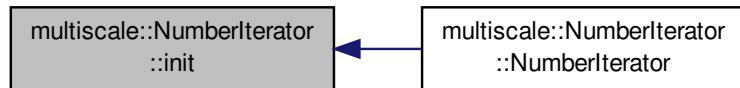
<i>upperBound</i>	The upper bound
-------------------	-----------------

Definition at line 10 of file [NumberIterator.cpp](#).

References [upperBound](#).

Referenced by [NumberIterator\(\)](#).

Here is the caller graph for this function:



7.24.3.5 virtual void multiscale::NumberIterator::initialise() [private], [pure virtual]

Initialisation of the members of the class.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

7.24.3.6 virtual unsigned int multiscale::NumberIterator::number() [pure virtual]

Get the number pointed by the iterator.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

7.24.3.7 void NumberIterator::reset()

Reset the iterator.

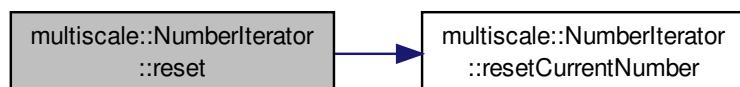
Reset the iterator such that it is not initialised and the value of the current number is reset to its initial value

Definition at line 22 of file NumberIterator.cpp.

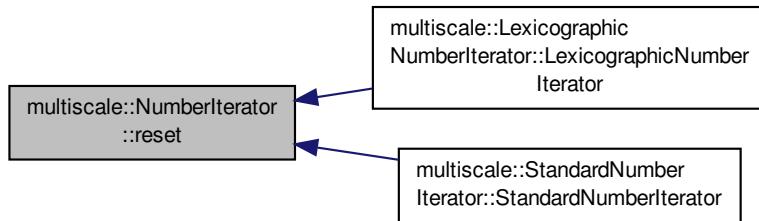
References [isInitialised](#), and [resetCurrentNumber\(\)](#).

Referenced by [multiscale::LexicographicNumberIterator::LexicographicNumberIterator\(\)](#), and [multiscale::StandardNumberIterator::StandardNumberIterator\(\)](#).

Here is the call graph for this function:



Here is the caller graph for this function:



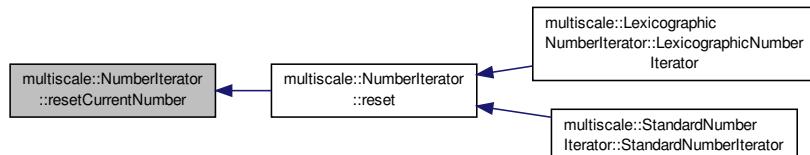
7.24.3.8 virtual void multiscale::NumberIterator::resetCurrentNumber() [private], [pure virtual]

Reset the current number to its initial value.

Implemented in [multiscale::LexicographicNumberIterator](#), and [multiscale::StandardNumberIterator](#).

Referenced by [reset\(\)](#).

Here is the caller graph for this function:



7.24.4 Member Data Documentation

7.24.4.1 bool multiscale::NumberIterator::isInitialised [protected]

Flag for checking if the iterator was initialised

Definition at line 11 of file [NumberIterator.hpp](#).

Referenced by [hasNext\(\)](#), [hasNextNotInitialised\(\)](#), and [reset\(\)](#).

7.24.4.2 unsigned int multiscale::NumberIterator::upperBound [protected]

Upper bound of the iterator

Definition at line 12 of file [NumberIterator.hpp](#).

Referenced by [multiscale::StandardNumberIterator::hasNextInitialised\(\)](#), [init\(\)](#), [multiscale::LexicographicNumberIterator::initialise\(\)](#), and [multiscale::LexicographicNumberIterator::padWithZeros\(\)](#).

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

- [include/multiscale/util/NumberIterator.hpp](#)

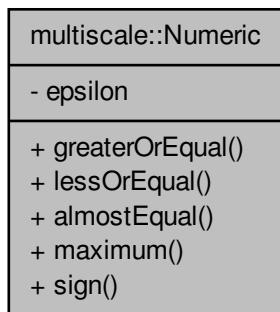
- src/multiscale/util/NumberIterator.cpp

7.25 multiscale::Numeric Class Reference

Class for manipulating numbers (shorts, ints, floats, doubles etc.)

```
#include <Numeric.hpp>
```

Collaboration diagram for multiscale::Numeric:



Static Public Member Functions

- static bool [greaterOrEqual](#) (double number1, double number2)

Check if the first number is greater than or equal to the second number.
- static bool [lessOrEqual](#) (double number1, double number2)

Check if the first number is less than or equal to the second number.
- static bool [almostEqual](#) (double number1, double number2)

Check if the two numbers are equal (almost)
- static double [maximum](#) (double number1, double number2, double number3)

Return the maximum of the provided numbers.
- static int [sign](#) (double number)

Return the sign of the number.

Static Private Attributes

- static double [epsilon](#) = 1E-5

7.25.1 Detailed Description

Class for manipulating numbers (shorts, ints, floats, doubles etc.)

Definition at line 14 of file Numeric.hpp.

7.25.2 Member Function Documentation

7.25.2.1 bool Numeric::almostEqual (double number1, double number2) [static]

Check if the two numbers are equal (almost)

The expression for determining if two real numbers are equal is: if ($\text{Abs}(x - y) \leq \text{EPSILON} * \text{Max}(1.0f, \text{Abs}(x), \text{Abs}(y))$).

Parameters

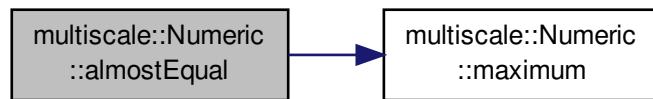
<i>number1</i>	First number
<i>number2</i>	Second number

Definition at line 17 of file Numeric.cpp.

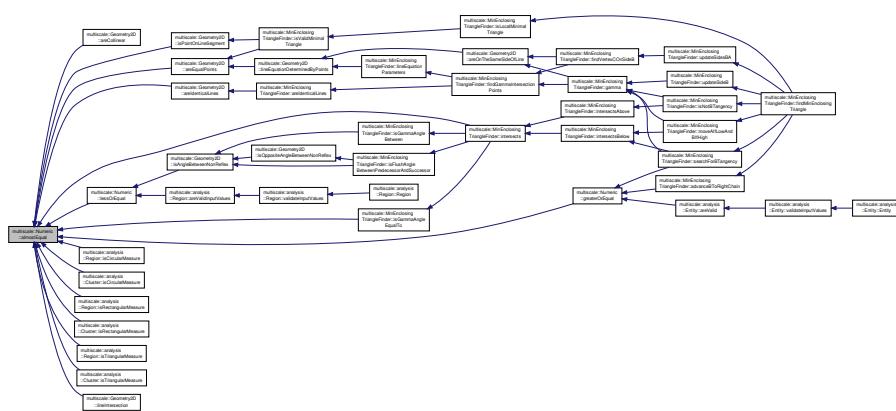
References epsilon, and maximum().

Referenced by multiscale::Geometry2D::areCollinear(), multiscale::Geometry2D::areEqualPoints(), multiscale::Geometry2D::areIdenticalLines(), greaterOrEqual(), multiscale::MinEnclosingTriangleFinder::intersects(), multiscale::analysis::Region::isCircularMeasure(), multiscale::analysis::Cluster::isCircularMeasure(), multiscale::MinEnclosingTriangleFinder::isGammaAngleEqualTo(), multiscale::Geometry2D::isPointOnLineSegment(), multiscale::analysis::Region::isRectangularMeasure(), multiscale::analysis::Cluster::isRectangularMeasure(), multiscale::analysis::Region::isTriangularMeasure(), multiscale::analysis::Cluster::isTriangularMeasure(), lessOrEqual(), and multiscale::Geometry2D::lineIntersection().

Here is the call graph for this function:



Here is the caller graph for this function:



7.25.2.2 bool Numeric::greaterOrEqual (double *number1*, double *number2*) [static]

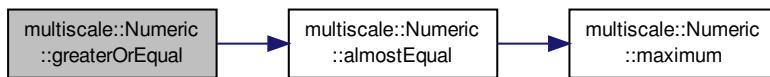
Check if the first number is greater than or equal to the second number.

Definition at line 9 of file Numeric.cpp.

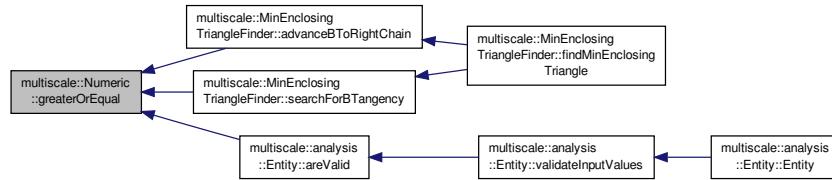
References almostEqual().

Referenced by multiscale::MinEnclosingTriangleFinder::advanceBToRightChain(), multiscale::analysis::Entity::isValid(), and multiscale::MinEnclosingTriangleFinder::searchForBTangency().

Here is the call graph for this function:



Here is the caller graph for this function:



7.25.2.3 bool Numeric::lessOrEqual (double *number1*, double *number2*) [static]

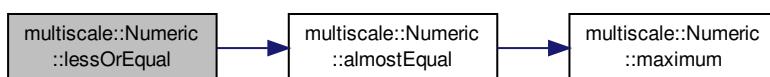
Check if the first number is less than or equal to the second number.

Definition at line 13 of file Numeric.cpp.

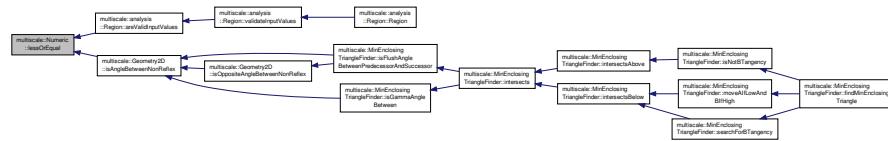
References almostEqual().

Referenced by multiscale::analysis::Region::isValidInputValues(), and multiscale::Geometry2D::isAngleBetweenNonReflex().

Here is the call graph for this function:



Here is the caller graph for this function:



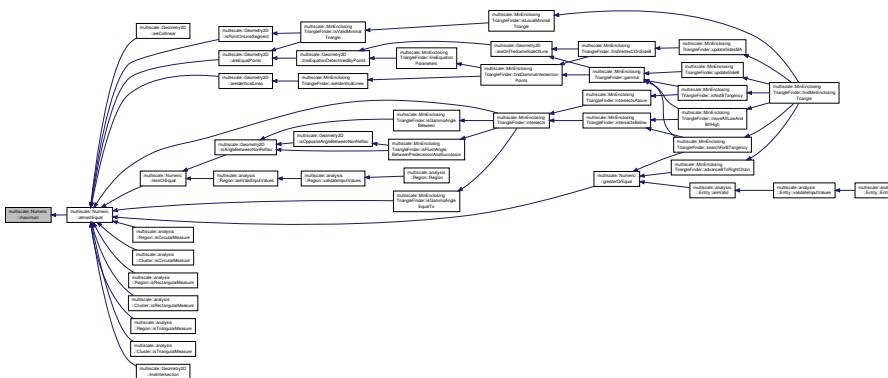
7.25.2.4 double Numeric::maximum (double *number1*, double *number2*, double *number3*) [static]

Return the maximum of the provided numbers.

Definition at line 21 of file Numeric.cpp.

Referenced by almostEqual().

Here is the caller graph for this function:



7.25.2.5 int Numeric::sign (double *number*) [static]

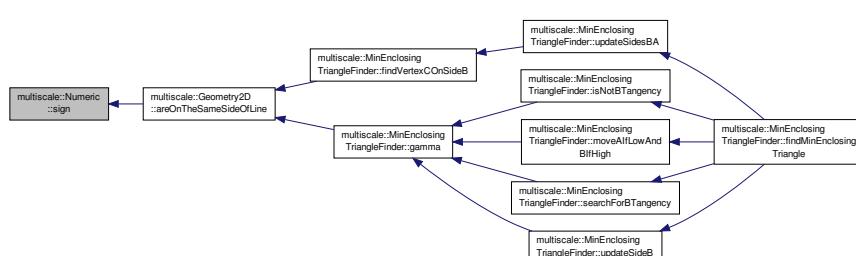
Return the sign of the number.

The sign function returns: -1, if number < 0 +1, if number > 0 0, otherwise

Definition at line 25 of file Numeric.cpp.

Referenced by multiscale::Geometry2D::areOnTheSameSideOfLine().

Here is the caller graph for this function:



7.25.3 Member Data Documentation

7.25.3.1 double Numeric::epsilon = 1E-5 [static], [private]

Value of epsilon used to compare two real numbers

Definition at line 18 of file Numeric.hpp.

Referenced by almostEqual().

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

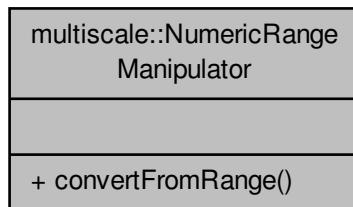
- include/multiscale/util/Numeric.hpp
- src/multiscale/util/Numeric.cpp

7.26 multiscale::NumericRangeManipulator Class Reference

Operations for ranges of numeric values.

```
#include <NumericRangeManipulator.hpp>
```

Collaboration diagram for multiscale::NumericRangeManipulator:



Static Public Member Functions

- template<class T , class U >
static U [convertFromRange](#) (T oldRangeMin, T oldRangeMax, U newRangeMin, U newRangeMax, T oldValue)
Convert a value from an old range to a new one.

7.26.1 Detailed Description

Operations for ranges of numeric values.

Definition at line 7 of file NumericRangeManipulator.hpp.

7.26.2 Member Function Documentation

7.26.2.1 template<class T , class U > static U multiscale::NumericRangeManipulator::convertFromRange (T oldRangeMin, T oldRangeMax, U newRangeMin, U newRangeMax, T oldValue) [inline], [static]

Convert a value from an old range to a new one.

Parameters

<i>oldRangeMin</i>	The minimum of the old range
<i>oldRangeMax</i>	The maximum of the old range
<i>newRangeMin</i>	The minimum of the new range
<i>newRangeMax</i>	The maximum of the new range
<i>oldValue</i>	The old value

Definition at line 20 of file NumericRangeManipulator.hpp.

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

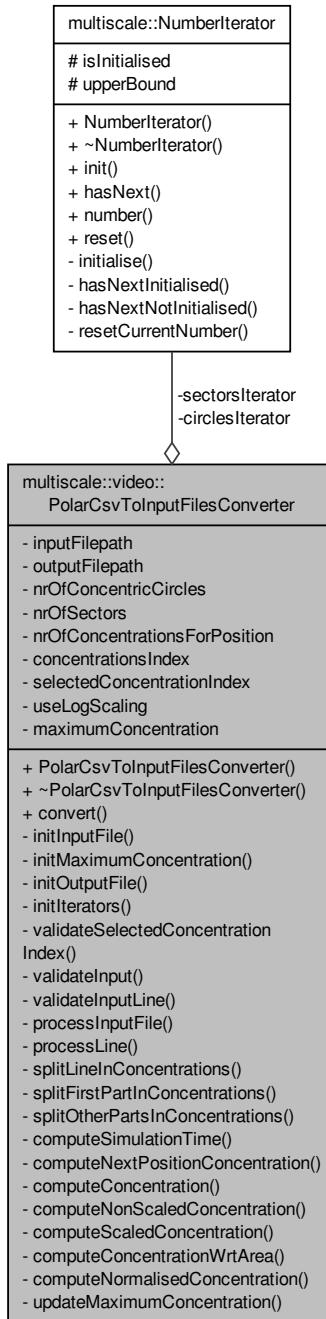
- include/multiscale/util/[NumericRangeManipulator.hpp](#)

7.27 multiscale::video::PolarCsvToInputFilesConverter Class Reference

Csv file to input file converter considering polar coordinates.

```
#include <PolarCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::video::PolarCsvToInputFilesConverter:



Public Member Functions

- `PolarCsvToInputFilesConverter` (const string &`inputfilepath`, const string &`outputfilepath`, unsigned int `nrOfConcentricCircles`, unsigned int `nrOfSectors`, unsigned int `nrOfConcentrationsForPosition`, unsigned int `selectedConcentrationIndex`, bool `useLogScaling`, `NumberIteratorType` `numberIteratorType`)
- `~PolarCsvToInputFilesConverter ()`
- `void convert ()`

Start the conversion.

Private Member Functions

- void `initInputFile` (`ifstream &fin`)

Initialise the input file stream over the given input file.
- void `initMaximumConcentration` (`ifstream &fin`)

Compute the value of member maximum concentration.
- void `initOutputFile` (`ofstream &fout`, `unsigned int index`, `double &simulationTime`)

Initialise the output file with the given index and simulation time.
- void `initIterators` (`const NumberIteratorType &numberIteratorType`)

Initialise the iterators considering the given number iterator type.
- void `validateSelectedConcentrationIndex` ()

Validate the selected concentration index in case of more than one concentration for each position.
- void `validateInput` (`ifstream &fin`)

Validate the input.
- void `validateInputLine` (`const string &line`, `unsigned int lineNumber`)

Validate the provided line identified by a line number.
- void `processInputFile` (`ifstream &fin`)

Process the input file.
- void `processLine` (`const string &line`, `unsigned int outputIndex`)

Process the provided line.
- `vector< double > splitLineInConcentrations` (`const string &line`, `double &simulationTime`)

Split the line in concentrations.
- void `splitFirstPartInConcentrations` (`vector< double > &concentrations`, `const vector< string > &tokens`, `unsigned int circleIndex`)

Split first part of the line (i.e. part representing the origin) into concentrations.
- void `splitOtherPartsInConcentrations` (`vector< double > &concentrations`, `const vector< string > &tokens`, `unsigned int circleIndex`)

Split other parts of the line (i.e. non-first part) into concentrations.
- double `computeSimulationTime` (`const string &token`)

Compute the simulation time from the given token and check if it is valid.
- double `computeNextPositionConcentration` (`unsigned int circleIndex`, `int concentrationIndex`, `const vector< string > &tokens`)

Compute the concentration for the next position.
- double `computeConcentration` (`const string &concentration`, `int circleIndex`)

Compute the concentration from the given string considering the index of the current concentric circle.
- double `computeNonScaledConcentration` (`const string &concentration`, `int circleIndex`)

Compute the non-scaled concentration from the given string considering the index of the current concentric circle.
- double `computeScaledConcentration` (`const string &concentration`, `int circleIndex`)

Compute the scaled concentration from the given string considering the index of the current concentric circle.
- double `computeConcentrationWrtArea` (`double amount`, `int circleIndex`)

Compute the concentration wrt. the area of the annular sector.
- double `computeNormalisedConcentration` (`double concentration`, `int circleIndex`)

Normalise the concentration considering the index of the current concentric circle by dividing it to the maximum concentration.
- void `updateMaximumConcentration` (`const string &line`, `double &maximumConcentration`)

Update the maximum concentration if the values from the given line are greater than it.

Private Attributes

- string `inputFilepath`
- string `outputFilepath`
- unsigned int `nrOfConcentricCircles`
- unsigned int `nrOfSectors`
- unsigned int `nrOfConcentrationsForPosition`
- unsigned int `concentrationsIndex`
- unsigned int `selectedConcentrationIndex`
- bool `useLogScaling`
- double `maximumConcentration`
- `NumberIterator * circlesIterator`
- `NumberIterator * sectorsIterator`

7.27.1 Detailed Description

Csv file to input file converter considering polar coordinates.

Definition at line 33 of file PolarCsvToInputFilesConverter.hpp.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 PolarCsvToInputFilesConverter::PolarCsvToInputFilesConverter (const string & *inputFilepath*, const string & *outputFilepath*, unsigned int *nrOfConcentricCircles*, unsigned int *nrOfSectors*, unsigned int *nrOfConcentrationsForPosition*, unsigned int *selectedConcentrationIndex*, bool *useLogScaling*, NumberIteratorType *numberIteratorType*)

Definition at line 20 of file PolarCsvToInputFilesConverter.cpp.

7.27.2.2 PolarCsvToInputFilesConverter::~PolarCsvToInputFilesConverter ()

Definition at line 44 of file PolarCsvToInputFilesConverter.cpp.

7.27.3 Member Function Documentation

7.27.3.1 double PolarCsvToInputFilesConverter::computeConcentration (const string & *concentration*, int *circleIndex*) [inline], [private]

Compute the concentration from the given string considering the index of the current concentric circle.

Parameters

<code>concentration</code>	String representing the concentration
<code>circleIndex</code>	Index of the concentric circle

Definition at line 306 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.2 double PolarCsvToInputFilesConverter::computeConcentrationWrtArea (double *amount*, int *circleIndex*) [inline], [private]

Compute the concentration wrt. the area of the annular sector.

Parameters

<i>amount</i>	Amount in annular sector
<i>circleIndex</i>	Index of the concentric circle which will be used to determine the area

Definition at line 332 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.3 double PolarCsvToInputFilesConverter::computeNextPositionConcentration (unsigned int *circleIndex*, int *concentrationIndex*, const vector< string > & *t*okens) [inline], [private]

Compute the concentration for the next position.

Parameters

<i>circleIndex</i>	Index of the current concentric circle
<i>concentration-Index</i>	Index of the current concentration from the vector of tokens
<i>t</i> okens	Vector of tokens

Definition at line 277 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.4 double PolarCsvToInputFilesConverter::computeNonScaledConcentration (const string & *concentration*, int *circleIndex*) [inline], [private]

Compute the non-scaled concentration from the given string considering the index of the current concentric circle.

Parameters

<i>concentration</i>	String representing the concentration
<i>circleIndex</i>	Index of the concentric circle

Definition at line 312 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.5 double PolarCsvToInputFilesConverter::computeNormalisedConcentration (double *concentration*, int *circleIndex*) [inline], [private]

Normalise the concentration considering the index of the current concentric circle by dividing it to the maximum concentration.

Parameters

<i>concentration</i>	The concentration
<i>circleIndex</i>	Index of the concentric circle

Definition at line 336 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.6 double PolarCsvToInputFilesConverter::computeScaledConcentration (const string & *concentration*, int *circleIndex*) [inline], [private]

Compute the scaled concentration from the given string considering the index of the current concentric circle.

Compute the scaled concentration from the given string by applying a logit transformation to it

Parameters

<i>concentration</i>	String representing the concentration
<i>circleIndex</i>	Index of the concentric circle

Definition at line 318 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.7 double PolarCsvToInputFilesConverter::computeSimulationTime (const string & *token*) [inline], [private]

Compute the simulation time from the given token and check if it is valid.

Parameters

<i>token</i>	Token (string)
--------------	----------------

Definition at line 267 of file PolarCsvToInputFilesConverter.cpp.

References ERR_NEG_SIM_TIME.

7.27.3.8 void PolarCsvToInputFilesConverter::convert ()

Start the conversion.

Definition at line 49 of file PolarCsvToInputFilesConverter.cpp.

Referenced by main().

Here is the caller graph for this function:



7.27.3.9 void PolarCsvToInputFilesConverter::initInputModule (ifstream & *fin*) [private]

Initialise the input file stream over the given input file.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 62 of file PolarCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.27.3.10 void PolarCsvToInputFilesConverter::initIterators (const NumberIteratorType & *numberIteratorType*) [private]

Initialise the iterators considering the given number iterator type.

Parameters

<i>numberIterator- Type</i>	The type of the number iterator
---------------------------------	---------------------------------

Definition at line 111 of file PolarCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

7.27.3.11 void PolarCsvToInputFilesConverter::initMaximumConcentration (ifstream & *fin*) [private]

Compute the value of member maximum concentration.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 70 of file PolarCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.27.3.12 void PolarCsvToInputFilesConverter::initOutputFile (ofstream & *fout*, unsigned int *index*, double & *simulationTime*) [private]

Initialise the output file with the given index and simulation time.

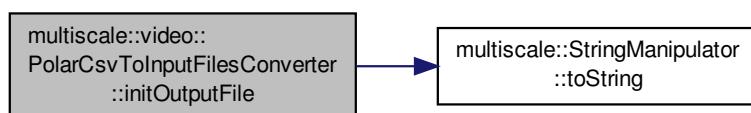
Parameters

<i>fout</i>	Output file stream
<i>index</i>	Index of the output file
<i>simulationTime</i>	Simulation time

Definition at line 94 of file PolarCsvToInputFilesConverter.cpp.

References OUTPUT_EXTENSION, OUTPUT_FILE_SEPARATOR, OUTPUT_SEPARATOR, and multiscale::StringManipulator::toString().

Here is the call graph for this function:



7.27.3.13 void PolarCsvToInputFilesConverter::processInputFile (ifstream & *fin*) [private]

Process the input file.

Read the concentrations and normalise them if it is the case.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 178 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.14 void PolarCsvToInputFilesConverter::processLine (const string & *line*, unsigned int *outputIndex*) [private]

Process the provided line.

Parameters

<i>line</i>	Line
<i>outputIndex</i>	Index integrated in the name of the output file

Definition at line 193 of file PolarCsvToInputFilesConverter.cpp.

References OUTPUT_SEPARATOR.

7.27.3.15 void PolarCsvToInputFilesConverter::splitFirstPartInConcentrations (vector< double > & *concentrations*, const vector< string > & *tokens*, unsigned int *circleIndex*) [private]

Split first part of the line (i.e. part representing the origin) into concentrations.

Parameters

<i>concentrations</i>	Concentrations extracted from tokens
<i>tokens</i>	Tokens representing the line
<i>circleIndex</i>	Index of the current concentric circle

Definition at line 237 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.16 vector< double > PolarCsvToInputFilesConverter::splitLineInConcentrations (const string & *line*, double & *simulationTime*) [private]

Split the line in concentrations.

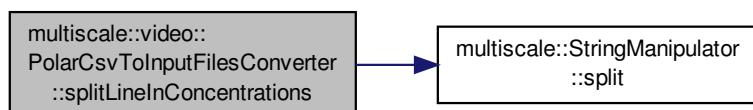
Parameters

<i>line</i>	Line
<i>simulationTime</i>	Simulation time associated with the line

Definition at line 212 of file PolarCsvToInputFilesConverter.cpp.

References INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.27.3.17 void PolarCsvToInputFilesConverter::splitOtherPartsInConcentrations (`vector< double > & concentrations`, const `vector< string > & tokens`, `unsigned int circleIndex`) [private]

Split other parts of the line (i.e. non-first part) into concentrations.

Parameters

<code>concentrations</code>	Concentrations extracted from tokens
<code>tokens</code>	Tokens representing the line
<code>circleIndex</code>	Index of the current concentric circle

Definition at line 251 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.18 void PolarCsvToInputFilesConverter::updateMaximumConcentration (`const string & line`, `double & maximumConcentration`) [private]

Update the maximum concentration if the values from the given line are greater than it.

Parameters

<code>line</code>	Line from input file
<code>maximum-Concentration</code>	The maximum concentration

Definition at line 340 of file PolarCsvToInputFilesConverter.cpp.

7.27.3.19 void PolarCsvToInputFilesConverter::validateInput (`ifstream & fin`) [private]

Validate the input.

Parameters

<code>fin</code>	Input file stream
------------------	-------------------

Definition at line 134 of file PolarCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.27.3.20 void PolarCsvToInputFilesConverter::validateInputLine (`const string & line`, `unsigned int lineNumber`) [private]

Validate the provided line identified by a line number.

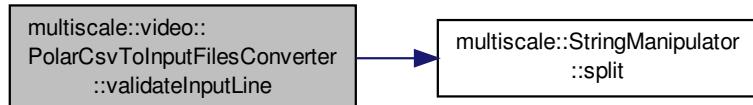
Parameters

<code>line</code>	Line from input file
<code>lineNumber</code>	Number of the line

Definition at line 158 of file PolarCsvToInputFilesConverter.cpp.

References ERR_INVALID_VALUE_LINE, ERR_INVALID_VALUE_TOKEN, ERR_NR_CONCENTRATIONS, INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.27.3.21 void PolarCsvToInputFilesConverter::validateSelectedConcentrationIndex () [private]

Validate the selected concentration index in case of more than one concentration for each position.

Definition at line 128 of file PolarCsvToInputFilesConverter.cpp.

References ERR_SELECTED_CONCENTRATION_INDEX.

7.27.4 Member Data Documentation

7.27.4.1 NumberIterator* multiscale::video::PolarCsvToInputFilesConverter::circlesIterator [private]

Iterator over the number of concentric circles

Definition at line 57 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.2 unsigned int multiscale::video::PolarCsvToInputFilesConverter::concentrationsIndex [private]

Index of the current concentration

Definition at line 44 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.3 string multiscale::video::PolarCsvToInputFilesConverter::inputFilepath [private]

Path to the input file

Definition at line 37 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.4 double multiscale::video::PolarCsvToInputFilesConverter::maximumConcentration [private]

The maximum concentration in the input file

Definition at line 55 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.5 unsigned int multiscale::video::PolarCsvToInputFilesConverter::nrOfConcentrationsForPosition [private]

Number of concentrations for each position

Definition at line 42 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.6 `unsigned int multiscale::video::PolarCsvToInputFilesConverter::nrOfConcentricCircles` [private]

Number of concentric circles

Definition at line 40 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.7 `unsigned int multiscale::video::PolarCsvToInputFilesConverter::nrOfSectors` [private]

Number of sectors

Definition at line 41 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.8 `string multiscale::video::PolarCsvToInputFilesConverter::outputFilepath` [private]

Path to the output file

Definition at line 38 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.9 `NumberIterator* multiscale::video::PolarCsvToInputFilesConverter::sectorsIterator` [private]

Iterator over the number of sectors

Definition at line 58 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.10 `unsigned int multiscale::video::PolarCsvToInputFilesConverter::selectedConcentrationIndex` [private]

Index of the concentration A in case the number of

concentrations for each position is greater than 1

$\text{finalConcentration} = A / (A_1 + A_2 + \dots + A_N)$, where N is the number of concentrations for each position

Definition at line 46 of file PolarCsvToInputFilesConverter.hpp.

7.27.4.11 `bool multiscale::video::PolarCsvToInputFilesConverter::useLogScaling` [private]

Flag for using logarithmic scaling for concentrations or not

Definition at line 53 of file PolarCsvToInputFilesConverter.hpp.

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

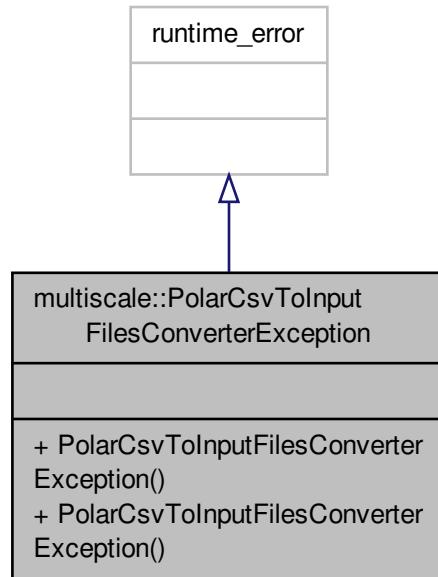
- include/multiscale/video/circular/[PolarCsvToInputFilesConverter.hpp](#)
- src/multiscale/video/circular/[PolarCsvToInputFilesConverter.cpp](#)

7.28 multiscale::PolarCsvToInputFilesConverterException Class Reference

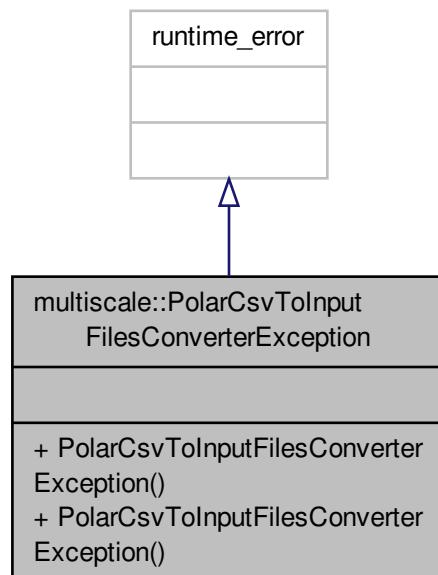
Exception class for the PolarCsvToInputFilesConverter class.

```
#include <PolarCsvToInputFilesConverterException.hpp>
```

Inheritance diagram for multiscale::PolarCsvToInputFilesConverterException:



Collaboration diagram for multiscale::PolarCsvToInputFilesConverterException:



Public Member Functions

- [PolarCsvToInputFilesConverterException \(const string &msg\)](#)
- [PolarCsvToInputFilesConverterException \(const char *msg\)](#)

7.28.1 Detailed Description

Exception class for the PolarCsvToInputFilesConverter class.

Definition at line 13 of file PolarCsvToInputFilesConverterException.hpp.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 [multiscale::PolarCsvToInputFilesConverterException::PolarCsvToInputFilesConverterException \(const string & msg \) \[inline\]](#)

Definition at line 17 of file PolarCsvToInputFilesConverterException.hpp.

7.28.2.2 [multiscale::PolarCsvToInputFilesConverterException::PolarCsvToInputFilesConverterException \(const char * msg \) \[inline\]](#)

Definition at line 18 of file PolarCsvToInputFilesConverterException.hpp.

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

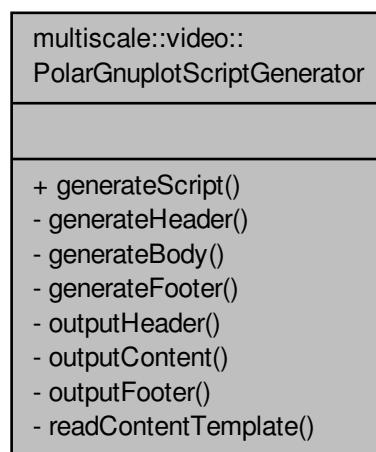
- [include/multiscale/exception/PolarCsvToInputFilesConverterException.hpp](#)

7.29 multiscale::video::PolarGnuplotScriptGenerator Class Reference

Gnuplot script generator from the provided annular sectors.

```
#include <PolarGnuplotScriptGenerator.hpp>
```

Collaboration diagram for multiscale::video::PolarGnuplotScriptGenerator:



Static Public Member Functions

- static void `generateScript` (const vector< `AnnularSector` > &annularSectors, double simulationTime, const string &outputFilepath)

Generate the script.

Static Private Member Functions

- static void `generateHeader` (ofstream &fout, const string &outputFilepath, double simulationTime)

Generate the header of the script.
- static void `generateBody` (const vector< `AnnularSector` > &annularSectors, ofstream &fout)

Generate the body/content of the script.
- static void `generateFooter` (ofstream &fout)

Generate the footer of the script.
- static void `outputHeader` (ifstream &fin, const string &outputFilename, double simulationTime, ofstream &fout)

Output the header of the script.
- static void `outputContent` (const vector< `AnnularSector` > &annularSectors, const string &contentTemplate, ofstream &fout)

Output the content of the script.
- static void `outputFooter` (ifstream &fin, ofstream &fout)

Output the footer of the script.
- static string `readContentTemplate` (ifstream &fin)

Read content template.

7.29.1 Detailed Description

Gnuplot script generator from the provided annular sectors.

Definition at line 31 of file PolarGnuplotScriptGenerator.hpp.

7.29.2 Member Function Documentation

7.29.2.1 void PolarGnuplotScriptGenerator::generateBody (const vector< `AnnularSector` > & `annularSectors`, ofstream & `fout`) [static], [private]

Generate the body/content of the script.

Parameters

<code>annularSectors</code>	Annular sectors
<code>fout</code>	Output file stream

Definition at line 40 of file PolarGnuplotScriptGenerator.cpp.

References CONTENT_IN.

7.29.2.2 void PolarGnuplotScriptGenerator::generateFooter (ofstream & `fout`) [static], [private]

Generate the footer of the script.

Parameters

<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 52 of file PolarGnuplotScriptGenerator.cpp.

References FOOTER_IN.

7.29.2.3 void PolarGnuplotScriptGenerator::generateHeader (ofstream & *fout*, const string & *outputFilepath*, double *simulationTime*) [static], [private]

Generate the header of the script.

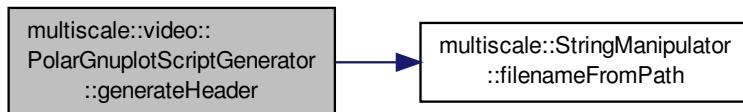
Parameters

<i>fout</i>	Output file stream
<i>outputFilepath</i>	Path to the output file
<i>simulationTime</i>	Simulation time

Definition at line 28 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::filenameFromPath(), and HEADER_IN.

Here is the call graph for this function:



7.29.2.4 void PolarGnuplotScriptGenerator::generateScript (const vector< AnnularSector > & *annularSectors*, double *simulationTime*, const string & *outputFilepath*) [static]

Generate the script.

Parameters

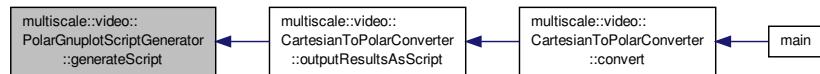
<i>annularSectors</i>	Annular sectors
<i>simulationTime</i>	Simulation time
<i>outputFilepath</i>	Path of the output file

Definition at line 14 of file PolarGnuplotScriptGenerator.cpp.

References GNPLOT_EXTENSION.

Referenced by multiscale::video::CartesianToPolarConverter::outputResultsAsScript().

Here is the caller graph for this function:



7.29.2.5 void PolarGnuplotScriptGenerator::outputContent (const vector< AnnularSector > & annularSectors, const string & contentTemplate, ofstream & fout) [static], [private]

Output the content of the script.

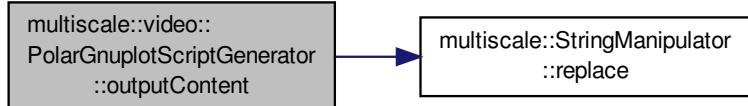
Parameters

<i>annularSectors</i>	Annular sectors
<i>contentTemplate</i>	Template used for generating output for each annular sector
<i>fout</i>	Output file stream

Definition at line 75 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE_CONTENT_CONCENTRATION, REPLACE_CONTENT_END_ANGLE, REPLACE_CONTENT_INDEX, REPLACE_CONTENT_RADIUS, and REPLACE_CONTENT_START_ANGLE.

Here is the call graph for this function:



7.29.2.6 void PolarGnuplotScriptGenerator::outputFooter (ifstream & fin, ofstream & fout) [static], [private]

Output the footer of the script.

Parameters

<i>fin</i>	Input file stream
<i>fout</i>	Output file stream

Definition at line 93 of file PolarGnuplotScriptGenerator.cpp.

7.29.2.7 void PolarGnuplotScriptGenerator::outputHeader (ifstream & fin, const string & outputFilename, double simulationTime, ofstream & fout) [static], [private]

Output the header of the script.

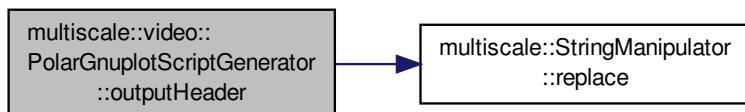
Parameters

<i>fin</i>	Input file stream
<i>outputFilename</i>	Name of the output file
<i>simulationTime</i>	Simulation time
<i>fout</i>	Output file stream

Definition at line 62 of file PolarGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE_HEADER_FILENAME, and REPLACE_HEADER_SIM_TIME.

Here is the call graph for this function:



7.29.2.8 string PolarGnuplotScriptGenerator::readContentTemplate (ifstream & *fin*) [static], [private]

Read content template.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 103 of file PolarGnuplotScriptGenerator.cpp.

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

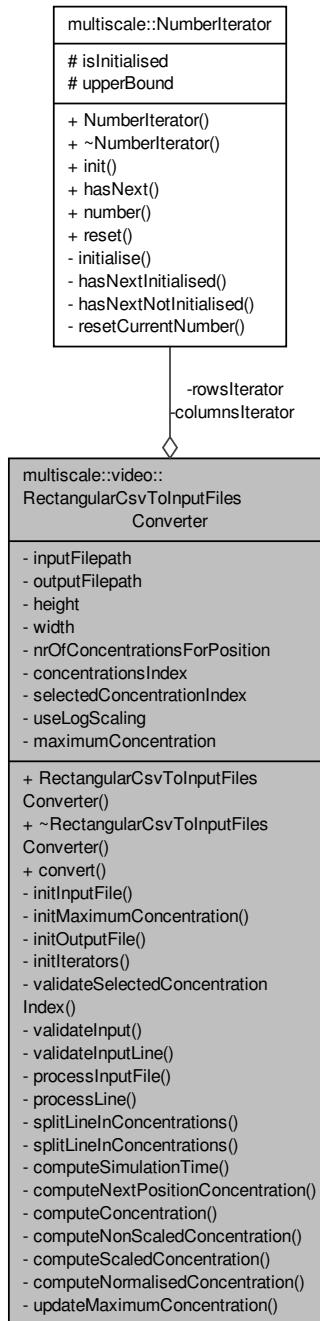
- include/multiscale/video/circular/[PolarGnuplotScriptGenerator.hpp](#)
- src/multiscale/video/circular/[PolarGnuplotScriptGenerator.cpp](#)

7.30 multiscale::video::RectangularCsvToInputFilesConverter Class Reference

Csv file to input file converter considering cartesian coordinates.

```
#include <RectangularCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::video::RectangularCsvToInputFilesConverter:



Public Member Functions

- `RectangularCsvToInputFilesConverter (const string &, const string &, unsigned int , unsigned int , unsigned int , unsigned int , bool , NumberIteratorType)`
- `~RectangularCsvToInputFilesConverter ()`
- `void convert ()`

Start the conversion.

Private Member Functions

- void `initInputFile` (ifstream &fin)
Initialise the input file stream over the given input file.
- void `initMaximumConcentration` (ifstream &fin)
Compute the value of member maximum concentration.
- void `initOutputFile` (ofstream &fout, unsigned int index, double &simulationTime)
Initialise the output file with the given index and simulation time.
- void `initIterators` (const NumberIteratorType &numberIteratorType)
Initialise the iterators considering the given number iterator type.
- void `validateSelectedConcentrationIndex` ()
Validate the selected concentration index in case of more than one concentration for each position.
- void `validateInput` (ifstream &fin)
Validate the input.
- void `validateInputLine` (const string &line, unsigned int lineNumber)
Validate the provided line identified by a line number.
- void `processInputFile` (ifstream &fin)
Process the input file.
- void `processLine` (const string &line, unsigned int outputIndex)
Process the provided line.
- vector< double > `splitLineInConcentrations` (const string &line, double &simulationTime)
Split the line in concentrations.
- void `splitLineInConcentrations` (vector< double > &concentrations, vector< string > &tokens, unsigned int rowIndex)
Split line into concentrations.
- double `computeSimulationTime` (const string &token)
Compute the simulation time from the given token and check if it is valid.
- double `computeNextPositionConcentration` (int concentrationIndex, vector< string > &tokens)
Compute the concentration for the next position.
- double `computeConcentration` (const string &concentration)
Compute the concentration from the given string.
- double `computeNonScaledConcentration` (const string &concentration)
Compute the non-scaled concentration from the given string.
- double `computeScaledConcentration` (const string &concentration)
Compute the scaled concentration from the given string.
- double `computeNormalisedConcentration` (double concentration)
Normalise the given concentration by dividing it to the maximum concentration.
- void `updateMaximumConcentration` (const string &line, double &maximumConcentration)
Update the maximum concentration if the values from the given line are greater than it.

Private Attributes

- string `inputfilepath`
- string `outputfilepath`
- unsigned int `height`
- unsigned int `width`
- unsigned int `nrOfConcentrationsForPosition`
- unsigned int `concentrationsIndex`
- unsigned int `selectedConcentrationIndex`
- bool `useLogScaling`
- double `maximumConcentration`
- NumberIterator * `rowsiterator`
- NumberIterator * `columnsiterator`

7.30.1 Detailed Description

Csv file to input file converter considering cartesian coordinates.

Definition at line 31 of file RectangularCsvToInputFilesConverter.hpp.

7.30.2 Constructor & Destructor Documentation

7.30.2.1 `RectangularCsvToInputFilesConverter::RectangularCsvToInputFilesConverter (const string & inputFilepath, const string & outputFilepath, unsigned int height, unsigned int width, unsigned int nrOfConcentrationsForPosition, unsigned int selectedConcentrationIndex, bool useLogScaling, NumberIteratorType numberIteratorType)`

Definition at line 19 of file RectangularCsvToInputFilesConverter.cpp.

7.30.2.2 `RectangularCsvToInputFilesConverter::~RectangularCsvToInputFilesConverter ()`

Definition at line 43 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3 Member Function Documentation

7.30.3.1 `double RectangularCsvToInputFilesConverter::computeConcentration (const string & concentration) [inline], [private]`

Compute the concentration from the given string.

Parameters

<code>concentration</code>	String representing the concentration
----------------------------	---------------------------------------

Definition at line 282 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.2 `double RectangularCsvToInputFilesConverter::computeNextPositionConcentration (int concentrationIndex, vector<string > & tokens) [inline], [private]`

Compute the concentration for the next position.

Parameters

<code>concentration-Index</code>	Index of the current concentration from the vector of tokens
<code>tokens</code>	Vector of tokens

Definition at line 255 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.3 `double RectangularCsvToInputFilesConverter::computeNonScaledConcentration (const string & concentration) [inline], [private]`

Compute the non-scaled concentration from the given string.

Parameters

<code>concentration</code>	String representing the concentration
----------------------------	---------------------------------------

Definition at line 288 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.4 double RectangularCsvToInputFilesConverter::computeNormalisedConcentration (double *concentration*)
 [inline], [private]

Normalise the given concentration by dividing it to the maximum concentration.

Parameters

<i>concentration</i>	The concentration
----------------------	-------------------

Definition at line 304 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.5 double RectangularCsvToInputFilesConverter::computeScaledConcentration (const string & *concentration*)
 [inline], [private]

Compute the scaled concentration from the given string.

Compute the scaled concentration from the given string by applying a logit transformation to it

Parameters

<i>concentration</i>	String representing the concentration
----------------------	---------------------------------------

Definition at line 292 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.6 double RectangularCsvToInputFilesConverter::computeSimulationTime (const string & *token*) [inline],
 [private]

Compute the simulation time from the given token and check if it is valid.

Parameters

<i>token</i>	Token (string)
--------------	----------------

Definition at line 245 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_NEG_SIM_TIME.

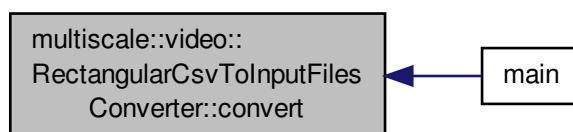
7.30.3.7 void RectangularCsvToInputFilesConverter::convert ()

Start the conversion.

Definition at line 48 of file RectangularCsvToInputFilesConverter.cpp.

Referenced by main().

Here is the caller graph for this function:



7.30.3.8 void RectangularCsvToInputFilesConverter::initInputFile (*ifstream & fin*) [private]

Initialise the input file stream over the given input file.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 61 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.30.3.9 void RectangularCsvToInputFilesConverter::initIterators (*const NumberIteratorType & numberIteratorType*) [private]

Initialise the iterators considering the given number iterator type.

Parameters

<i>numberIteratorType</i>	The type of the number iterator
---------------------------	---------------------------------

Definition at line 110 of file RectangularCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

7.30.3.10 void RectangularCsvToInputFilesConverter::initMaximumConcentration (*ifstream & fin*) [private]

Compute the value of member maximum concentration.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 69 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.30.3.11 void RectangularCsvToInputFilesConverter::initOutputFile (*ofstream & fout*, *unsigned int index*, *double & simulationTime*) [private]

Initialise the output file with the given index and simulation time.

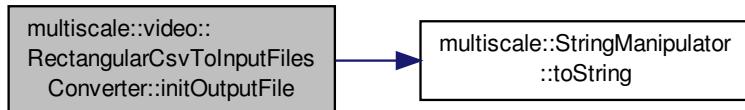
Parameters

<i>fout</i>	Output file stream
<i>index</i>	Index of the output file
<i>simulationTime</i>	Simulation time

Definition at line 93 of file RectangularCsvToInputFilesConverter.cpp.

References OUTPUT_EXTENSION, OUTPUT_FILE_SEPARATOR, OUTPUT_SEPARATOR, and multiscale::StringManipulator::toString().

Here is the call graph for this function:



7.30.3.12 void RectangularCsvToInputFilesConverter::process inputFile (ifstream & fin) [private]

Process the input file.

Read the concentrations and normalise them if it is the case.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 177 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.13 void RectangularCsvToInputFilesConverter::processLine (const string & line, unsigned int outputIndex) [private]

Process the provided line.

Parameters

<i>line</i>	Line
<i>outputIndex</i>	Index integrated in the name of the output file

Definition at line 192 of file RectangularCsvToInputFilesConverter.cpp.

References OUTPUT_SEPARATOR.

7.30.3.14 vector< double > RectangularCsvToInputFilesConverter::splitLineInConcentrations (const string & line, double & simulationTime) [private]

Split the line in concentrations.

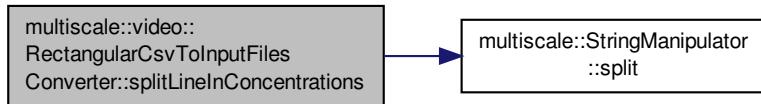
Parameters

<i>line</i>	Line
<i>simulationTime</i>	Simulation time associated with the line

Definition at line 209 of file RectangularCsvToInputFilesConverter.cpp.

References INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.30.3.15 void RectangularCsvToInputFilesConverter::splitLineInConcentrations (vector< double > & concentrations, vector< string > & tokens, unsigned int rowIndex) [private]

Split line into concentrations.

Parameters

<i>concentrations</i>	Concentrations extracted from tokens
<i>tokens</i>	Tokens representing the line
<i>rowIndex</i>	Index of the current row

Definition at line 230 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.16 void RectangularCsvToInputFilesConverter::updateMaximumConcentration (const string & line, double & maximumConcentration) [private]

Update the maximum concentration if the values from the given line are greater than it.

Parameters

<i>line</i>	Line from input file
<i>maximum-Concentration</i>	The maximum concentration

Definition at line 308 of file RectangularCsvToInputFilesConverter.cpp.

7.30.3.17 void RectangularCsvToInputFilesConverter::validateInput (ifstream & fin) [private]

Validate the input.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 133 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.30.3.18 void RectangularCsvToInputFilesConverter::validateInputLine (const string & line, unsigned int lineNumber) [private]

Validate the provided line identified by a line number.

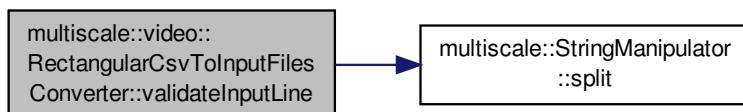
Parameters

<i>line</i>	Line from input file
<i>lineNumber</i>	Number of the line

Definition at line 157 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_INVALID_VALUE_LINE, ERR_INVALID_VALUE_TOKEN, ERR_NR_CONCENTRATIONS, INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.30.3.19 void RectangularCsvToInputFilesConverter::validateSelectedConcentrationIndex() [private]

Validate the selected concentration index in case of more than one concentration for each position.

Definition at line 127 of file RectangularCsvToInputFilesConverter.cpp.

References ERR_SELECTED_CONCENTRATION_INDEX.

7.30.4 Member Data Documentation

7.30.4.1 NumberIterator* multiscale::video::RectangularCsvToInputFilesConverter::columnsIterator [private]

Iterator over the number of columns

Definition at line 56 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.2 unsigned int multiscale::video::RectangularCsvToInputFilesConverter::concentrationsIndex [private]

Index of the current concentration

Definition at line 42 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.3 unsigned int multiscale::video::RectangularCsvToInputFilesConverter::height [private]

Height of the grid

Definition at line 38 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.4 string multiscale::video::RectangularCsvToInputFilesConverter::inputFilepath [private]

Path to the input file

Definition at line 35 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.5 double multiscale::video::RectangularCsvToInputFilesConverter::maximumConcentration [private]

The maximum concentration in the input file

Definition at line 53 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.6 unsigned int multiscale::video::RectangularCsvToInputFilesConverter::nrOfConcentrationsForPosition [private]

Number of concentrations for each position

Definition at line 40 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.7 string multiscale::video::RectangularCsvToInputFilesConverter::outputFilepath [private]

Path to the output file

Definition at line 36 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.8 NumberIterator* multiscale::video::RectangularCsvToInputFilesConverter::rowsIterator [private]

Iterator over the number of rows

Definition at line 55 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.9 unsigned int multiscale::video::RectangularCsvToInputFilesConverter::selectedConcentrationIndex [private]

Index of the concentration A in case the number of

concentrations for each position is greater than 1

finalConcentration = A / (A1 + A2 + ... + AN), where N is the number of concentrations for each position

Definition at line 44 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.10 bool multiscale::video::RectangularCsvToInputFilesConverter::useLogScaling [private]

Flag for using logarithmic scaling for concentrations or not

Definition at line 51 of file RectangularCsvToInputFilesConverter.hpp.

7.30.4.11 unsigned int multiscale::video::RectangularCsvToInputFilesConverter::width [private]

Width of the grid

Definition at line 39 of file RectangularCsvToInputFilesConverter.hpp.

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

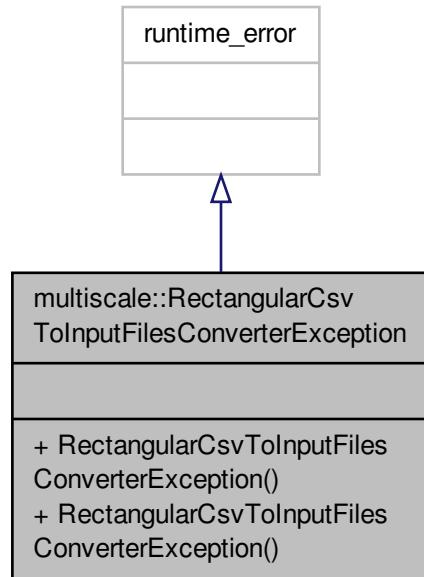
- include/multiscale/video/rectangular/[RectangularCsvToInputFilesConverter.hpp](#)
- src/multiscale/video/rectangular/[RectangularCsvToInputFilesConverter.cpp](#)

7.31 multiscale::RectangularCsvToInputFilesConverterException Class Reference

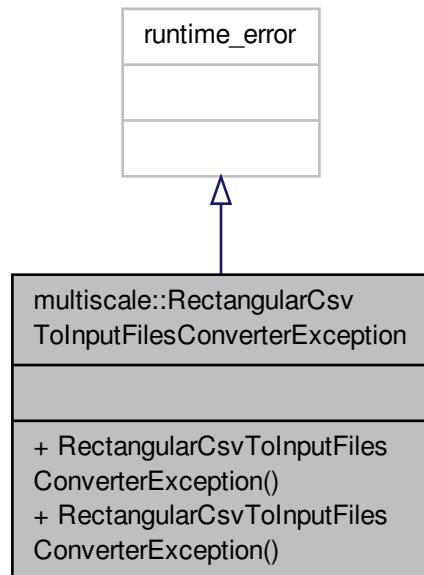
Exception class for the RectangularCsvToInputFilesConverter class.

```
#include <RectangularCsvToInputFilesConverterException.hpp>
```

Inheritance diagram for multiscale::RectangularCsvToInputFilesConverterException:



Collaboration diagram for multiscale::RectangularCsvToInputFilesConverterException:



Public Member Functions

- [RectangularCsvToInputFilesConverterException \(const string &msg\)](#)
- [RectangularCsvToInputFilesConverterException \(const char *msg\)](#)

7.31.1 Detailed Description

Exception class for the RectangularCsvToInputFilesConverter class.

Definition at line 13 of file RectangularCsvToInputFilesConverterException.hpp.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 multiscale::RectangularCsvToInputFilesConverterException::RectangularCsvToInputFilesConverterException (const string & msg) [inline]

Definition at line 17 of file RectangularCsvToInputFilesConverterException.hpp.

7.31.2.2 multiscale::RectangularCsvToInputFilesConverterException::RectangularCsvToInputFilesConverterException (const char * msg) [inline]

Definition at line 18 of file RectangularCsvToInputFilesConverterException.hpp.

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

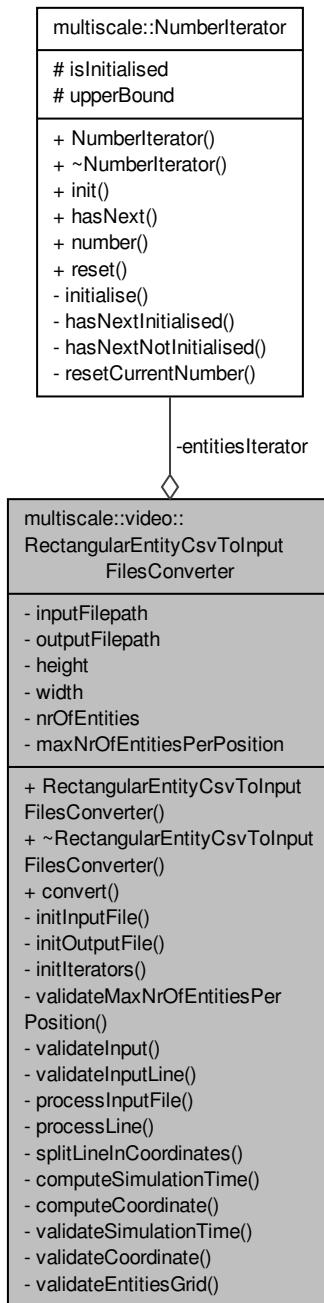
- [include/multiscale/exception/RectangularCsvToInputFilesConverterException.hpp](#)

7.32 multiscale::video::RectangularEntityCsvToInputFilesConverter Class Reference

Csv entity file to input file converter considering cartesian coordinates.

```
#include <RectangularEntityCsvToInputFilesConverter.hpp>
```

Collaboration diagram for multiscale::video::RectangularEntityCsvToInputFilesConverter:



Public Member Functions

- `RectangularEntityCsvToInputFilesConverter` (const string &`inputfilepath`, const string &`outputfilepath`, unsigned int `height`, unsigned int `width`, unsigned int `nrOfEntities`, unsigned int `maxNrOfEntitiesPerPosition`, `NumberIteratorType` `numberIteratorType`)
- `~RectangularEntityCsvToInputFilesConverter` ()
- `void convert` ()

Start the conversion.

Private Member Functions

- void `initInputFile` (ifstream &fin)
Initialise the input file stream over the given input file.
- void `initOutputFile` (ofstream &fout, unsigned int index, double &simulationTime)
Initialise the output file with the given index and simulation time.
- void `initIterators` (const NumberIteratorType &numberIteratorType)
Initialise the iterators considering the given number iterator type.
- void `validateMaxNrOfEntitiesPerPosition` ()
Check if the maximum number of entities per position is a non-zero natural number.
- void `validateInput` (ifstream &fin)
Validate the input.
- void `validateInputLine` (const string &line, unsigned int lineNumber)
Validate the provided line identified by a line number.
- void `processInputFile` (ifstream &fin)
Process the input file.
- void `processLine` (const string &line, unsigned int outputIndex)
Process the provided line.
- vector< double > `splitLineInCoordinates` (const string &line, double &simulationTime)
*Split the line in coordinates and return the grid of size height * width showing the position of the entities.*
- double `computeSimulationTime` (const string &token)
Compute the simulation time from the given token and check if it is valid.
- unsigned int `computeCoordinate` (const string &token, bool isOxCoordinate)
Compute the coordinate from the given string and check if it is valid.
- void `validateSimulationTime` (const string &token, unsigned int lineNumber)
Check if the simulation time is valid.
- void `validateCoordinate` (const string &token, unsigned int lineNumber, bool isOxCoordinate)
Check if the coordinate is valid.
- void `validateEntitiesGrid` (const vector< double > &entitiesGrid)
Check if the entities grid contains only values between zero and one.

Private Attributes

- string `inputFilepath`
- string `outputFilepath`
- unsigned int `height`
- unsigned int `width`
- unsigned int `nrOfEntities`
- unsigned int `maxNrOfEntitiesPerPosition`
- NumberIterator * `entitiesIterator`

7.32.1 Detailed Description

Csv entity file to input file converter considering cartesian coordinates.

Definition at line 33 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 `RectangularEntityCsvToInputFilesConverter::RectangularEntityCsvToInputFilesConverter (const string & inputFilepath, const string & outputFilepath, unsigned int height, unsigned int width, unsigned int nrOfEntities, unsigned int maxNrOfEntitiesPerPosition, NumberIteratorType numberIteratorType)`

Definition at line 19 of file RectangularEntityCsvToInputFilesConverter.cpp.

7.32.2.2 RectangularEntityCsvToInputFilesConverter::~RectangularEntityCsvToInputFilesConverter()

Definition at line 38 of file RectangularEntityCsvToInputFilesConverter.cpp.

7.32.3 Member Function Documentation

7.32.3.1 unsigned int RectangularEntityCsvToInputFilesConverter::computeCoordinate (const string & *token*, bool *isOxCoordinate*) [inline], [private]

Compute the coordinate from the given string and check if it is valid.

Parameters

<i>token</i>	Token (string)
<i>isOxCoordinate</i>	Flag which indicates if the coordinate corresponds to Ox axis or not

Definition at line 208 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INVALID_OX_COORDINATE, and ERR_INVALID_OY_COORDINATE.

7.32.3.2 double RectangularEntityCsvToInputFilesConverter::computeSimulationTime (const string & *token*) [inline], [private]

Compute the simulation time from the given token and check if it is valid.

Parameters

<i>token</i>	Token (string)
--------------	----------------

Definition at line 198 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_NEG_SIM_TIME.

7.32.3.3 void RectangularEntityCsvToInputFilesConverter::convert()

Start the conversion.

Definition at line 42 of file RectangularEntityCsvToInputFilesConverter.cpp.

Referenced by main().

Here is the caller graph for this function:



7.32.3.4 void RectangularEntityCsvToInputFilesConverter::initInputFile (ifstream & *fin*) [private]

Initialise the input file stream over the given input file.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 54 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.32.3.5 void RectangularEntityCsvToInputFilesConverter::initIterators (const NumberIteratorType & *numberIteratorType*) [private]

Initialise the iterators considering the given number iterator type.

Parameters

<i>numberIteratorType</i>	The type of the number iterator
---------------------------	---------------------------------

Definition at line 79 of file RectangularEntityCsvToInputFilesConverter.cpp.

References multiscale::LEXICOGRAPHIC, and multiscale::STANDARD.

7.32.3.6 void RectangularEntityCsvToInputFilesConverter::initOutputFile (ofstream & *fout*, unsigned int *index*, double & *simulationTime*) [private]

Initialise the output file with the given index and simulation time.

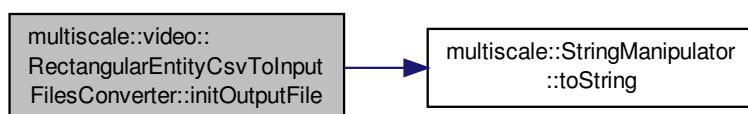
Parameters

<i>fout</i>	Output file stream
<i>index</i>	Index of the output file
<i>simulationTime</i>	Simulation time

Definition at line 62 of file RectangularEntityCsvToInputFilesConverter.cpp.

References OUTPUT_EXTENSION, OUTPUT_FILE_SEPARATOR, OUTPUT_SEPARATOR, and multiscale::StringManipulator::toString().

Here is the call graph for this function:



7.32.3.7 void RectangularEntityCsvToInputFilesConverter::processInputFile (ifstream & *fin*) [private]

Process the input file.

Read the concentrations and normalise them if it is the case.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 142 of file RectangularEntityCsvToInputFilesConverter.cpp.

7.32.3.8 void RectangularEntityCsvToInputFilesConverter::processLine (const string & *line*, unsigned int *outputIndex*) [private]

Process the provided line.

Parameters

<i>line</i>	Line
<i>outputIndex</i>	Index integrated in the name of the output file

Definition at line 157 of file RectangularEntityCsvToInputFilesConverter.cpp.

References OUTPUT_SEPARATOR.

7.32.3.9 vector< double > RectangularEntityCsvToInputFilesConverter::splitLineInCoordinates (const string & *line*, double & *simulationTime*) [private]

Split the line in coordinates and return the grid of size height * width showing the position of the entities.

The number of entities per grid position is normalised to the range [0, 1]

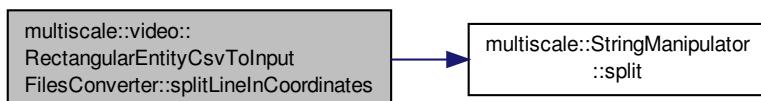
Parameters

<i>line</i>	Line
<i>simulationTime</i>	Simulation time associated with the line

Definition at line 176 of file RectangularEntityCsvToInputFilesConverter.cpp.

References INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.32.3.10 void RectangularEntityCsvToInputFilesConverter::validateCoordinate (const string & *token*, unsigned int *lineNumber*, bool *isOxCoordinate*) [inline], [private]

Check if the coordinate is valid.

Parameters

<i>token</i>	Token (string)
<i>lineNumber</i>	Number of the line
<i>isOxCoordinate</i>	Flag which indicates if the coordinate corresponds to Ox axis or not

Definition at line 237 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INVALID_VALUE_LINE, and ERR_INVALID_VALUE_TOKEN.

7.32.3.11 void RectangularEntityCsvToInputFilesConverter::validateEntitiesGrid (const vector< double > & entitiesGrid) [inline], [private]

Check if the entities grid contains only values between zero and one.

Parameters

<i>entitiesGrid</i>	The grid of entities
---------------------	----------------------

Definition at line 250 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INVALID_NR_ENTITIES.

7.32.3.12 void RectangularEntityCsvToInputFilesConverter::validateInput (ifstream & fin) [private]

Validate the input.

Parameters

<i>fin</i>	Input file stream
------------	-------------------

Definition at line 101 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INPUT_OPEN.

7.32.3.13 void RectangularEntityCsvToInputFilesConverter::validateInputLine (const string & line, unsigned int lineNumber) [private]

Validate the provided line identified by a line number.

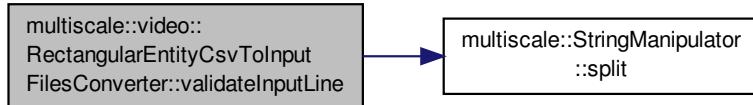
Parameters

<i>line</i>	Line from input file
<i>lineNumber</i>	Number of the line

Definition at line 125 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_NR_COORDINATES, INPUT_FILE_SEPARATOR, and multiscale::StringManipulator::split().

Here is the call graph for this function:



7.32.3.14 void RectangularEntityCsvToInputFilesConverter::validateMaxNrOfEntitiesPerPosition () [private]

Check if the maximum number of entities per position is a non-zero natural number.

Definition at line 95 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_MAX_NR_ENTITIES.

7.32.3.15 void RectangularEntityCsvToInputFilesConverter::validateSimulationTime (const string & token, unsigned int lineNumber) [inline], [private]

Check if the simulation time is valid.

Parameters

<i>token</i>	Token (string)
<i>lineNumber</i>	Number of the line

Definition at line 224 of file RectangularEntityCsvToInputFilesConverter.cpp.

References ERR_INVALID_VALUE_LINE, and ERR_INVALID_VALUE_TOKEN.

7.32.4 Member Data Documentation

7.32.4.1 NumberIterator* multiscale::video::RectangularEntityCsvToInputFilesConverter::entitiesIterator [private]

Iterator over the number of rows

Definition at line 46 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.2 unsigned int multiscale::video::RectangularEntityCsvToInputFilesConverter::height [private]

Height of the grid

Definition at line 40 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.3 string multiscale::video::RectangularEntityCsvToInputFilesConverter::inputFilepath [private]

Path to the input file

Definition at line 37 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.4 `unsigned int multiscale::video::RectangularEntityCsvToInputFilesConverter::maxNrOfEntitiesPerPosition` [private]

The maximum number of entities per position

Definition at line 44 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.5 `unsigned int multiscale::video::RectangularEntityCsvToInputFilesConverter::nrOfEntities` [private]

Number of entities

Definition at line 42 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.6 `string multiscale::video::RectangularEntityCsvToInputFilesConverter::outputFilepath` [private]

Path to the output file

Definition at line 38 of file RectangularEntityCsvToInputFilesConverter.hpp.

7.32.4.7 `unsigned int multiscale::video::RectangularEntityCsvToInputFilesConverter::width` [private]

Width of the grid

Definition at line 41 of file RectangularEntityCsvToInputFilesConverter.hpp.

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

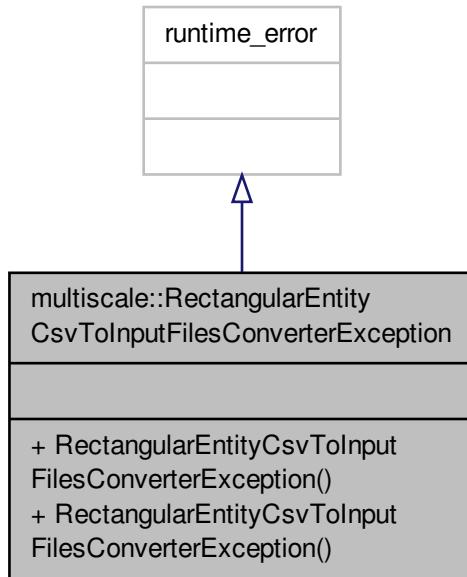
- include/multiscale/video/rectangular/[RectangularEntityCsvToInputFilesConverter.hpp](#)
- src/multiscale/video/rectangular/[RectangularEntityCsvToInputFilesConverter.cpp](#)

7.33 multiscale::RectangularEntityCsvToInputFilesConverterException Class Reference

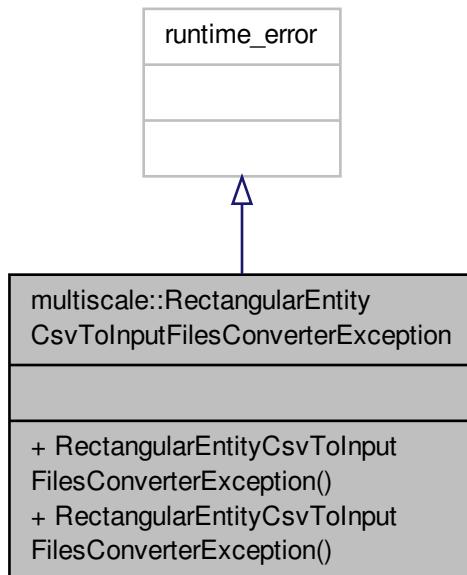
Exception class for the RectangularEntityCsvToInputFilesConverter class.

```
#include <RectangularEntityCsvToInputFilesConverterException.hpp>
```

Inheritance diagram for multiscale::RectangularEntityCsvToInputFilesConverterException:



Collaboration diagram for multiscale::RectangularEntityCsvToInputFilesConverterException:



Public Member Functions

- [RectangularEntityCsvToInputFilesConverterException](#) (const string &msg)
- [RectangularEntityCsvToInputFilesConverterException](#) (const char *msg)

7.33.1 Detailed Description

Exception class for the RectangularEntityCsvToInputFilesConverter class.

Definition at line 13 of file RectangularEntityCsvToInputFilesConverterException.hpp.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 multiscale::RectangularEntityCsvToInputFilesConverterException::RectangularEntityCsvToInputFilesConverterException (const string & msg) [inline]

Definition at line 17 of file RectangularEntityCsvToInputFilesConverterException.hpp.

7.33.2.2 multiscale::RectangularEntityCsvToInputFilesConverterException::RectangularEntityCsvToInputFilesConverterException (const char * msg) [inline]

Definition at line 18 of file RectangularEntityCsvToInputFilesConverterException.hpp.

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

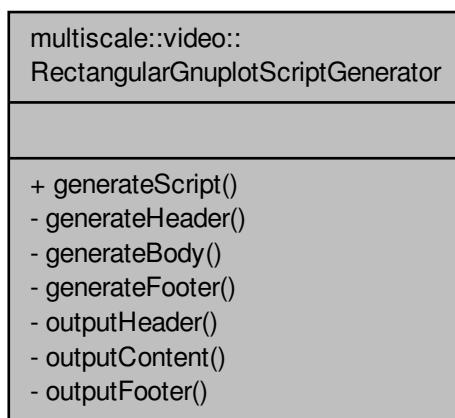
- [include/multiscale/exception/RectangularEntityCsvToInputFilesConverterException.hpp](#)

7.34 multiscale::video::RectangularGnuplotScriptGenerator Class Reference

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

```
#include <RectangularGnuplotScriptGenerator.hpp>
```

Collaboration diagram for multiscale::video::RectangularGnuplotScriptGenerator:



Static Public Member Functions

- static void `generateScript` (const vector< double > &concentrations, double simulationTime, unsigned long height, unsigned long width, const string &outputfilepath)

Generate the script.

Static Private Member Functions

- static void `generateHeader` (ofstream &fout, const string &outputfilepath, double simulationTime, unsigned long height, unsigned long width)

Generate the header of the script.

- static void `generateBody` (const vector< double > &concentrations, unsigned long height, unsigned long width, ofstream &fout)

Generate the body/content of the script.

- static void `generateFooter` (ofstream &fout)

Generate the footer of the script.

- static void `outputHeader` (ifstream &fin, const string &outputfilename, double simulationTime, unsigned long height, unsigned long width, ofstream &fout)

Output the header of the script.

- static void `outputContent` (const vector< double > &concentrations, unsigned long height, unsigned long width, ofstream &fout)

Output the content of the script.

- static void `outputFooter` (ifstream &fin, ofstream &fout)

Output the footer of the script.

7.34.1 Detailed Description

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

Definition at line 30 of file RectangularGnuplotScriptGenerator.hpp.

7.34.2 Member Function Documentation

- 7.34.2.1 void RectangularGnuplotScriptGenerator::generateBody (const vector< double > & *concentrations*, unsigned long *height*, unsigned long *width*, ofstream & *fout*) [static], [private]

Generate the body/content of the script.

Parameters

<i>concentrations</i>	The concentrations
<i>height</i>	The height of the grid
<i>width</i>	The width of the grid
<i>fout</i>	Output file stream

Definition at line 44 of file RectangularGnuplotScriptGenerator.cpp.

References CONTENT_IN.

- 7.34.2.2 void RectangularGnuplotScriptGenerator::generateFooter (ofstream & *fout*) [static], [private]

Generate the footer of the script.

Parameters

<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 55 of file RectangularGnuplotScriptGenerator.cpp.

References FOOTER_IN.

7.34.2.3 void RectangularGnuplotScriptGenerator::generateHeader (ostream & *fout*, const string & *outputfilepath*, double *simulationTime*, unsigned long *height*, unsigned long *width*) [static], [private]

Generate the header of the script.

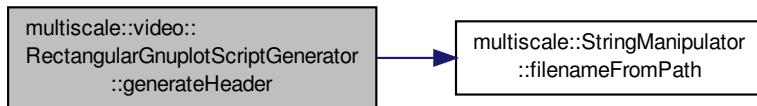
Parameters

<i>fout</i>	Output file stream
<i>outputfilepath</i>	Path to the output file
<i>simulationTime</i>	Simulation time
<i>height</i>	Height of the grid
<i>width</i>	Width of the grid

Definition at line 30 of file RectangularGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::filenameFromPath(), and HEADER_IN.

Here is the call graph for this function:



7.34.2.4 void RectangularGnuplotScriptGenerator::generateScript (const vector< double > & *concentrations*, double *simulationTime*, unsigned long *height*, unsigned long *width*, const string & *outputfilepath*) [static]

Generate the script.

Parameters

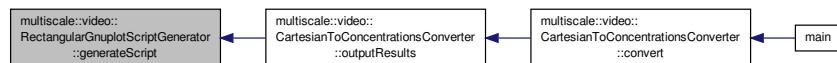
<i>concentrations</i>	Concentrations
<i>simulationTime</i>	Simulation time
<i>height</i>	Height of the grid
<i>width</i>	Width of the grid
<i>outputfilepath</i>	Path of the output file

Definition at line 14 of file RectangularGnuplotScriptGenerator.cpp.

References GNUPLOT_EXTENSION.

Referenced by multiscale::video::CartesianToConcentrationsConverter::outputResults().

Here is the caller graph for this function:



7.34.2.5 void RectangularGnuplotScriptGenerator::outputContent (const vector< double > & concentrations, unsigned long height, unsigned long width, ofstream & fout) [static], [private]

Output the content of the script.

Parameters

<i>concentrations</i>	The concentrations
<i>height</i>	The height of the grid
<i>width</i>	The width of the grid
<i>fout</i>	Output file stream

Definition at line 81 of file RectangularGnuplotScriptGenerator.cpp.

References OUTPUT_SEPARATOR.

7.34.2.6 void RectangularGnuplotScriptGenerator::outputFooter (ifstream & fin, ofstream & fout) [static], [private]

Output the footer of the script.

Parameters

<i>fin</i>	Input file stream
<i>fout</i>	Output file stream

Definition at line 94 of file RectangularGnuplotScriptGenerator.cpp.

7.34.2.7 void RectangularGnuplotScriptGenerator::outputHeader (ifstream & fin, const string & outputFilename, double simulationTime, unsigned long height, unsigned long width, ofstream & fout) [static], [private]

Output the header of the script.

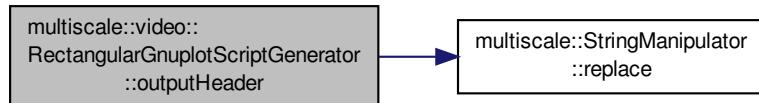
Parameters

<i>fin</i>	Input file stream
<i>outputFilename</i>	Name of the output file
<i>simulationTime</i>	Simulation time
<i>height</i>	The height of the grid
<i>width</i>	The width of the grid
<i>fout</i>	Output file stream

Definition at line 65 of file RectangularGnuplotScriptGenerator.cpp.

References multiscale::StringManipulator::replace(), REPLACE_DIMENSION_EXTRA, REPLACE_HEADER_FILENAME, REPLACE_HEADER_HEIGHT, REPLACE_HEADER_SIM_TIME, and REPLACE_HEADER_WIDTH.

Here is the call graph for this function:



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

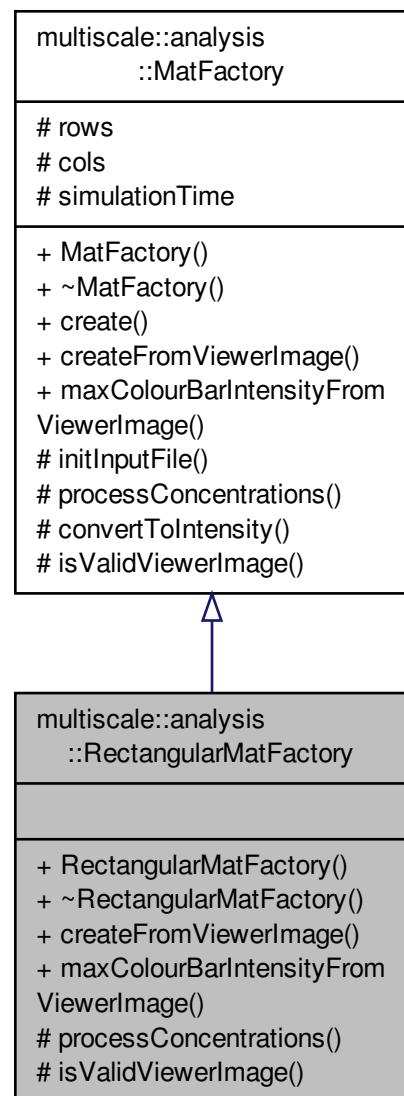
- include/multiscale/video/rectangular/[RectangularGnuplotScriptGenerator.hpp](#)
- src/multiscale/video/rectangular/[RectangularGnuplotScriptGenerator.cpp](#)

7.35 multiscale::analysis::RectangularMatFactory Class Reference

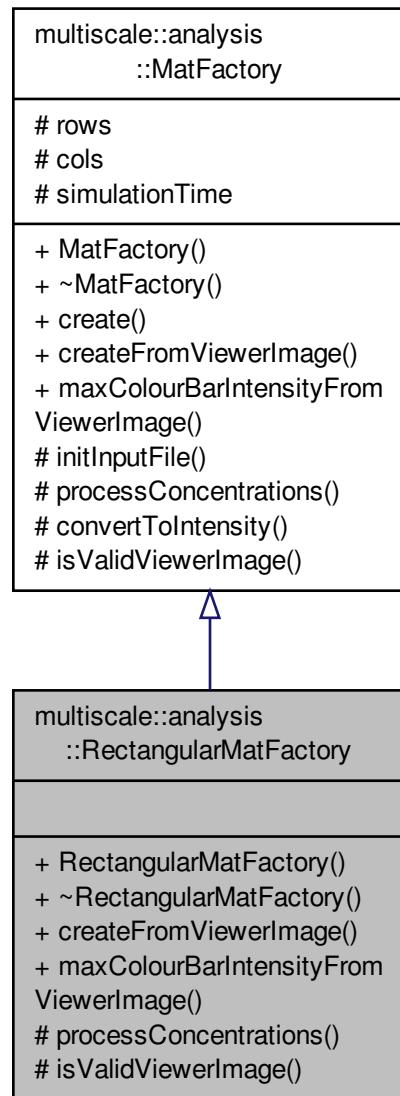
Class for creating a Mat object considering a rectangular grid.

```
#include <RectangularMatFactory.hpp>
```

Inheritance diagram for multiscale::analysis::RectangularMatFactory:



Collaboration diagram for multiscale::analysis::RectangularMatFactory:



Public Member Functions

- `RectangularMatFactory ()`
- `~RectangularMatFactory ()`
- Mat `createFromViewerImage (const string &inputFile)` override
Create a Mat object from the image file obtained from the RectangularGeometryViewer.
- double `maxColourBarIntensityFromViewerImage (const string &inputFile)` override
Get the maximum grayscale intensity of the colour bar in the image.

Protected Member Functions

- `unsigned char * processConcentrations (ifstream &fin)` override

Process the concentrations from the input file.

- bool `isValidViewerImage` (const Mat &image) override

Check if the image generated by the viewer has the required resolution.

Additional Inherited Members

7.35.1 Detailed Description

Class for creating a Mat object considering a rectangular grid.

Definition at line 27 of file RectangularMatFactory.hpp.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 RectangularMatFactory::RectangularMatFactory()

Definition at line 9 of file RectangularMatFactory.cpp.

7.35.2.2 RectangularMatFactory::~RectangularMatFactory()

Definition at line 11 of file RectangularMatFactory.cpp.

7.35.3 Member Function Documentation

7.35.3.1 Mat RectangularMatFactory::createFromViewerImage(const string & *inputFile*) [override], [virtual]

Create a Mat object from the image file obtained from the RectangularGeometryViewer.

Create the Mat instance from the given image file

Parameters

<code>inputFile</code>	The path to the image file
------------------------	----------------------------

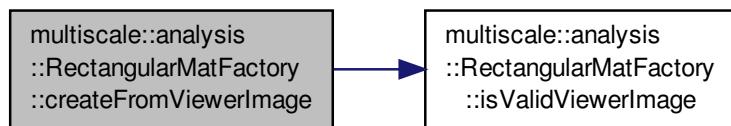
Implements [multiscale::analysis::MatFactory](#).

Definition at line 13 of file RectangularMatFactory.cpp.

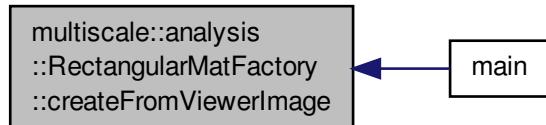
References `isValidViewerImage()`, ROI_HEIGHT, ROI_START_X, ROI_START_Y, and ROI_WIDTH.

Referenced by `main()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.35.3.2 bool RectangularMatFactory::isValidViewerImage (const Mat & *image*) [override], [protected], [virtual]

Check if the image generated by the viewer has the required resolution.

Parameters

<i>image</i>	Image generated by the viewer
--------------	-------------------------------

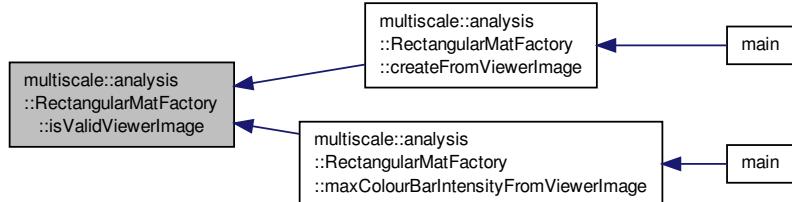
Implements [multiscale::analysis::MatFactory](#).

Definition at line 46 of file `RectangularMatFactory.cpp`.

References `ERR_IMG_RESOLUTION`, `ERR_INPUT_OPEN`, `INPUT_IMG_HEIGHT`, and `INPUT_IMG_WIDTH`.

Referenced by `createFromViewerImage()`, and `maxColourBarIntensityFromViewerImage()`.

Here is the caller graph for this function:



7.35.3.3 double RectangularMatFactory::maxColourBarIntensityFromViewerImage (const string & *inputFile*) [override], [virtual]

Get the maximum grayscale intensity of the colour bar in the image.

Parameters

<i>inputFile</i>	The path to the image file
------------------	----------------------------

Implements [multiscale::analysis::MatFactory](#).

Definition at line 21 of file RectangularMatFactory.cpp.

References COLOURBAR_MAX_X, COLOURBAR_MAX_Y, and isValidViewerImage().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.35.3.4 `unsigned char * RectangularMatFactory::processConcentrations (ifstream & fin) [override], [protected], [virtual]`

Process the concentrations from the input file.

Read the concentrations from the input file and return them as an array which can be used afterwards to create a Mat object from them

REMARK: The constructor of Mat does not copy the data. Therefore, DO NOT deallocate it in this class.

Parameters

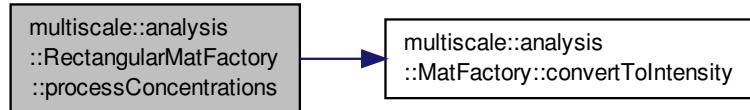
<code>fin</code>	Input file stream from which the concentrations are read
------------------	--

Implements [multiscale::analysis::MatFactory](#).

Definition at line 29 of file RectangularMatFactory.cpp.

References multiscale::analysis::MatFactory::cols, multiscale::analysis::MatFactory::convertToIntensity(), ERR_C-ONC, and multiscale::analysis::MatFactory::rows.

Here is the call graph for this function:



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

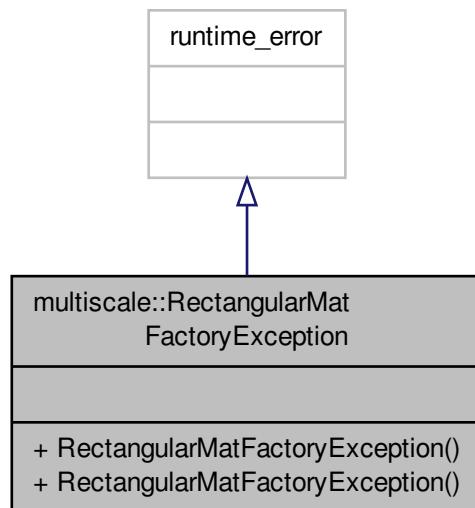
- include/multiscale/analysis/spatial/factory/[RectangularMatFactory.hpp](#)
- src/multiscale/analysis/spatial/factory/[RectangularMatFactory.cpp](#)

7.36 multiscale::RectangularMatFactoryException Class Reference

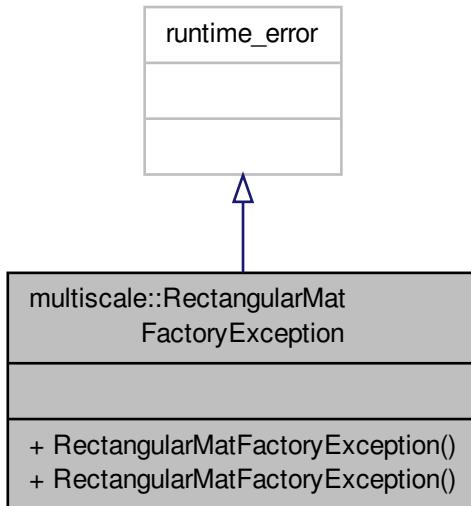
Exception class for the RectangularMatFactory instances.

```
#include <RectangularMatFactoryException.hpp>
```

Inheritance diagram for multiscale::RectangularMatFactoryException:



Collaboration diagram for multiscale::RectangularMatFactoryException:



Public Member Functions

- [RectangularMatFactoryException \(const string &msg\)](#)
- [RectangularMatFactoryException \(const char *msg\)](#)

7.36.1 Detailed Description

Exception class for the RectangularMatFactory instances.

Definition at line 13 of file RectangularMatFactoryException.hpp.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 [multiscale::RectangularMatFactoryException::RectangularMatFactoryException \(const string & msg \) \[inline\]](#)

Definition at line 17 of file RectangularMatFactoryException.hpp.

7.36.2.2 [multiscale::RectangularMatFactoryException::RectangularMatFactoryException \(const char * msg \) \[inline\]](#)

Definition at line 18 of file RectangularMatFactoryException.hpp.

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

- [include/multiscale/exception/RectangularMatFactoryException.hpp](#)

7.37 multiscale::analysis::Region Class Reference

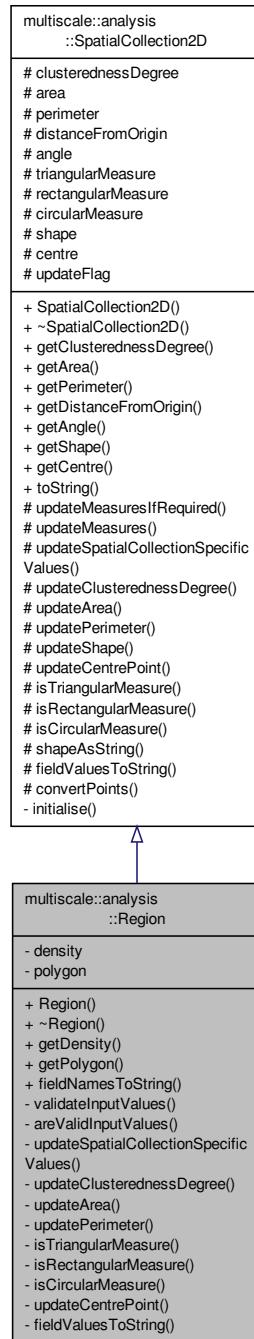
Class for representing a region.

```
#include <Region.hpp>
```

Inheritance diagram for multiscale::analysis::Region:



Collaboration diagram for multiscale::analysis::Region:



Public Member Functions

- `Region (double clusterednessDegree, double density, double area, double distanceFromOrigin, double angle-WrtOrigin, const vector< Point > &polygon)`
- `~Region ()`
- `double getDensity ()`

Get the density.

- const vector< Point > & `getPolygon ()`

Get the polygon defining the region.

Static Public Member Functions

- static string `fieldNamesToString ()`

Get a string representation of all the field names except polygon.

Private Member Functions

- void `validateInputValues` (double `clusterednessDegree`, double `density`, double `area`, double `distanceFromOrigin`, double `angleWrtOrigin`, const vector< Point > &`polygon`)

Validate the input values.

- bool `isValidInputValues` (double `clusterednessDegree`, double `density`, double `area`, double `distanceFromOrigin`, double `angleWrtOrigin`, const vector< Point > &`polygon`)

Check if the input values are valid or not.

- void `updateSpatialCollectionSpecificValues ()` override

Update the value of all class specific measures.

- void `updateClusterednessDegree ()` override

Update the value of the clusteredness degree.

- void `updateArea ()` override

Update the area.

- void `updatePerimeter ()` override

Update the perimeter.

- double `isTriangularMeasure ()` override

Get the measure that the cluster has a triangular shape.

- double `isRectangularMeasure ()` override

Get the measure that the cluster has a rectangular shape.

- double `isCircularMeasure ()` override

Get the measure that the cluster has a circular shape.

- void `updateCentrePoint ()` override

Update the centre of the region.

- string `fieldValuesToString ()` override

Get a string representation of all the field values except polygon.

Private Attributes

- double `density`
- vector< Point > `polygon`

Additional Inherited Members

7.37.1 Detailed Description

Class for representing a region.

Definition at line 22 of file Region.hpp.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 Region::Region (double *clusterednessDegree*, double *density*, double *area*, double *distanceFromOrigin*, double *angleWrtOrigin*, const vector< Point > & *polygon*)

Definition at line 11 of file Region.cpp.

References multiscale::analysis::SpatialCollection2D::angle, multiscale::analysis::SpatialCollection2D::area, multiscale::analysis::SpatialCollection2D::clusterednessDegree, density, multiscale::analysis::SpatialCollection2D::distanceFromOrigin, polygon, and validateInputValues().

Here is the call graph for this function:



7.37.2.2 Region::~Region ()

Definition at line 24 of file Region.cpp.

References polygon.

7.37.3 Member Function Documentation

7.37.3.1 bool Region::areValidInputValues (double *clusterednessDegree*, double *density*, double *area*, double *distanceFromOrigin*, double *angleWrtOrigin*, const vector< Point > & *polygon*) [private]

Check if the input values are valid or not.

Validation rules: $0 < \text{clusterednessDegree}$ $0 < \text{density}$ $0 < \text{distanceFromOrigin}$ $0 \leq \text{angleWrtOrigin} \leq 360$

For each polygon point p: $0 \leq p.x \leq p.y$

Parameters

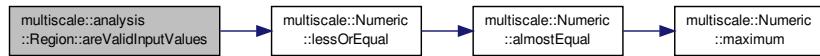
<i>clusterednessDegree</i>	The clusteredness degree of the region
<i>density</i>	The density of the region
<i>area</i>	The area of the region considering holes
<i>distanceFromOrigin</i>	The distance from the origin
<i>angleWrtOrigin</i>	The angle computed wrt to the origin
<i>polygon</i>	The polygon

Definition at line 49 of file Region.cpp.

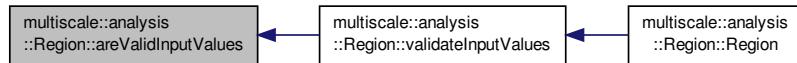
References multiscale::Numeric::lessOrEqual().

Referenced by validateInputValues().

Here is the call graph for this function:



Here is the caller graph for this function:



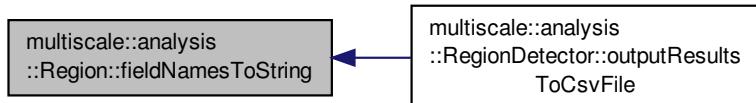
7.37.3.2 string Region::fieldNamesToString() [static]

Get a string representation of all the field names except polygon.

Definition at line 38 of file Region.cpp.

Referenced by multiscale::analysis::RegionDetector::outputResultsToCsvFile().

Here is the caller graph for this function:



7.37.3.3 string Region::fieldValuesToString() [override], [private], [virtual]

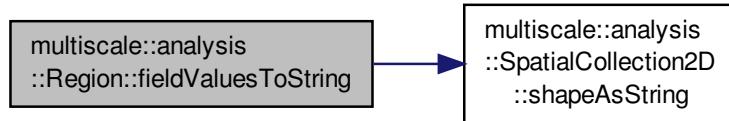
Get a string representation of all the field values except polygon.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 118 of file Region.cpp.

References multiscale::analysis::SpatialCollection2D::angle, multiscale::analysis::SpatialCollection2D::area, multiscale::analysis::SpatialCollection2D::centre, multiscale::analysis::SpatialCollection2D::circularMeasure, multiscale::analysis::SpatialCollection2D::clusterednessDegree, density, multiscale::analysis::SpatialCollection2D::distanceFromOrigin, OUTPUT_SEPARATOR, multiscale::analysis::SpatialCollection2D::perimeter, multiscale::analysis::SpatialCollection2D::rectangularMeasure, multiscale::analysis::SpatialCollection2D::shapeAsString(), and multiscale::analysis::SpatialCollection2D::triangularMeasure.

Here is the call graph for this function:



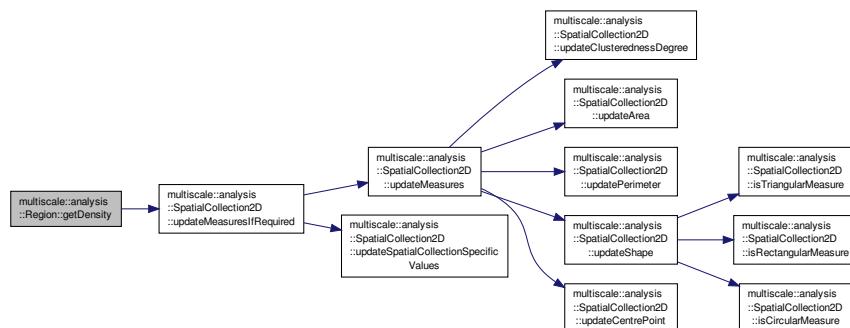
7.37.3.4 double Region::getDensity ()

Get the density.

Definition at line 28 of file Region.cpp.

References density, and multiscale::analysis::SpatialCollection2D::updateMeasuresIfRequired().

Here is the call graph for this function:



7.37.3.5 const vector< Point > & Region::getPolygon ()

Get the polygon defining the region.

Definition at line 34 of file Region.cpp.

References polygon.

7.37.3.6 double Region::isCircularMeasure () [override], [private], [virtual]

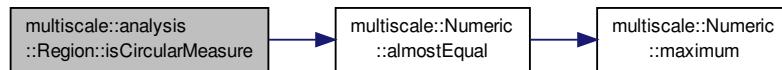
Get the measure that the cluster has a circular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 99 of file Region.cpp.

References multiscale::Numeric::almostEqual(), multiscale::analysis::SpatialCollection2D::area, PI, and polygon.

Here is the call graph for this function:



7.37.3.7 double Region::isRectangularMeasure() [override], [private], [virtual]

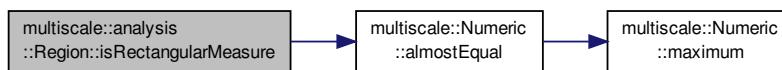
Get the measure that the cluster has a rectangular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 89 of file Region.cpp.

References [multiscale::Numeric::almostEqual\(\)](#), [multiscale::analysis::SpatialCollection2D::area](#), and [polygon](#).

Here is the call graph for this function:



7.37.3.8 double Region::isTriangularMeasure() [override], [private], [virtual]

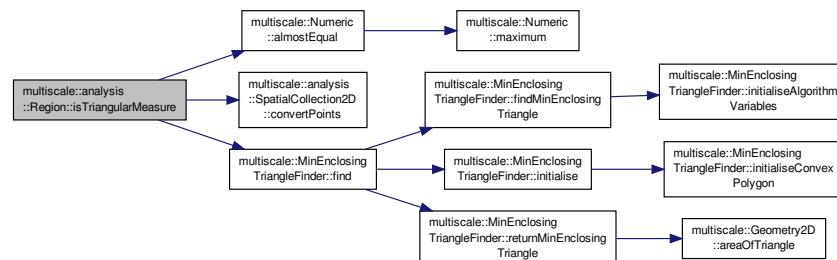
Get the measure that the cluster has a triangular shape.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 77 of file Region.cpp.

References [multiscale::Numeric::almostEqual\(\)](#), [multiscale::analysis::SpatialCollection2D::area](#), [multiscale::analysis::SpatialCollection2D::convertPoints\(\)](#), [CONVEX_HULL_CLOCKWISE](#), [multiscale::MinEnclosingTriangleFinder::find\(\)](#), and [polygon](#).

Here is the call graph for this function:



7.37.3.9 void Region::updateArea() [override], [private], [virtual]

Update the area.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 71 of file Region.cpp.

7.37.3.10 void Region::updateCentrePoint() [override], [private], [virtual]

Update the centre of the region.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 112 of file Region.cpp.

References multiscale::analysis::SpatialCollection2D::centre, and polygon.

7.37.3.11 void Region::updateClusterednessDegree() [override], [private], [virtual]

Update the value of the clusteredness degree.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 69 of file Region.cpp.

7.37.3.12 void Region::updatePerimeter() [override], [private], [virtual]

Update the perimeter.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 73 of file Region.cpp.

References CONTOUR_CLOSED, multiscale::analysis::SpatialCollection2D::perimeter, and polygon.

7.37.3.13 void Region::updateSpatialCollectionSpecificValues() [override], [private], [virtual]

Update the value of all class specific measures.

Implements [multiscale::analysis::SpatialCollection2D](#).

Definition at line 67 of file Region.cpp.

7.37.3.14 void Region::validateInputValues(double clusterednessDegree, double density, double area, double distanceFromOrigin, double angleWrtOrigin, const vector< Point > & polygon) [private]

Validate the input values.

Validation rules: $0 < \text{clusterednessDegree}$ $0 < \text{density}$ $0 < \text{distanceFromOrigin}$ $0 \leq \text{angleWrtOrigin} \leq 360$

For each polygon point p: $0 \leq p.x \leq p.y$

Parameters

<i>clusterednessDegree</i>	The clusteredness degree of the region
<i>density</i>	The density of the region
<i>area</i>	The area of the region considering holes
<i>distanceFromOrigin</i>	The distance from the origin
<i>angleWrtOrigin</i>	The angle computed wrt to the origin
<i>polygon</i>	The polygon

Definition at line 42 of file Region.cpp.

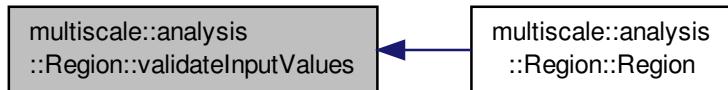
References areValidInputValues(), and ERR_INPUT.

Referenced by Region().

Here is the call graph for this function:



Here is the caller graph for this function:



7.37.4 Member Data Documentation

7.37.4.1 double multiscale::analysis::Region::density [private]

The average intensity of the pixels in the region

normalised to the interval [0, 1]

Definition at line 26 of file Region.hpp.

Referenced by fieldValuesToString(), getDensity(), and Region().

7.37.4.2 vector<Point> multiscale::analysis::Region::polygon [private]

Polygon defining the region

Definition at line 29 of file Region.hpp.

Referenced by getPolygon(), isCircularMeasure(), isRectangularMeasure(), isTriangularMeasure(), Region(), updateCentrePoint(), updatePerimeter(), and ~Region().

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

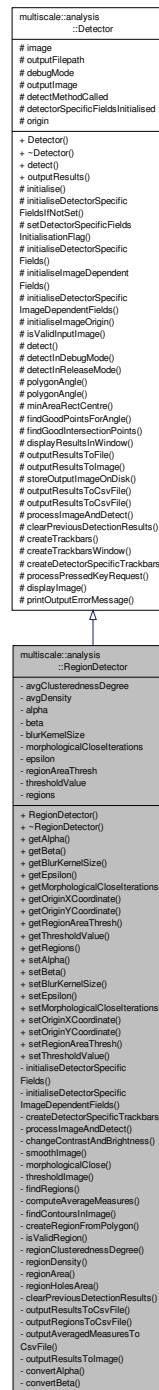
- include/multiscale/analysis/spatial/Region.hpp
- src/multiscale/analysis/spatial/Region.cpp

7.38 multiscale::analysis::RegionDetector Class Reference

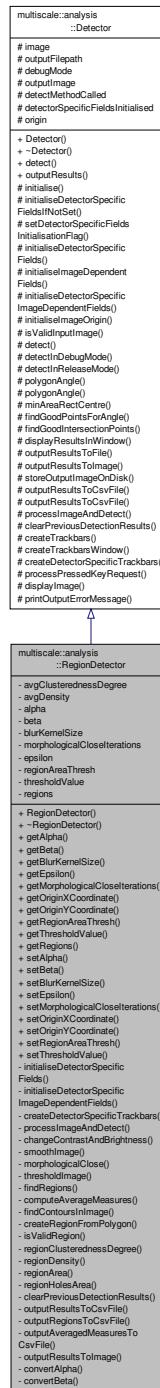
Class for detecting regions of high intensity in grayscale images.

```
#include <RegionDetector.hpp>
```

Inheritance diagram for multiscale::analysis::RegionDetector:



Collaboration diagram for multiscale::analysis::RegionDetector:



Public Member Functions

- `RegionDetector (bool debugMode=false)`
- `~RegionDetector ()`
- int `getAlpha ()`

Get the value of field alpha.

- int `getBeta ()`

- `Get the value of field beta.`
• int `getBlurKernelSize ()`
Get the value of field blurKernelSize.
- int `getEpsilon ()`
Get the value of field epsilon.
- int `getMorphologicalCloselterations ()`
Get the value of field morphologicalCloselterations.
- int `getOriginXCoordinate ()`
Get the value of field originXCoordinate.
- int `getOriginYCoordinate ()`
Get the value of field originYCoordinate.
- int `getRegionAreaThresh ()`
Get the value of field regionAreaThresh.
- int `getThresholdValue ()`
Get the value of field thresholdValue.
- vector< `Region` > const & `getRegions ()`
Get a const reference to the vector of detected regions.
- void `setAlpha (int alpha)`
Set the value of field alpha.
- void `setBeta (int beta)`
Set the value of field beta.
- void `setBlurKernelSize (int blurKernelSize)`
Set the value of field blurKernelSize.
- void `setEpsilon (int epsilon)`
Set the value of field epsilon.
- void `setMorphologicalCloselterations (int morphologicalCloselterations)`
Set the value of field morphologicalCloselterations.
- void `setOriginXCoordinate (int originXCoordinate)`
Set the value of field originXCoordinate.
- void `setOriginYCoordinate (int originYCoordinate)`
Set the value of field originYCoordinate.
- void `setRegionAreaThresh (int regionAreaThresh)`
Set the value of field regionAreaThresh.
- void `setThresholdValue (int thresholdValue)`
Set the value of field thresholdValue.

Private Member Functions

- void `initialiseDetectorSpecificFields ()` override
Initialise the vision members.
- void `initialiseDetectorSpecificImageDependentFields ()` override
Initialisation of the detector specific image dependent values.
- void `createDetectorSpecificTrackbars ()` override
Create the trackbars.
- void `processImageAndDetect ()` override
Process the given image.
- void `changeContrastAndBrightness (Mat &processedImage)`
Change the contrast and brightness of the image.
- void `smoothImage (Mat &image)`
Smooth out differences in the image.

- void **morphologicalClose** (Mat &image)

Apply the morphological close operator on the image.
- void **thresholdImage** (const Mat &image, Mat &thresholdedImage)

Apply the threshold filter on the image.
- void **findRegions** (const Mat &image, vector< Region > ®ions)

Find the regions in the image.
- void **computeAverageMeasures** (vector< Region > ®ions)

Compute the average clusteredness degree and average density.
- vector< vector< Point > > **findContoursInImage** (const Mat &image)

Find contours in image.
- Region **createRegionFromPolygon** (const vector< Point > &polygon)

Create a new region from the given polygon.
- bool **isValidRegion** (const vector< Point > &polygon)

Check if the region is valid.
- double **regionClusterednessDegree** (const vector< Point > &polygon)

Compute the clusteredness degree of the region delimited by the given polygon.
- double **regionDensity** (const vector< Point > &polygon)

Compute the density of the area delimited by the given polygon.
- double **regionArea** (const vector< Point > &polygon)

Compute the area of the given polygon considering holes.
- double **regionHolesArea** (const Mat &matrix)

Compute the area of the white holes in the given matrix.
- void **clearPreviousDetectionResults** () override

Clear the element present in the regions vector.
- void **outputResultsToCsvFile** (ofstream &fout) override

Output the regions and averaged measures to a csv file.
- void **outputRegionsToCsvFile** (ofstream &fout)

Output the regions to a csv file.
- void **outputAveragedMeasuresToCsvFile** (ofstream &fout)

Output the averaged measures to a csv file.
- void **outputResultsToImage** () override

Output the results to the outputImage instance.
- double **convertAlpha** (int alpha)

Convert alpha from the range [0, ALPHA_MAX] to [ALPHA_REAL_MIN, ALPHA_REAL_MAX].
- int **convertBeta** (int beta)

Convert beta from the range [0, BETA_MAX] to [BETA_REAL_MIN, BETA_REAL_MAX].

Private Attributes

- double avgClusterednessDegree
- double avgDensity
- int alpha
- int beta
- int blurKernelSize
- int morphologicalCloselterations
- int epsilon
- int regionAreaThresh
- int thresholdValue
- vector< Region > regions

Additional Inherited Members

7.38.1 Detailed Description

Class for detecting regions of high intensity in grayscale images.

Definition at line 56 of file RegionDetector.hpp.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 RegionDetector::RegionDetector (bool *debugMode* = false)

Definition at line 15 of file RegionDetector.cpp.

References alpha, avgClusterednessDegree, avgDensity, beta, blurKernelSize, epsilon, morphologicalCloseIterations, regionAreaThresh, and thresholdValue.

7.38.2.2 RegionDetector::~RegionDetector ()

Definition at line 28 of file RegionDetector.cpp.

7.38.3 Member Function Documentation

7.38.3.1 void RegionDetector::changeContrastAndBrightness (Mat & *processedImage*) [private]

Change the contrast and brightness of the image.

Change the contrast and brightness of the image by the factors alpha and gamma

Parameters

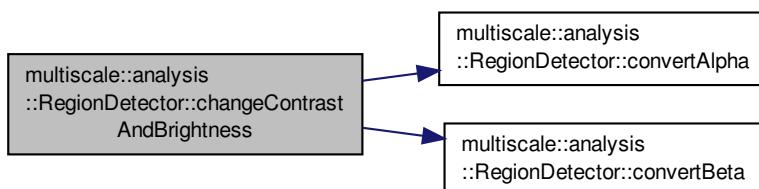
<i>processedImage</i>	The processed image
-----------------------	---------------------

Definition at line 158 of file RegionDetector.cpp.

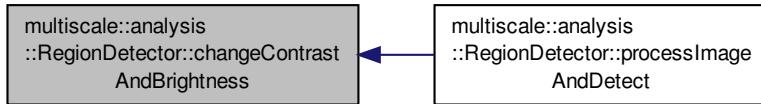
References alpha, beta, convertAlpha(), convertBeta(), and multiscale::analysis::Detector::image.

Referenced by processImageAndDetect().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.2 void RegionDetector::clearPreviousDetectionResults() [override], [private], [virtual]

Clear the element present in the regions vector.

Implements [multiscale::analysis::Detector](#).

Definition at line 288 of file RegionDetector.cpp.

References regions.

7.38.3.3 void RegionDetector::computeAverageMeasures(vector< Region > & regions) [private]

Compute the average clusteredness degree and average density.

Parameters

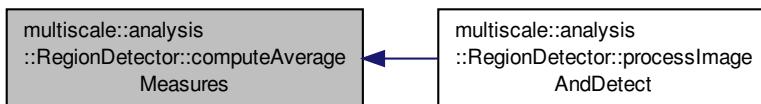
<i>regions</i>	The regions in the image
----------------	--------------------------

Definition at line 198 of file RegionDetector.cpp.

References avgClusterednessDegree, and avgDensity.

Referenced by processImageAndDetect().

Here is the caller graph for this function:



7.38.3.4 double RegionDetector::convertAlpha(int alpha) [private]

Convert alpha from the range [0, ALPHA_MAX] to [ALPHA_REAL_MIN, ALPHA_REAL_MAX].

Parameters

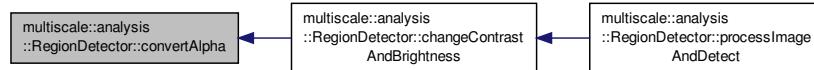
<i>alpha</i>	Alpha
--------------	-------

Definition at line 335 of file RegionDetector.cpp.

References alpha, ALPHA_MAX, ALPHA_REAL_MAX, and ALPHA_REAL_MIN.

Referenced by changeContrastAndBrightness().

Here is the caller graph for this function:



7.38.3.5 int RegionDetector::convertBeta (int beta) [private]

Convert beta from the range [0, BETA_MAX] to [BETA_REAL_MIN, BETA_REAL_MAX].

Parameters

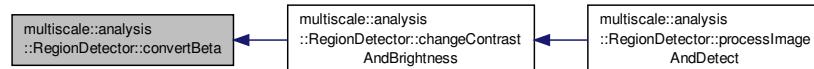
<i>beta</i>	Beta
-------------	------

Definition at line 339 of file RegionDetector.cpp.

References beta, BETA_MAX, BETA_REAL_MAX, and BETA_REAL_MIN.

Referenced by changeContrastAndBrightness().

Here is the caller graph for this function:



7.38.3.6 void RegionDetector::createDetectorSpecificTrackbars () [override], [private], [virtual]

Create the trackbars.

Implements [multiscale::analysis::Detector](#).

Definition at line 136 of file RegionDetector.cpp.

References alpha, ALPHA_MAX, beta, BETA_MAX, blurKernelSize, epsilon, EPSILON_MAX, KERNEL_MAX, MORPH_ITER_MAX, morphologicalCloselterations, REGION_AREA_THRESH_MAX, regionAreaThresh, THRESHOLD_MAX, thresholdValue, TRACKBAR_ALPHA, TRACKBAR_BETA, TRACKBAR_EPSILON, TRACKBAR_KERNEL, TRACKBAR_MORPH, TRACKBAR_REGION_AREA_THRESH, TRACKBAR_THRESHOLD, and WIN_OUTPUT_IMAGE.

7.38.3.7 Region RegionDetector::createRegionFromPolygon (const vector< Point > & polygon) [private]

Create a new region from the given polygon.

Process the polygon in order to get the required information and create a region using this information

Parameters

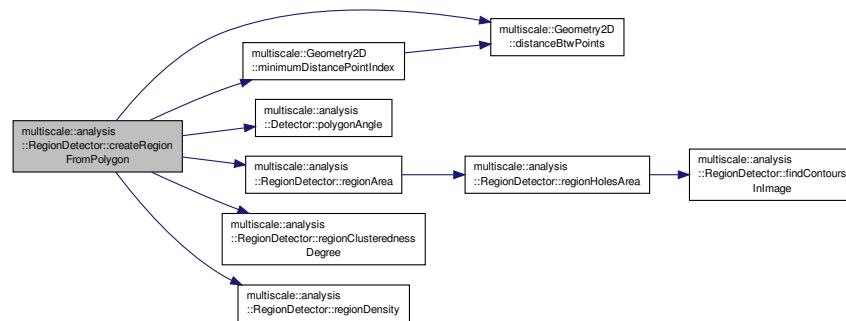
<i>polygon</i>	Polygon determining the region
----------------	--------------------------------

Definition at line 226 of file RegionDetector.cpp.

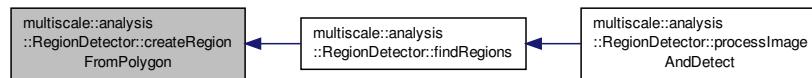
References multiscale::Geometry2D::distanceBtwPoints(), multiscale::Geometry2D::minimumDistancePointIndex(), multiscale::analysis::Detector::origin, multiscale::analysis::Detector::polygonAngle(), regionArea(), regionClusterednessDegree(), and regionDensity().

Referenced by findRegions().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.8 `vector< vector< Point > > RegionDetector::findContoursInImage (const Mat & image) [private]`

Find contours in image.

Parameters

<i>image</i>	The image
--------------	-----------

Definition at line 211 of file RegionDetector.cpp.

Referenced by findRegions(), and regionHolesArea().

Here is the caller graph for this function:



7.38.3.9 void RegionDetector::findRegions (const Mat & *image*, vector< Region > & *regions*) [private]

Find the regions in the image.

Find the contours, approximate the polygons and extract the required information from them.

Parameters

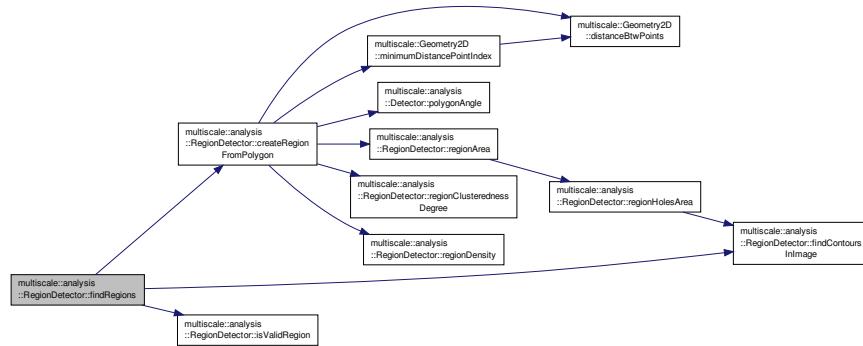
<i>image</i>	The image
<i>regions</i>	The regions in the image

Definition at line 180 of file RegionDetector.cpp.

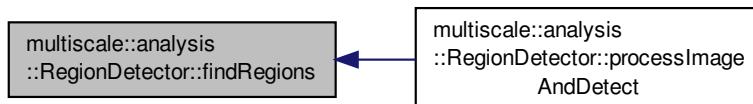
References createRegionFromPolygon(), epsilon, findContoursInImage(), and isValidRegion().

Referenced by processImageAndDetect().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.10 int RegionDetector::getAlpha()

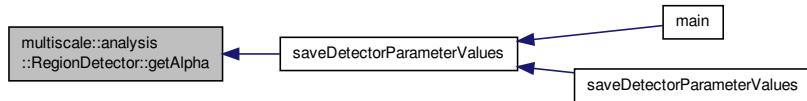
Get the value of field alpha.

Definition at line 30 of file RegionDetector.cpp.

References alpha.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.38.3.11 int RegionDetector::getBeta()

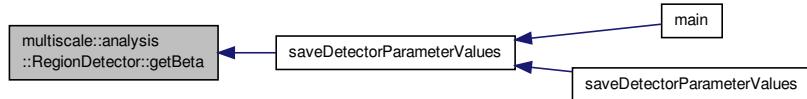
Get the value of field beta.

Definition at line 34 of file RegionDetector.cpp.

References beta.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.38.3.12 int RegionDetector::getBlurKernelSize()

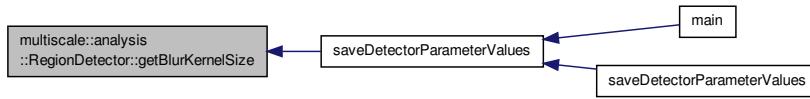
Get the value of field blurKernelSize.

Definition at line 38 of file RegionDetector.cpp.

References blurKernelSize.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.38.3.13 int RegionDetector::getEpsilon()

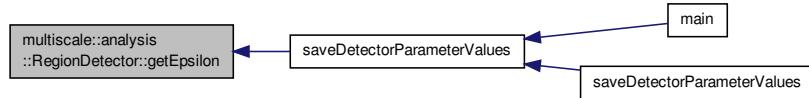
Get the value of field epsilon.

Definition at line 46 of file RegionDetector.cpp.

References epsilon.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.38.3.14 int RegionDetector::getMorphologicalCloselterations ()

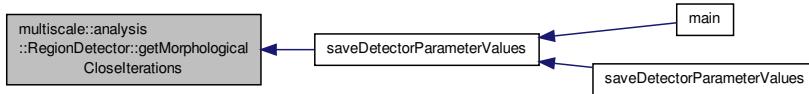
Get the value of field morphologicalCloselterations.

Definition at line 42 of file RegionDetector.cpp.

References morphologicalCloselterations.

Referenced by saveDetectorParameterValues().

Here is the caller graph for this function:



7.38.3.15 int RegionDetector::getOriginXCoordinate ()

Get the value of field originXCoordinate.

Definition at line 54 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin.

7.38.3.16 int RegionDetector::getOriginYCoordinate ()

Get the value of field originYCoordinate.

Definition at line 58 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin.

7.38.3.17 int RegionDetector::getRegionAreaThresh ()

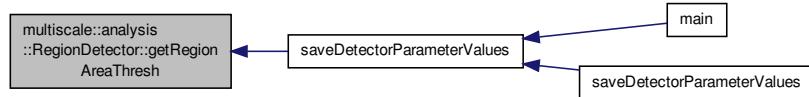
Get the value of field regionAreaThresh.

Definition at line 50 of file RegionDetector.cpp.

References regionAreaThresh.

Referenced by `saveDetectorParameterValues()`.

Here is the caller graph for this function:



7.38.3.18 `vector< Region > const & RegionDetector::getRegions()`

Get a const reference to the vector of detected regions.

Definition at line 66 of file `RegionDetector.cpp`.

References regions.

7.38.3.19 `int RegionDetector::getThresholdValue()`

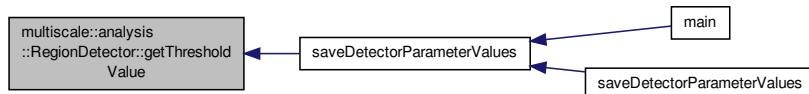
Get the value of field `thresholdValue`.

Definition at line 62 of file `RegionDetector.cpp`.

References `thresholdValue`.

Referenced by `saveDetectorParameterValues()`.

Here is the caller graph for this function:



7.38.3.20 `void RegionDetector::initialiseDetectorSpecificFields() [override], [private], [virtual]`

Initialise the vision members.

Implements [multiscale::analysis::Detector](#).

Definition at line 124 of file `RegionDetector.cpp`.

References alpha, beta, blurKernelSize, epsilon, morphologicalCloselterations, regionAreaThresh, and `thresholdValue`.

7.38.3.21 `void RegionDetector::initialiseDetectorSpecificImageDependentFields() [override], [private], [virtual]`

Initialisation of the detector specific image dependent values.

Implements [multiscale::analysis::Detector](#).

Definition at line 134 of file RegionDetector.cpp.

7.38.3.22 bool RegionDetector::isValidRegion (const vector< Point > & *polygon*) [private]

Check if the region is valid.

Check if the area of the region > regionAreaThreshold

Parameters

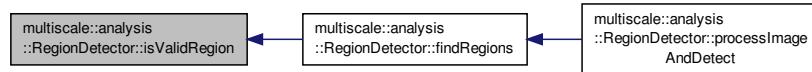
<i>polygon</i>	The polygon defining the region
----------------	---------------------------------

Definition at line 238 of file RegionDetector.cpp.

References CONTOUR_AREA_ORIENTED, and regionAreaThresh.

Referenced by findRegions().

Here is the caller graph for this function:



7.38.3.23 void RegionDetector::morphologicalClose (Mat & *image*) [private]

Apply the morphological close operator on the image.

Parameters

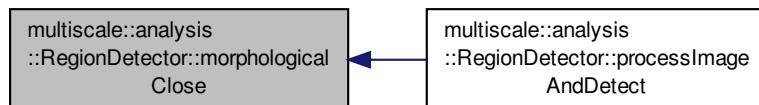
<i>image</i>	The image
--------------	-----------

Definition at line 170 of file RegionDetector.cpp.

References morphologicalCloselterations.

Referenced by processImageAndDetect().

Here is the caller graph for this function:



7.38.3.24 void RegionDetector::outputAveragedMeasuresToCsvFile (ofstream & *fout*) [private]

Output the averaged measures to a csv file.

Parameters

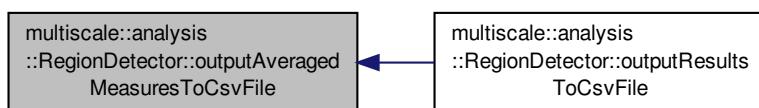
<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 312 of file RegionDetector.cpp.

References avgClusterednessDegree, avgDensity, OUTPUT_CLUSTEREDNESS, and OUTPUT_DENSITY.

Referenced by outputResultsToCsvFile().

Here is the caller graph for this function:



7.38.3.25 void RegionDetector::outputRegionsToCsvFile (ostream & *fout*) [private]

Output the regions to a csv file.

Parameters

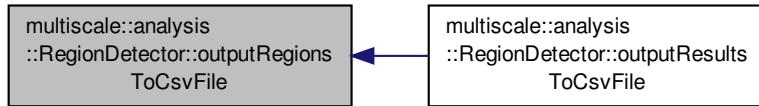
<i>fout</i>	Output file stream
-------------	--------------------

Definition at line 306 of file RegionDetector.cpp.

References regions.

Referenced by outputResultsToCsvFile().

Here is the caller graph for this function:



7.38.3.26 void RegionDetector::outputResultsToCsvFile (ostream & *fout*) [override], [private], [virtual]

Output the regions and averaged measures to a csv file.

Parameters

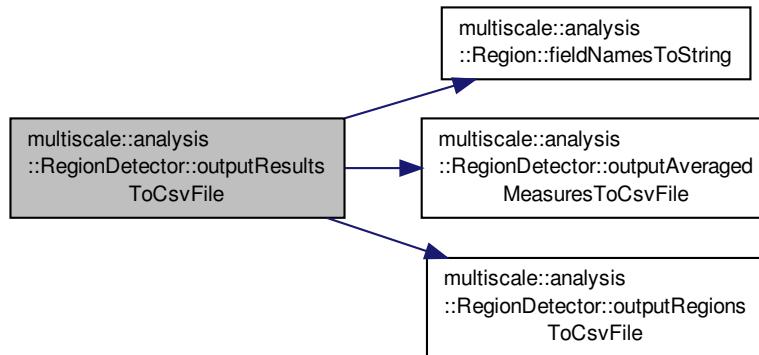
<i>fout</i>	Output file stream
-------------	--------------------

Implements [multiscale::analysis::Detector](#).

Definition at line 292 of file RegionDetector.cpp.

References `multiscale::analysis::Region::fieldNamesToString()`, `outputAveragedMeasuresToCsvFile()`, `outputRegionsToCsvFile()`, and `regions`.

Here is the call graph for this function:



7.38.3.27 void RegionDetector::outputResultsToImage() [override], [private], [virtual]

Output the results to the `outputImage` instance.

Implements [multiscale::analysis::Detector](#).

Definition at line 317 of file RegionDetector.cpp.

References `DISPLAY_LINE_THICKNESS`, `multiscale::analysis::Detector::image`, `INTENSITY_MAX`, `multiscale::analysis::Detector::outputImage`, `POLYGON_CLOSED`, and `regions`.

7.38.3.28 void RegionDetector::processImageAndDetect() [override], [private], [virtual]

Process the given image.

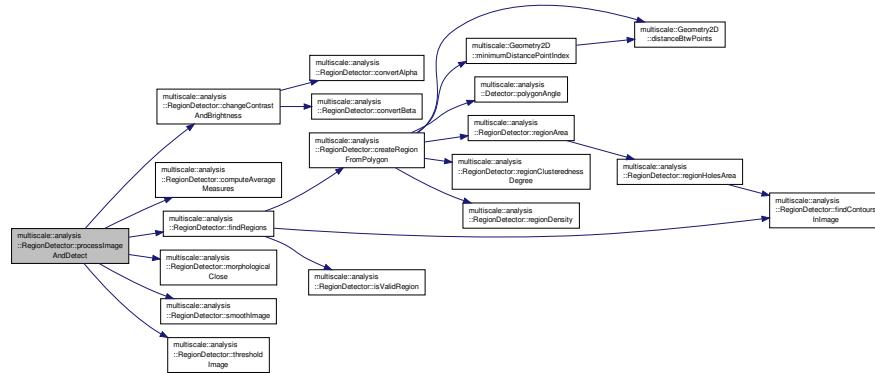
Apply filters to the image, threshold it, find its contours, approximate the polygons from these contours. Afterwards, process the polygons to find their distance from the origin, their area and the angle determined by the points from the contour which are on the edge and the closest point to the origin. Return all the polygons together with the processed information as a vector of regions.

Implements [multiscale::analysis::Detector](#).

Definition at line 146 of file RegionDetector.cpp.

References `changeContrastAndBrightness()`, `computeAverageMeasures()`, `findRegions()`, `morphologicalClose()`, `regions`, `smoothImage()`, and `thresholdImage()`.

Here is the call graph for this function:



7.38.3.29 double RegionDetector::regionArea (const vector< Point > & polygon) [private]

Compute the area of the given polygon considering holes.

Parameters

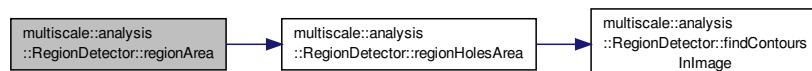
<i>polygon</i>	The given polygon
----------------	-------------------

Definition at line 266 of file RegionDetector.cpp.

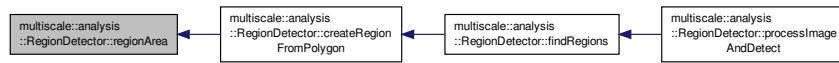
References CONTOUR_AREA_ORIENTED, multiscale::analysis::Detector::image, INTENSITY_MAX, regionHolesArea(), THRESHOLD_CLUSTEREDNESS, and THRESHOLD_MAX.

Referenced by createRegionFromPolygon().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.30 double RegionDetector::regionClusterednessDegree (const vector< Point > & polygon) [private]

Compute the clusteredness degree of the region delimited by the given polygon.

The density is equal to the average intensity of the pixels in the area delimited by the given polygon divided by

INTENSITY_MAX. The intensities are considered from the thresholded image where the threshold value is 1 i.e. only the black patches are discarded.

Parameters

<i>polygon</i>	The given polygon
----------------	-------------------

Definition at line 244 of file RegionDetector.cpp.

References multiscale::analysis::Detector::image, INTENSITY_MAX, THRESHOLD_CLUSTEREDNESS, and THRESHOLD_MAX.

Referenced by createRegionFromPolygon().

Here is the caller graph for this function:



7.38.3.31 double RegionDetector::regionDensity (const vector< Point > & *polygon*) [private]

Compute the density of the area delimited by the given polygon.

The density is equal to the average intensity of the pixels in the area delimited by the given polygon divided by INTENSITY_MAX.

Parameters

<i>polygon</i>	The given polygon
----------------	-------------------

Definition at line 256 of file RegionDetector.cpp.

References multiscale::analysis::Detector::image, and INTENSITY_MAX.

Referenced by createRegionFromPolygon().

Here is the caller graph for this function:



7.38.3.32 double RegionDetector::regionHolesArea (const Mat & *matrix*) [private]

Compute the area of the white holes in the given matrix.

Parameters

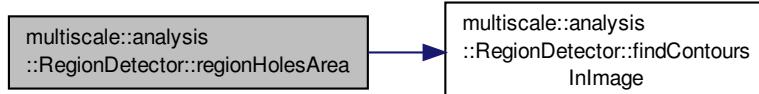
<i>matrix</i>	The given matrix
---------------	------------------

Definition at line 277 of file RegionDetector.cpp.

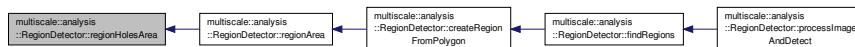
References CONTOUR_AREA_ORIENTED, and findContoursInImage().

Referenced by `regionArea()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.33 void RegionDetector::setAlpha (int *alpha*)

Set the value of field *alpha*.

Parameters

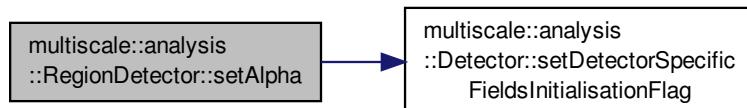
<i>alpha</i>	Value of <i>alpha</i>
--------------	-----------------------

Definition at line 70 of file `RegionDetector.cpp`.

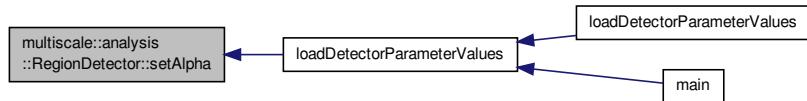
References *alpha*, and `multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()`.

Referenced by `loadDetectorParameterValues()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.34 void RegionDetector::setBeta (int *beta*)

Set the value of field beta.

Parameters

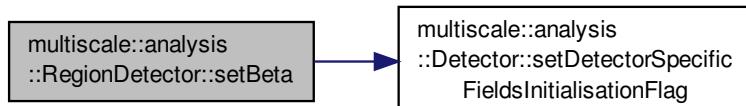
<i>beta</i>	Value of beta
-------------	---------------

Definition at line 76 of file RegionDetector.cpp.

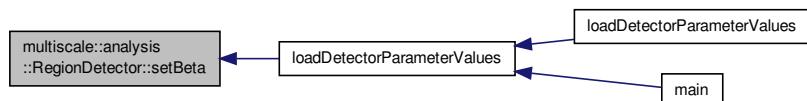
References beta, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Referenced by loadDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.35 void RegionDetector::setBlurKernelSize (int *blurKernelSize*)

Set the value of field blurKernelSize.

Parameters

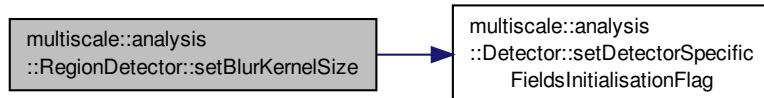
<i>blurKernelSize</i>	Value of blurKernelSize
-----------------------	-------------------------

Definition at line 82 of file RegionDetector.cpp.

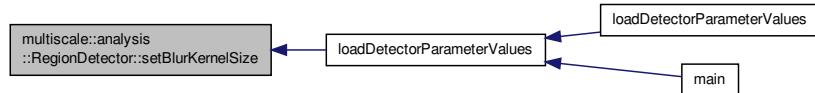
References blurKernelSize, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Referenced by loadDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.36 void RegionDetector::setEpsilon (int epsilon)

Set the value of field epsilon.

Parameters

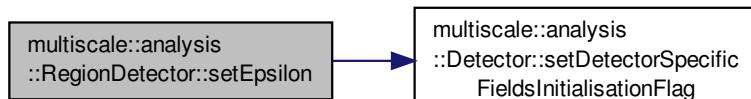
<i>epsilon</i>	Value of epsilon
----------------	------------------

Definition at line 88 of file RegionDetector.cpp.

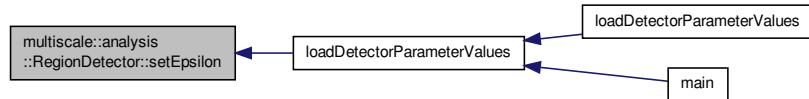
References epsilon, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Referenced by loadDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.37 void RegionDetector::setMorphologicalCloselterations (int *morphologicalCloselterations*)

Set the value of field *morphologicalCloselterations*.

Parameters

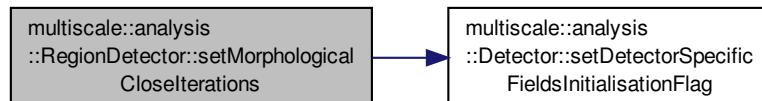
<i>morphological- Closelterations</i>	Value of <i>morphologicalCloselterations</i>
---	--

Definition at line 94 of file RegionDetector.cpp.

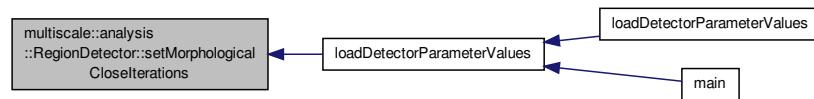
References *morphologicalCloselterations*, and *multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag()*.

Referenced by *loadDetectorParameterValues()*.

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.38 void RegionDetector::setOriginXCoordinate (int *originXCoordinate*)

Set the value of field *originXCoordinate*.

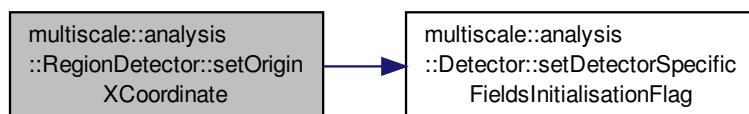
Parameters

<i>originX-</i> <i>Coordinate</i>	Value of originXCoordinate
--------------------------------------	----------------------------

Definition at line 100 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Here is the call graph for this function:



7.38.3.39 void RegionDetector::setOriginYCoordinate (int *originYCoordinate*)

Set the value of field originYCoordinate.

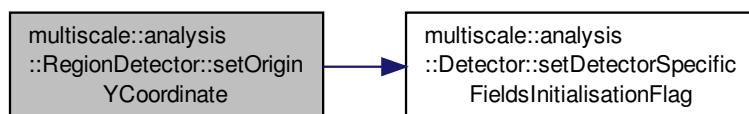
Parameters

<i>originY-</i> <i>Coordinate</i>	Value of originYCoordinate
--------------------------------------	----------------------------

Definition at line 106 of file RegionDetector.cpp.

References multiscale::analysis::Detector::origin, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Here is the call graph for this function:



7.38.3.40 void RegionDetector::setRegionAreaThresh (int *regionAreaThresh*)

Set the value of field regionAreaThresh.

Parameters

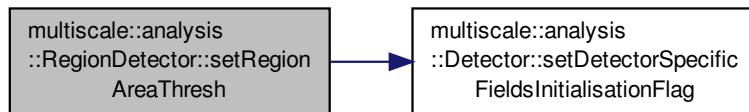
<i>regionArea-</i> <i>Thresh</i>	Value of regionAreaThresh
-------------------------------------	---------------------------

Definition at line 112 of file RegionDetector.cpp.

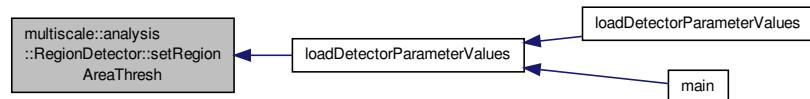
References regionAreaThresh, and multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag().

Referenced by loadDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.41 void RegionDetector::setThresholdValue (int *thresholdValue*)

Set the value of field thresholdValue.

Parameters

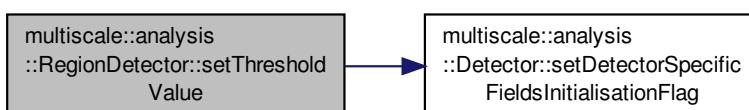
<i>thresholdValue</i>	Value of thresholdValue
-----------------------	-------------------------

Definition at line 118 of file RegionDetector.cpp.

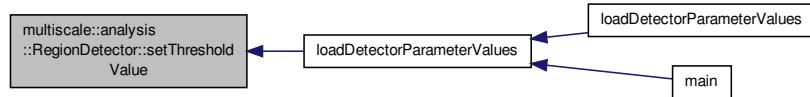
References multiscale::analysis::Detector::setDetectorSpecificFieldsInitialisationFlag(), and thresholdValue.

Referenced by loadDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:



7.38.3.42 void RegionDetector::smoothImage (Mat & *image*) [private]

Smooth out differences in the image.

Apply a Gaussian blur filter

Parameters

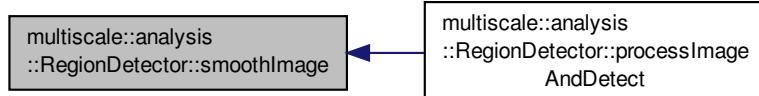
<i>image</i>	The image
--------------	-----------

Definition at line 162 of file RegionDetector.cpp.

References blurKernelSize.

Referenced by processImageAndDetect().

Here is the caller graph for this function:



7.38.3.43 void RegionDetector::thresholdImage (const Mat & *image*, Mat & *thresholdedImage*) [private]

Apply the threshold filter on the image.

Parameters

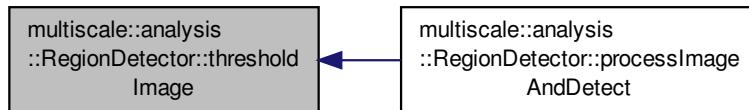
<i>image</i>	The image
<i>thresholdedImage</i>	The thresholded image

Definition at line 176 of file RegionDetector.cpp.

References THRESHOLD_MAX, and thresholdValue.

Referenced by processImageAndDetect().

Here is the caller graph for this function:



7.38.4 Member Data Documentation

7.38.4.1 int multiscale::analysis::RegionDetector::alpha [private]

Alpha for brightness and contrast adjustments

Definition at line 63 of file RegionDetector.hpp.

Referenced by changeContrastAndBrightness(), convertAlpha(), createDetectorSpecificTrackbars(), getAlpha(), initialiseDetectorSpecificFields(), RegionDetector(), and setAlpha().

7.38.4.2 double multiscale::analysis::RegionDetector::avgClusterednessDegree [private]

Average degree of clusteredness of all regions

Definition at line 60 of file RegionDetector.hpp.

Referenced by computeAverageMeasures(), outputAveragedMeasuresToCsvFile(), and RegionDetector().

7.38.4.3 double multiscale::analysis::RegionDetector::avgDensity [private]

Average density of all regions

Definition at line 61 of file RegionDetector.hpp.

Referenced by computeAverageMeasures(), outputAveragedMeasuresToCsvFile(), and RegionDetector().

7.38.4.4 int multiscale::analysis::RegionDetector::beta [private]

Beta for brightness and contrast adjustments

Definition at line 64 of file RegionDetector.hpp.

Referenced by changeContrastAndBrightness(), convertBeta(), createDetectorSpecificTrackbars(), getBeta(), initialiseDetectorSpecificFields(), RegionDetector(), and setBeta().

7.38.4.5 int multiscale::analysis::RegionDetector::blurKernelSize [private]

Kernel size for Gaussian blur

Definition at line 65 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getBlurKernelSize(), initialiseDetectorSpecificFields(), RegionDetector(), setBlurKernelSize(), and smoothImage().

7.38.4.6 int multiscale::analysis::RegionDetector::epsilon [private]

Epsilon for polygon approximation

Definition at line 67 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), findRegions(), getEpsilon(), initialiseDetectorSpecificFields(), RegionDetector(), and setEpsilon().

7.38.4.7 int multiscale::analysis::RegionDetector::morphologicalCloselterations [private]

Number of iterations for morphological close operator

Definition at line 66 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getMorphologicalCloselterations(), initialiseDetectorSpecificFields(), morphologicalClose(), RegionDetector(), and setMorphologicalCloselterations().

7.38.4.8 int multiscale::analysis::RegionDetector::regionAreaThresh [private]

Threshold for considering a region

Definition at line 68 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getRegionAreaThresh(), initialiseDetectorSpecificFields(), is-ValidRegion(), RegionDetector(), and setRegionAreaThresh().

7.38.4.9 vector<Region> multiscale::analysis::RegionDetector::regions [private]

Regions detected in the image

Definition at line 71 of file RegionDetector.hpp.

Referenced by clearPreviousDetectionResults(), getRegions(), outputRegionsToCsvFile(), outputResultsToCsvFile(), outputResultsToImage(), and processImageAndDetect().

7.38.4.10 int multiscale::analysis::RegionDetector::thresholdValue [private]

Value of the threshold for the threshold filter

Definition at line 69 of file RegionDetector.hpp.

Referenced by createDetectorSpecificTrackbars(), getThresholdValue(), initialiseDetectorSpecificFields(), RegionDetector(), setThresholdValue(), and thresholdImage().

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

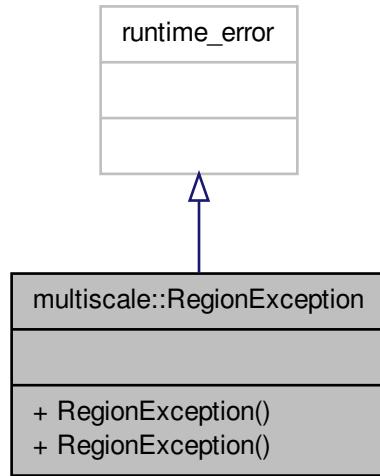
- include/multiscale/analysis/spatial/RegionDetector.hpp
- src/multiscale/analysis/spatial/RegionDetector.cpp

7.39 multiscale::RegionException Class Reference

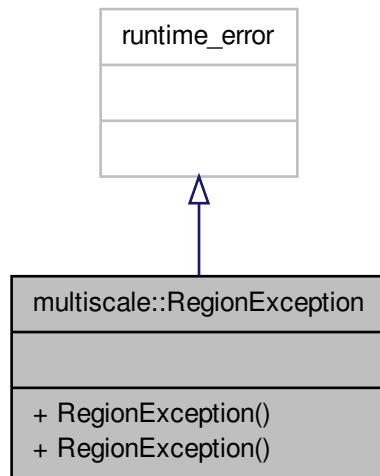
Exception class for the Region instances.

```
#include <RegionException.hpp>
```

Inheritance diagram for multiscale::RegionException:



Collaboration diagram for multiscale::RegionException:



Public Member Functions

- [RegionException \(const string &msg\)](#)
- [RegionException \(const char *msg\)](#)

7.39.1 Detailed Description

Exception class for the Region instances.

Definition at line 13 of file RegionException.hpp.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 multiscale::RegionException::RegionException (const string & msg) [inline]

Definition at line 17 of file RegionException.hpp.

7.39.2.2 multiscale::RegionException::RegionException (const char * msg) [inline]

Definition at line 18 of file RegionException.hpp.

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

- [include/multiscale/exception/RegionException.hpp](#)

7.40 multiscale::RGBColourGenerator Class Reference

Generate a RGB colour.

```
#include <RGBColourGenerator.hpp>
```

Collaboration diagram for multiscale::RGBColourGenerator:

multiscale::RGBColourGenerator
<ul style="list-style-type: none"> - red - green - blue
<ul style="list-style-type: none"> + generate() + generate() - convertHSVToRGB() - computeRGBValues() - convertRGBToString()

Static Public Member Functions

- static string [generate](#) (double concentrationMin, double concentrationMax, double concentration)

Generate a RGB colour for the given concentration.
- static Scalar [generate](#) (RNG &randomNumberGenerator)

Generate a random RGB colour.

Static Private Member Functions

- static string `convertHSVToRGB` (double hue, double saturation, double value)
Convert a colour from HSV to RGB colour space.
- static void `computeRGBValues` (int huePrime, double X, double chroma, double m)
Compute RGB values from HSV specific values.
- static string `convertRGBToString` ()
Convert the RGB colour to a string.

Static Private Attributes

- static double `red` = 0
- static double `green` = 0
- static double `blue` = 0

7.40.1 Detailed Description

Generate a RGB colour.

Generate a RGB colour given the possible range for concentrations and the value of one of the concentrations

The conversion HSV->RGB is based on the wikipedia page on this topic

Definition at line 25 of file RGBColourGenerator.hpp.

7.40.2 Member Function Documentation

7.40.2.1 void RGBColourGenerator::computeRGBValues (int *huePrime*, double *X*, double *chroma*, double *m*) [static], [private]

Compute RGB values from HSV specific values.

Parameters

<code>huePrime</code>	Hue'
<code>X</code>	X
<code>chroma</code>	Chroma
<code>m</code>	m

Definition at line 47 of file RGBColourGenerator.cpp.

7.40.2.2 string RGBColourGenerator::convertHSVToRGB (double *hue*, double *saturation*, double *value*) [static], [private]

Convert a colour from HSV to RGB colour space.

Parameters

<code>hue</code>	Hue
<code>saturation</code>	Saturation
<code>value</code>	Value

Definition at line 33 of file RGBColourGenerator.cpp.

7.40.2.3 string RGBColourGenerator::convertRGBToString() [static], [private]

Convert the RGB colour to a string.

Definition at line 92 of file RGBColourGenerator.cpp.

7.40.2.4 string RGBColourGenerator::generate(double concentrationMin, double concentrationMax, double concentration) [static]

Generate a RGB colour for the given concentration.

Generate a RGB colour considering the range of values a concentration can have and the value of the concentration

Parameters

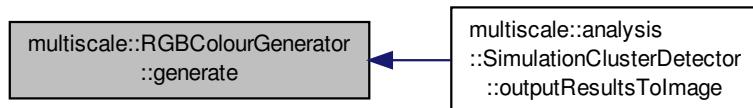
<i>concentration-Min</i>	The minimum of the range of values a concentration can take
<i>concentration-Max</i>	The maximum of the range of values a concentration can take
<i>concentration</i>	The concentration

Definition at line 17 of file RGBColourGenerator.cpp.

References HUE_MAX, HUE_MIN, SATURATION, and VALUE.

Referenced by multiscale::analysis::SimulationClusterDetector::outputResultsToImage().

Here is the caller graph for this function:



7.40.2.5 Scalar RGBColourGenerator::generate(RNG & randomNumberGenerator) [static]

Generate a random RGB colour.

Generate a random RGB colour using the given random number generator

Parameters

<i>randomNumber-Generator</i>	Random number generator
-------------------------------	-------------------------

Definition at line 27 of file RGBColourGenerator.cpp.

7.40.3 Member Data Documentation

7.40.3.1 double RGBColourGenerator::blue = 0 [static], [private]

The amount of blue

Definition at line 31 of file RGBColourGenerator.hpp.

7.40.3.2 double RGBColourGenerator::green = 0 [static], [private]

The amount of green

Definition at line 30 of file RGBColourGenerator.hpp.

7.40.3.3 double RGBColourGenerator::red = 0 [static], [private]

The amount of red

Definition at line 29 of file RGBColourGenerator.hpp.

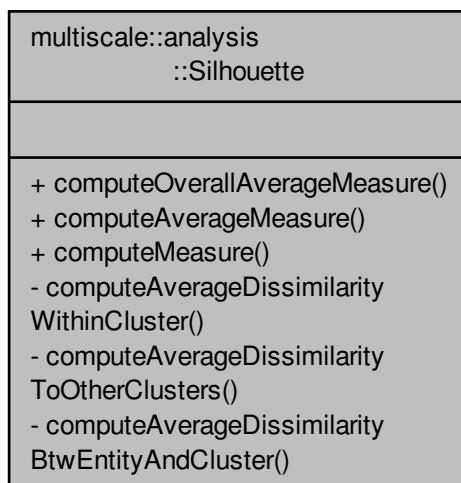
The documentation for this class was generated from the following files:

- include/multiscale/util/[RGBColourGenerator.hpp](#)
- src/multiscale/util/[RGBColourGenerator.cpp](#)

7.41 multiscale::analysis::Silhouette Class Reference

```
#include <Silhouette.hpp>
```

Collaboration diagram for multiscale::analysis::Silhouette:



Static Public Member Functions

- static double [computeOverallAverageMeasure](#) (const vector< [Cluster](#) > &clusters)
Compute the overall average silhouette measure for the given collection of clusters.
- static double [computeAverageMeasure](#) (unsigned int clusterIndex, const vector< [Cluster](#) > &clusters)
Compute the average silhouette measure for the given cluster.

- static double `computeMeasure` (unsigned int entityIndex, unsigned int clusterIndex, const vector< `Cluster` > &clusters)

Compute the silhouette measure for the given entity.

Static Private Member Functions

- static double `computeAverageDissimilarityWithinCluster` (unsigned int entityIndex, unsigned int clusterIndex, const vector< `Cluster` > &clusters)

Compute the average dissimilarity within cluster to which the entity belongs.

- static double `computeAverageDissimilarityToOtherClusters` (unsigned int entityIndex, unsigned int clusterIndex, const vector< `Cluster` > &clusters)

Compute the average dissimilarity of the entity to the other clusters (i.e. clusters which are different from the cluster to which the entity belongs)

- static double `computeAverageDissimilarityBtwEntityAndCluster` (unsigned int entityIndex, unsigned int entityClusterIndex, unsigned int clusterIndex, const vector< `Cluster` > &clusters)

Compute the average dissimilarity between entity and cluster.

7.41.1 Detailed Description

Definition at line 13 of file Silhouette.hpp.

7.41.2 Member Function Documentation

7.41.2.1 double `Silhouette::computeAverageDissimilarityBtwEntityAndCluster` (`unsigned int entityIndex, unsigned int entityClusterIndex, unsigned int clusterIndex, const vector< Cluster > & clusters`) [static], [private]

Compute the average dissimilarity between entity and cluster.

Parameters

<code>entityIndex</code>	The index of the entity in the cluster for which the distance is computed
<code>entityClusterIndex</code>	The index of the cluster to which the entity belongs
<code>clusterIndex</code>	The index of the cluster to which the average distance is computed
<code>clusters</code>	Collection of all clusters

Definition at line 84 of file Silhouette.cpp.

References multiscale::Geometry2D::distanceBtwPoints().

Here is the call graph for this function:



7.41.2.2 double Silhouette::computeAverageDissimilarityToOtherClusters (*unsigned int entityIndex, unsigned int clusterIndex, const vector< Cluster > & clusters*) [static], [private]

Compute the average dissimilarity of the entity to the other clusters (i.e. clusters which are different from the cluster to which the entity belongs)

Parameters

<i>entityIndex</i>	The index of the entity in the cluster for which the silhouette measure is computed
<i>clusterIndex</i>	The index of the cluster to which the entity belongs
<i>clusters</i>	Collection of all clusters

Definition at line 66 of file Silhouette.cpp.

7.41.2.3 double Silhouette::computeAverageDissimilarityWithinCluster (*unsigned int entityIndex, unsigned int clusterIndex, const vector< Cluster > & clusters*) [static], [private]

Compute the average dissimilarity within cluster to which the entity belongs.

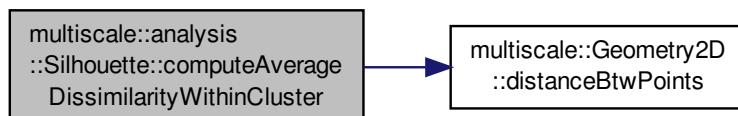
Parameters

<i>entityIndex</i>	The index of the entity in the cluster for which the silhouette measure is computed
<i>clusterIndex</i>	The index of the cluster to which the entity belongs
<i>clusters</i>	Collection of all clusters

Definition at line 52 of file Silhouette.cpp.

References multiscale::Geometry2D::distanceBtwPoints().

Here is the call graph for this function:



7.41.2.4 double Silhouette::computeAverageMeasure (*unsigned int clusterIndex, const vector< Cluster > & clusters*) [static]

Compute the average silhouette measure for the given cluster.

Parameters

<i>clusterIndex</i>	The index of the cluster for which the average silhouette measure is computed
<i>clusters</i>	Collection of all clusters

Definition at line 26 of file Silhouette.cpp.

7.41.2.5 double Silhouette::computeMeasure (unsigned int *entityIndex*, unsigned int *clusterIndex*, const vector< Cluster > & *clusters*) [static]

Compute the silhouette measure for the given entity.

Parameters

<i>entityIndex</i>	The index of the entity in the cluster for which the silhouette measure is computed
<i>clusterIndex</i>	The index of the cluster to which the entity belongs
<i>clusters</i>	Collection of all clusters

Definition at line 42 of file Silhouette.cpp.

7.41.2.6 double Silhouette::computeOverallAverageMeasure (const vector< Cluster > & *clusters*) [static]

Compute the overall average silhouette measure for the given collection of clusters.

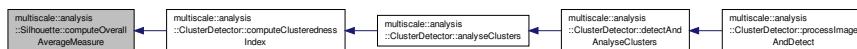
Parameters

<i>clusters</i>	Collection of all clusters
-----------------	----------------------------

Definition at line 11 of file Silhouette.cpp.

Referenced by multiscale::analysis::ClusterDetector::computeClusterednessIndex().

Here is the caller graph for this function:



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

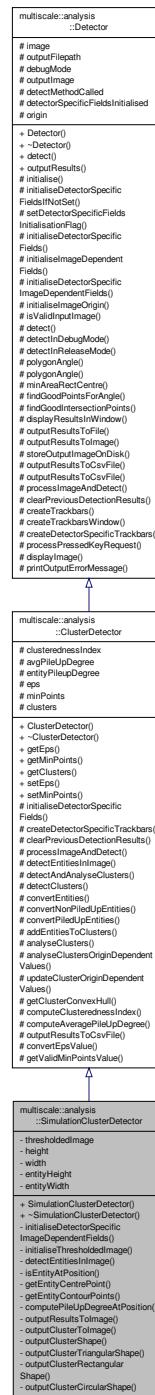
- include/multiscale/analysis/spatial/[Silhouette.hpp](#)
- src/multiscale/analysis/spatial/[Silhouette.cpp](#)

7.42 multiscale::analysis::SimulationClusterDetector Class Reference

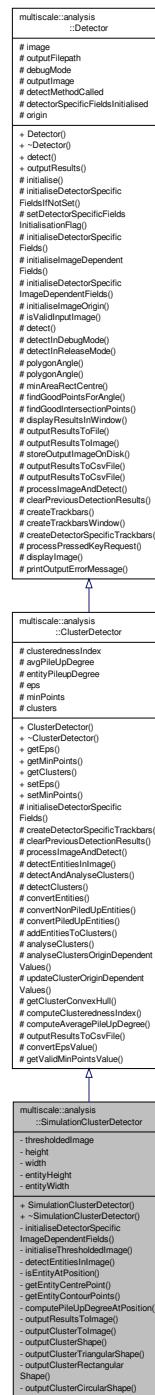
Class for detecting clusters in 2D images obtained from simulations.

```
#include <SimulationClusterDetector.hpp>
```

Inheritance diagram for multiscale::analysis::SimulationClusterDetector:



Collaboration diagram for multiscale::analysis::SimulationClusterDetector:



Public Member Functions

- `SimulationClusterDetector (unsigned int height, unsigned int width, int maxPileupNumber, double maxPileupIntensity, bool debugMode=false)`
- `~SimulationClusterDetector ()`

Private Member Functions

- void `initialiseDetectorSpecificImageDependentFields ()` override
Initialise the image dependent values.
- void `initialiseThresholdedImage ()`
Initialise the thresholdedImage field.
- void `detectEntitiesInImage (vector< Entity > &entities)` override
Detect the entities in the image.
- bool `isEntityAtPosition (int x, int y)`
Check if there is an entity in the image at the given position.
- Point2f `getEntityCentrePoint (int x, int y)`
Get the point representing the centre of the entity.
- vector< Point2f > `getEntityContourPoints (int x, int y)`
Get the points representing the contour of the entity.
- unsigned int `computePileUpDegreeAtPosition (int x, int y)`
Compute the pile up degree at the given position.
- void `outputResultsToImage ()` override
Display clusters on image.
- void `outputClusterToImage (Cluster &cluster, Scalar colour, Mat &image)`
Display cluster on the image.
- void `outputClusterShape (Cluster &cluster, Scalar colour, Mat &image)`
Draw the best matching shape (triangular, rectangular, circular) of the cluster on the image.
- void `outputClusterTriangularShape (Cluster &cluster, Scalar colour, Mat &image)`
Draw the best matching triangular shape of the cluster on the image.
- void `outputClusterRectangularShape (Cluster &cluster, Scalar colour, Mat &image)`
Draw the best matching rectangular shape of the cluster on the image.
- void `outputClusterCircularShape (Cluster &cluster, Scalar colour, Mat &image)`
Draw the best matching circular shape of the cluster on the image.

Private Attributes

- Mat `thresholdedImage`
- unsigned int `height`
- unsigned int `width`
- double `entityHeight`
- double `entityWidth`

Additional Inherited Members

7.42.1 Detailed Description

Class for detecting clusters in 2D images obtained from simulations.

Definition at line 26 of file SimulationClusterDetector.hpp.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 SimulationClusterDetector::SimulationClusterDetector (`unsigned int height, unsigned int width, int maxPileupNumber, double maxPileupIntensity, bool debugMode = false`)

Parameters

<i>height</i>	Height of the grid used in the simulation
<i>width</i>	Width of the grid used in the simulation
<i>debugMode</i>	Flag indicating if detector should run in debug mode or not
<i>maxPileup-Number</i>	The maximum number of entities which can occupy a grid position at the same time
<i>maxPileup-Intensity</i>	The grayscale intensity of a maximally piled up grid position

Definition at line 10 of file SimulationClusterDetector.cpp.

References entityHeight, entityWidth, height, and width.

7.42.2.2 SimulationClusterDetector::~SimulationClusterDetector ()

Definition at line 20 of file SimulationClusterDetector.cpp.

7.42.3 Member Function Documentation

7.42.3.1 unsigned int SimulationClusterDetector::computePileUpDegreeAtPosition (int x, int y) [private]

Compute the pile up degree at the given position.

Parameters

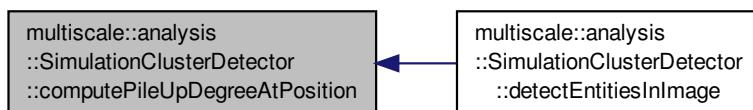
<i>x</i>	Coordinate for Ox axis
<i>y</i>	Coordinate for Oy axis

Definition at line 76 of file SimulationClusterDetector.cpp.

References entityHeight, multiscale::analysis::ClusterDetector::entityPileupDegree, entityWidth, and multiscale::analysis::Detector::image.

Referenced by detectEntitiesInImage().

Here is the caller graph for this function:



7.42.3.2 void SimulationClusterDetector::detectEntitiesInImage (vector< Entity > & entities) [override], [private], [virtual]

Detect the entities in the image.

Detect the entities in the image, compute their centre point and degree of pile up

Parameters

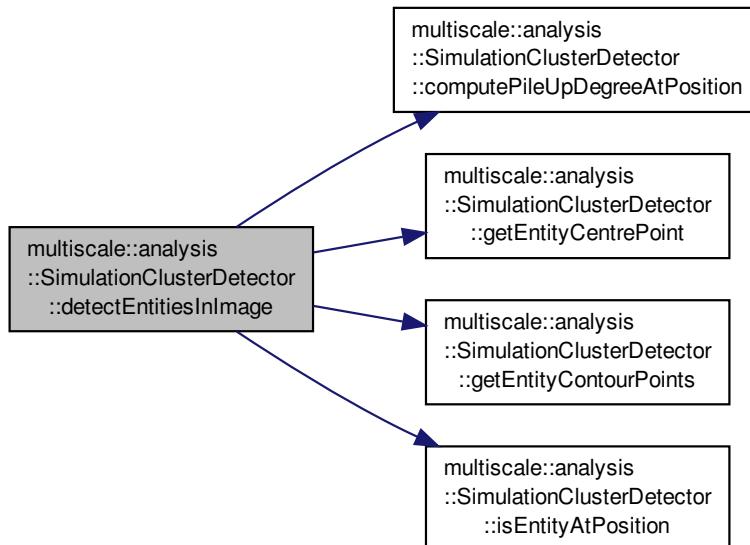
<code>entities</code>	Entities detected in the image
-----------------------	--------------------------------

Implements [multiscale::analysis::ClusterDetector](#).

Definition at line 33 of file `SimulationClusterDetector.cpp`.

References `computePileUpDegreeAtPosition()`, `entityHeight`, `entityWidth`, `getEntityCentrePoint()`, `getEntityContourPoints()`, `height`, `isEntityAtPosition()`, and `width`.

Here is the call graph for this function:



7.42.3.3 Point2f `SimulationClusterDetector::getEntityCentrePoint(int x, int y)` [private]

Get the point representing the centre of the entity.

Parameters

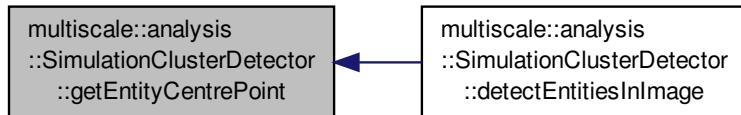
<code>x</code>	Ox coordinate
<code>y</code>	Oy coordinate

Definition at line 57 of file `SimulationClusterDetector.cpp`.

References `entityHeight`, and `entityWidth`.

Referenced by `detectEntitiesInImage()`.

Here is the caller graph for this function:



7.42.3.4 `vector< Point2f > SimulationClusterDetector::getEntityContourPoints (int x, int y) [private]`

Get the points representing the contour of the entity.

Parameters

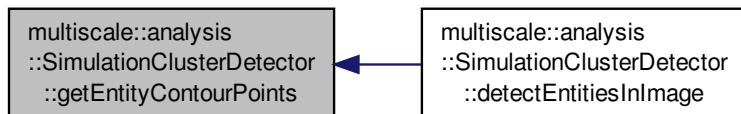
<code>x</code>	Ox coordinate
<code>y</code>	Oy coordinate

Definition at line 64 of file `SimulationClusterDetector.cpp`.

References `entityHeight`, and `entityWidth`.

Referenced by `detectEntitiesInImage()`.

Here is the caller graph for this function:



7.42.3.5 `void SimulationClusterDetector::initialiseDetectorSpecificImageDependentFields () [override], [private], [virtual]`

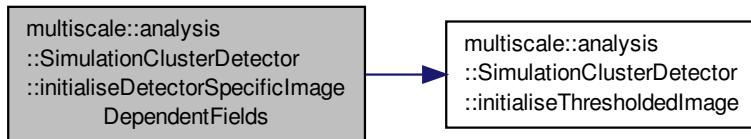
Initialise the image dependent values.

Implements [multiscale::analysis::Detector](#).

Definition at line 22 of file `SimulationClusterDetector.cpp`.

References `entityHeight`, `entityWidth`, `height`, `multiscale::analysis::Detector::image`, `initialiseThresholdedImage()`, and `width`.

Here is the call graph for this function:



7.42.3.6 void SimulationClusterDetector::initialiseThresholdedImage () [private]

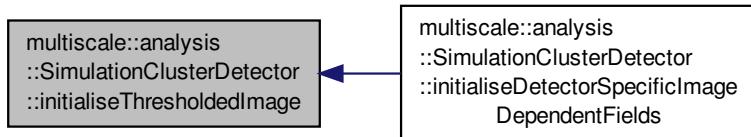
Initialise the thresholdedImage field.

Definition at line 29 of file SimulationClusterDetector.cpp.

References multiscale::analysis::Detector::image, THRESHOLD, THRESHOLD_MAX, and thresholdedImage.

Referenced by initialiseDetectorSpecificImageDependentFields().

Here is the caller graph for this function:



7.42.3.7 bool SimulationClusterDetector::isEntityAtPosition (int x, int y) [private]

Check if there is an entity in the image at the given position.

Parameters

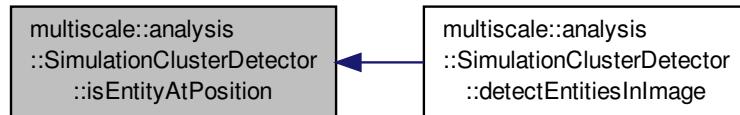
x	Coordinate for Ox axis
y	Coordinate for Oy axis

Definition at line 49 of file SimulationClusterDetector.cpp.

References ENTITY_THRESH, entityHeight, entityWidth, and thresholdedImage.

Referenced by detectEntitiesInImage().

Here is the caller graph for this function:



7.42.3.8 void SimulationClusterDetector::outputClusterCircularShape (Cluster & cluster, Scalar colour, Mat & image) [private]

Draw the best matching circular shape of the cluster on the image.

Parameters

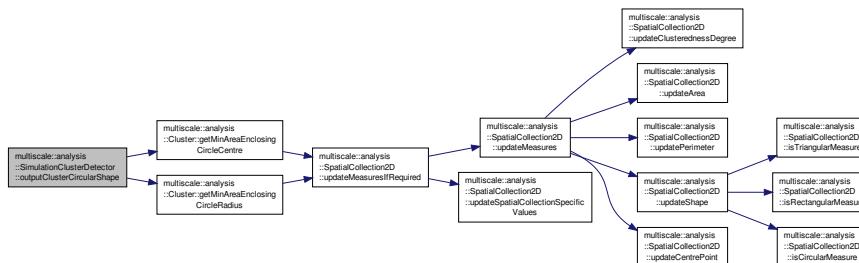
<i>cluster</i>	Cluster
<i>colour</i>	Colour associated to all entities in the cluster
<i>image</i>	The image on which to display the cluster related information

Definition at line 153 of file SimulationClusterDetector.cpp.

References DATAPoint_WIDTH, multiscale::analysis::Cluster::getMinAreaEnclosingCircleCentre(), and multiscale::analysis::Cluster::getMinAreaEnclosingCircleRadius().

Referenced by outputClusterShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.9 void **SimulationClusterDetector::outputClusterRectangularShape (Cluster & cluster, Scalar colour, Mat & image)**
[private]

Draw the best matching rectangular shape of the cluster on the image.

Parameters

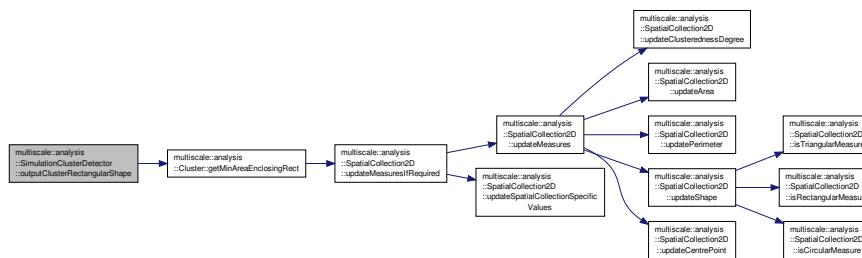
<i>cluster</i>	Cluster
<i>colour</i>	Colour associated to all entities in the cluster
<i>image</i>	The image on which to display the cluster related information

Definition at line 143 of file `SimulationClusterDetector.cpp`.

References `DATAPPOINT_WIDTH`, and `multiscale::analysis::Cluster::getMinAreaEnclosingRect()`.

Referenced by `outputClusterShape()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.10 void **SimulationClusterDetector::outputClusterShape (Cluster & cluster, Scalar colour, Mat & image)**
[private]

Draw the best matching shape (triangular, rectangular, circular) of the cluster on the image.

Parameters

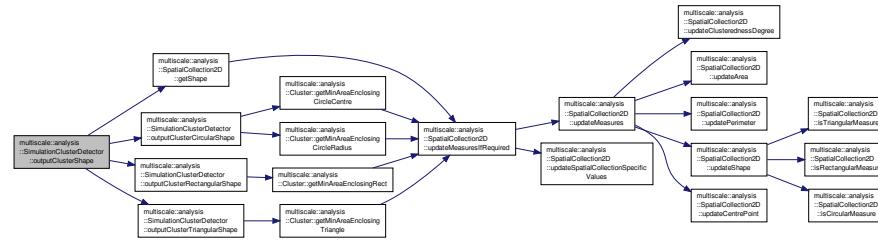
<i>cluster</i>	Cluster
<i>colour</i>	Colour associated to all entities in the cluster
<i>image</i>	The image on which to display the cluster related information

Definition at line 111 of file `SimulationClusterDetector.cpp`.

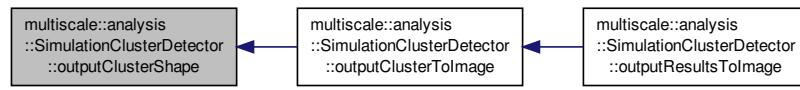
References `multiscale::analysis::Circle`, `ERR_UNDEFINED_SHAPE`, `multiscale::analysis::SpatialCollection2D::getShape()`, `outputClusterCircularShape()`, `outputClusterRectangularShape()`, `outputClusterTriangularShape()`, `multiscale::analysis::Rectangle`, and `multiscale::analysis::Triangle`.

Referenced by `outputClusterTolImage()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.11 void SimulationClusterDetector::outputClusterTolImage (Cluster & cluster, Scalar colour, Mat & image) [private]

Display cluster on the image.

Parameters

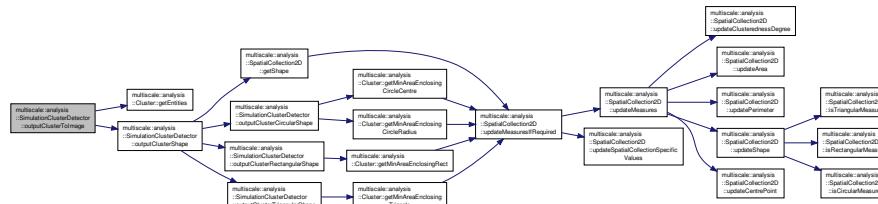
<i>cluster</i>	Cluster
<i>colour</i>	Colour associated to all entities in the cluster
<i>image</i>	The image on which to display the cluster related information

Definition at line 101 of file SimulationClusterDetector.cpp.

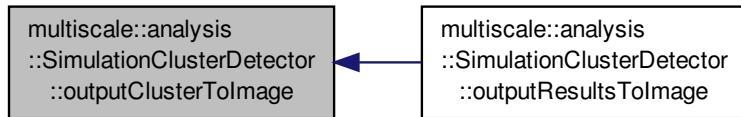
References DATAPOINT_THICKNESS, DATAPOINT_WIDTH, multiscale::analysis::Cluster::getEntities(), and outputClusterShape().

Referenced by outputResultsTolImage().

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.12 void SimulationClusterDetector::outputClusterTriangularShape (Cluster & cluster, Scalar colour, Mat & image) [private]

Draw the best matching triangular shape of the cluster on the image.

Parameters

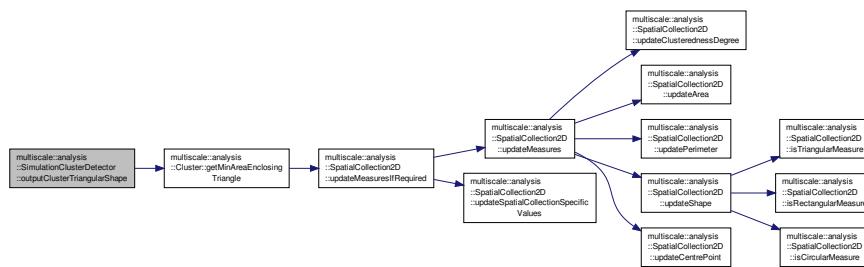
<i>cluster</i>	Cluster
<i>colour</i>	Colour associated to all entities in the cluster
<i>image</i>	The image on which to display the cluster related information

Definition at line 133 of file SimulationClusterDetector.cpp.

References DATAPOINT_WIDTH, and multiscale::analysis::Cluster::getMinAreaEnclosingTriangle().

Referenced by outputClusterShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.13 void SimulationClusterDetector::outputResultsToImage () [override], [private], [virtual]

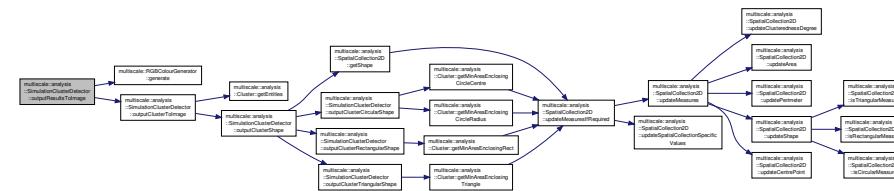
Display clusters on image.

Implements [multiscale::analysis::Detector](#).

Definition at line 85 of file [SimulationClusterDetector.cpp](#).

References [multiscale::analysis::ClusterDetector::clusters](#), [multiscale::RGBColourGenerator::generate\(\)](#), [multiscale::analysis::Detector::image](#), [outputClusterToImage\(\)](#), and [multiscale::analysis::Detector::outputImage](#).

Here is the call graph for this function:



7.42.4 Member Data Documentation

7.42.4.1 double multiscale::analysis::SimulationClusterDetector::entityHeight [private]

Height of an entity

Definition at line 35 of file [SimulationClusterDetector.hpp](#).

Referenced by [computePileUpDegreeAtPosition\(\)](#), [detectEntitiesInImage\(\)](#), [getEntityCentrePoint\(\)](#), [getEntityContourPoints\(\)](#), [initialiseDetectorSpecificImageDependentFields\(\)](#), [isEntityAtPosition\(\)](#), and [SimulationClusterDetector\(\)](#).

7.42.4.2 double multiscale::analysis::SimulationClusterDetector::entityWidth [private]

Width of an entity

Definition at line 36 of file [SimulationClusterDetector.hpp](#).

Referenced by [computePileUpDegreeAtPosition\(\)](#), [detectEntitiesInImage\(\)](#), [getEntityCentrePoint\(\)](#), [getEntityContourPoints\(\)](#), [initialiseDetectorSpecificImageDependentFields\(\)](#), [isEntityAtPosition\(\)](#), and [SimulationClusterDetector\(\)](#).

7.42.4.3 unsigned int multiscale::analysis::SimulationClusterDetector::height [private]

Height of the grid used in the simulation

Definition at line 32 of file [SimulationClusterDetector.hpp](#).

Referenced by [detectEntitiesInImage\(\)](#), [initialiseDetectorSpecificImageDependentFields\(\)](#), and [SimulationClusterDetector\(\)](#).

7.42.4.4 Mat multiscale::analysis::SimulationClusterDetector::thresholdedImage [private]

Thresholded version of the image

Definition at line 30 of file [SimulationClusterDetector.hpp](#).

Referenced by [initialiseThresholdedImage\(\)](#), and [isEntityAtPosition\(\)](#).

7.42.4.5 unsigned int multiscale::analysis::SimulationClusterDetector::width [private]

Width of the grid used in the simulation

Definition at line 33 of file SimulationClusterDetector.hpp.

Referenced by detectEntitiesInImage(), initialiseDetectorSpecificImageDependentFields(), and SimulationClusterDetector().

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

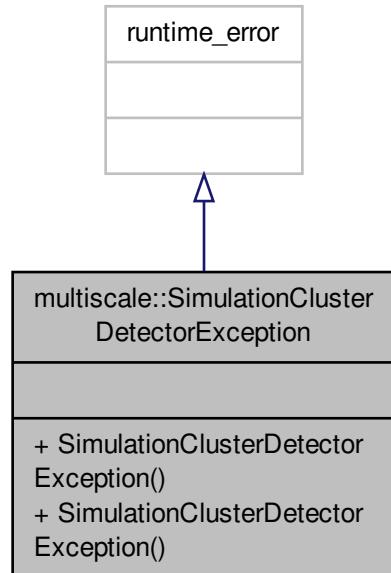
- include/multiscale/analysis/spatial/cluster/[SimulationClusterDetector.hpp](#)
- src/multiscale/analysis/spatial/cluster/[SimulationClusterDetector.cpp](#)

7.43 multiscale::SimulationClusterDetectorException Class Reference

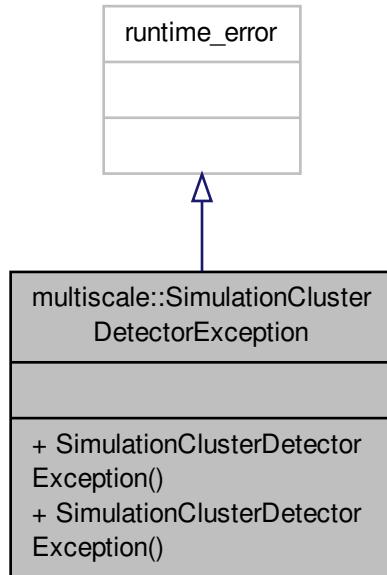
Exception class for the SimulationClusterDetector instances.

```
#include <SimulationClusterDetectorException.hpp>
```

Inheritance diagram for multiscale::SimulationClusterDetectorException:



Collaboration diagram for multiscale::SimulationClusterDetectorException:



Public Member Functions

- [SimulationClusterDetectorException \(const string &msg\)](#)
- [SimulationClusterDetectorException \(const char *msg\)](#)

7.43.1 Detailed Description

Exception class for the `SimulationClusterDetector` instances.

Definition at line 13 of file `SimulationClusterDetectorException.hpp`.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 `multiscale::SimulationClusterDetectorException::SimulationClusterDetectorException (const string & msg) [inline]`

Definition at line 17 of file `SimulationClusterDetectorException.hpp`.

7.43.2.2 `multiscale::SimulationClusterDetectorException::SimulationClusterDetectorException (const char * msg) [inline]`

Definition at line 18 of file `SimulationClusterDetectorException.hpp`.

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

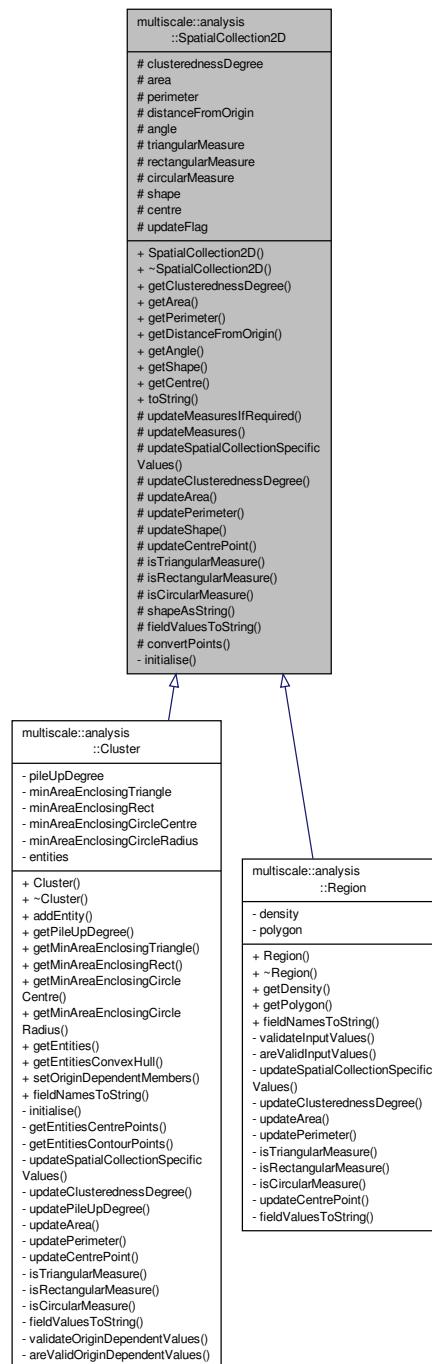
- [include/multiscale/exception/SimulationClusterDetectorException.hpp](#)

7.44 multiscale::analysis::SpatialCollection2D Class Reference

Class for representing a collection of objects in 2D space.

```
#include <SpatialCollection2D.hpp>
```

Inheritance diagram for multiscale::analysis::SpatialCollection2D:



Collaboration diagram for multiscale::analysis::SpatialCollection2D:

multiscale::analysis ::SpatialCollection2D
<pre># clusterednessDegree # area # perimeter # distanceFromOrigin # angle # triangularMeasure # rectangularMeasure # circularMeasure # shape # centre # updateFlag</pre>

<pre>+ SpatialCollection2D() + ~SpatialCollection2D() + getClusterednessDegree() + getArea() + getPerimeter() + getDistanceFromOrigin() + getAngle() + getShape() + getCentre() + toString() # updateMeasuresIfRequired() # updateMeasures() # updateSpatialCollectionSpecific Values() # updateClusterednessDegree() # updateArea() # updatePerimeter() # updateShape() # updateCentrePoint() # isTriangularMeasure() # isRectangularMeasure() # isCircularMeasure() # shapeAsString() # fieldValuesToString() # convertPoints() - initialise()</pre>
--

Public Member Functions

- [SpatialCollection2D \(\)](#)
- virtual [~SpatialCollection2D \(\)](#)
- double [getClusterednessDegree \(\)](#)

Get the clusteredness degree.

- double [getArea \(\)](#)

- `Get the area.`
 - `double getPerimeter ()`
Get the perimeter.
 - `double getDistanceFromOrigin ()`
Get the distance from the origin.
 - `double getAngle ()`
Get the angle.
 - `Shape2D getShape ()`
Get the shape best fitting the spatial collection.
 - `Point2f getCentre ()`
Get the point defining the centre of the entity.
 - `string toString ()`
Get the string representation of all field values.

Protected Member Functions

- `void updateMeasuresIfRequired ()`
Update the values of all measures if required.
- `void updateMeasures ()`
Update the values of all measures.
- `virtual void updateSpatialCollectionSpecificValues ()=0`
Update the values of all measures specific to the derived classes.
- `virtual void updateClusterednessDegree ()=0`
Update the value of the clusteredness degree.
- `virtual void updateArea ()=0`
Update the value of the area.
- `virtual void updatePerimeter ()=0`
Update the value of the perimeter.
- `void updateShape ()`
Update the shape of the cluster.
- `virtual void updateCentrePoint ()=0`
Update the point defining the centre of the cluster.
- `virtual double isTriangularMeasure ()=0`
Get the measure that the cluster has a triangular shape.
- `virtual double isRectangularMeasure ()=0`
Get the measure that the cluster has a rectangular shape.
- `virtual double isCircularMeasure ()=0`
Get the measure that the cluster has a circular shape.
- `string shapeAsString ()`
Return the shape of the cluster as a string.
- `virtual string fieldValuesToString ()=0`
Return the values of the fields as a string.
- `vector< Point2f > convertPoints (const vector< Point > &points)`
Convert the collection of points from type Point to type Point2f.

Protected Attributes

- double clusterednessDegree
- double area
- double perimeter
- double distanceFromOrigin
- double angle
- double triangularMeasure
- double rectangularMeasure
- double circularMeasure
- Shape2D shape
- Point2f centre
- bool updateFlag

Private Member Functions

- void initialise ()

Initialisation function for the class.

7.44.1 Detailed Description

Class for representing a collection of objects in 2D space.

Definition at line 29 of file SpatialCollection2D.hpp.

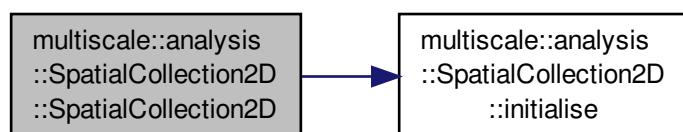
7.44.2 Constructor & Destructor Documentation

7.44.2.1 SpatialCollection2D::SpatialCollection2D ()

Definition at line 6 of file SpatialCollection2D.cpp.

References initialise().

Here is the call graph for this function:



7.44.2.2 SpatialCollection2D::~SpatialCollection2D () [virtual]

Definition at line 10 of file SpatialCollection2D.cpp.

7.44.3 Member Function Documentation

7.44.3.1 `vector< Point2f > SpatialCollection2D::convertPoints (const vector< Point > & points) [protected]`

Convert the collection of points from type Point to type Point2f.

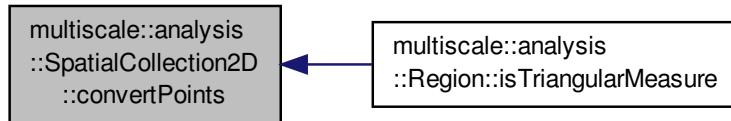
Parameters

<code>points</code>	Collection of points
---------------------	----------------------

Definition at line 119 of file SpatialCollection2D.cpp.

Referenced by multiscale::analysis::Region::isTriangularMeasure().

Here is the caller graph for this function:



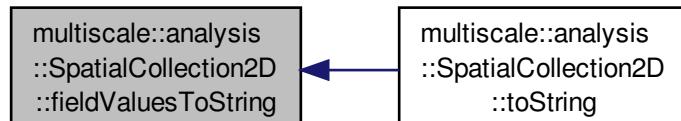
7.44.3.2 `virtual string multiscale::analysis::SpatialCollection2D::fieldValuesToString () [protected], [pure virtual]`

Return the values of the fields as a string.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by `toString()`.

Here is the caller graph for this function:



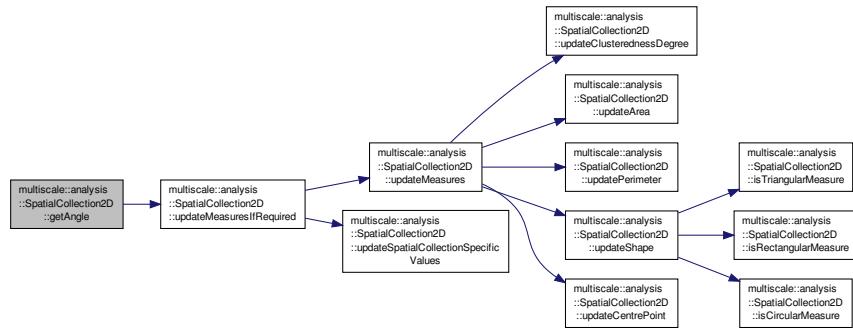
7.44.3.3 `double SpatialCollection2D::getAngle ()`

Get the angle.

Definition at line 36 of file SpatialCollection2D.cpp.

References `angle`, and `updateMeasuresIfRequired()`.

Here is the call graph for this function:



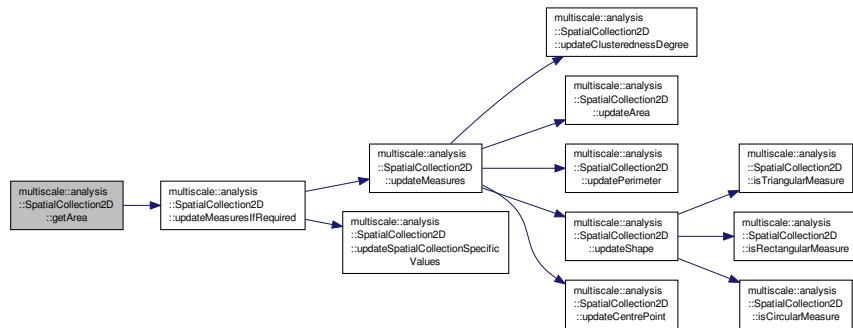
7.44.3.4 double SpatialCollection2D::getArea ()

Get the area.

Definition at line 18 of file SpatialCollection2D.cpp.

References area, and updateMeasuresIfRequired().

Here is the call graph for this function:



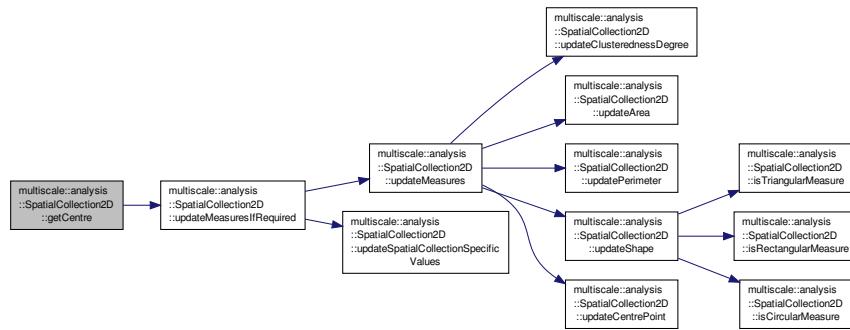
7.44.3.5 Point2f SpatialCollection2D::getCentre ()

Get the point defining the centre of the entity.

Definition at line 48 of file SpatialCollection2D.cpp.

References centre, and updateMeasuresIfRequired().

Here is the call graph for this function:



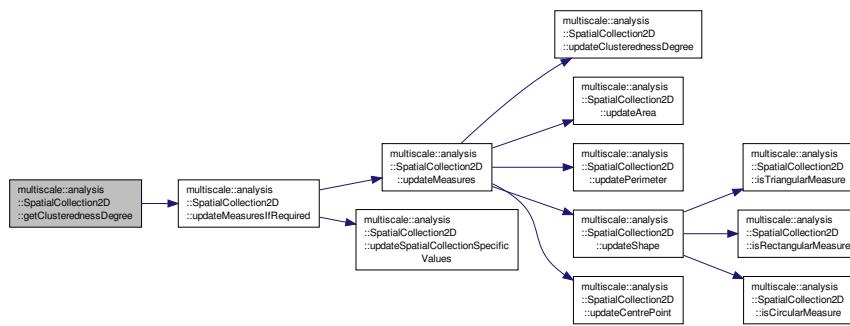
7.44.3.6 double SpatialCollection2D::getClusterednessDegree ()

Get the clusteredness degree.

Definition at line 12 of file SpatialCollection2D.cpp.

References clusterednessDegree, and updateMeasuresIfRequired().

Here is the call graph for this function:



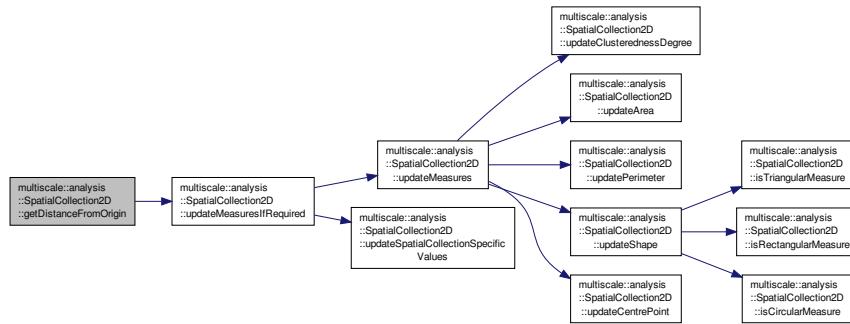
7.44.3.7 double SpatialCollection2D::getDistanceFromOrigin ()

Get the distance from the origin.

Definition at line 30 of file SpatialCollection2D.cpp.

References distanceFromOrigin, and updateMeasuresIfRequired().

Here is the call graph for this function:



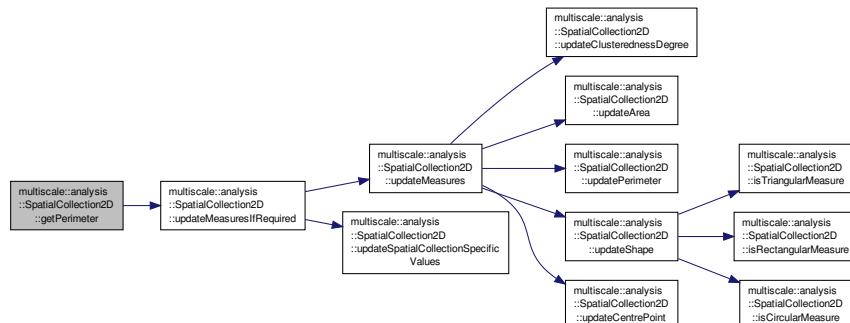
7.44.3.8 double SpatialCollection2D::getPerimeter ()

Get the perimeter.

Definition at line 24 of file SpatialCollection2D.cpp.

References perimeter, and updateMeasuresIfRequired().

Here is the call graph for this function:



7.44.3.9 Shape2D SpatialCollection2D::getShape ()

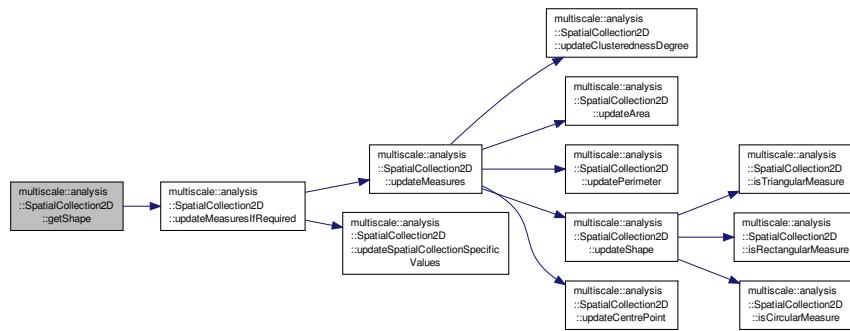
Get the shape best fitting the spatial collection.

Definition at line 42 of file SpatialCollection2D.cpp.

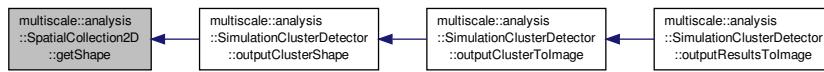
References shape, and updateMeasuresIfRequired().

Referenced by multiscale::analysis::SimulationClusterDetector::outputClusterShape().

Here is the call graph for this function:



Here is the caller graph for this function:



7.44.3.10 void SpatialCollection2D::initialise() [private]

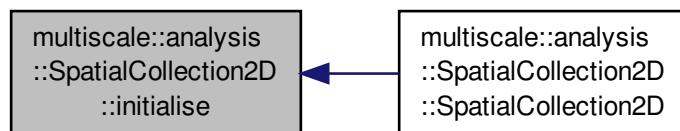
Initialisation function for the class.

Definition at line 129 of file SpatialCollection2D.cpp.

References area, circularMeasure, perimeter, rectangularMeasure, shape, triangularMeasure, multiscale::analysis::Undefined, and updateFlag.

Referenced by SpatialCollection2D().

Here is the caller graph for this function:



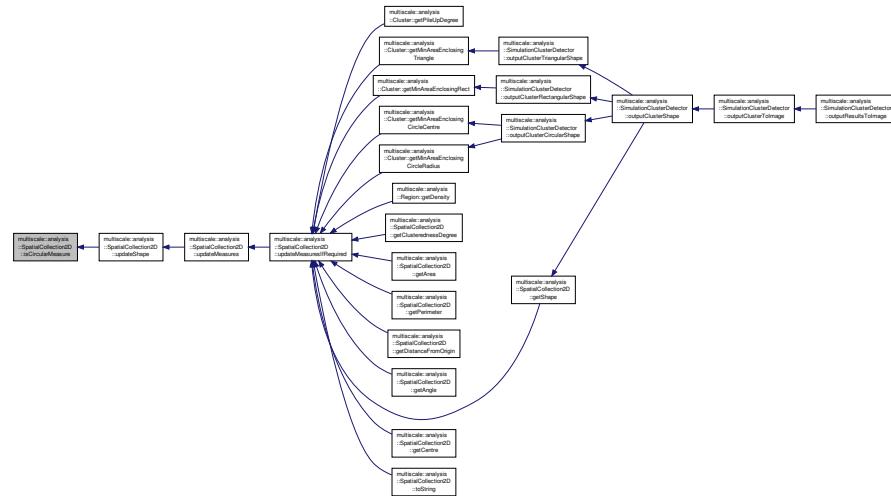
7.44.3.11 virtual double multiscale::analysis::SpatialCollection2D::isCircularMeasure() [protected], [pure virtual]

Get the measure that the cluster has a circular shape.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by `updateShape()`.

Here is the caller graph for this function:



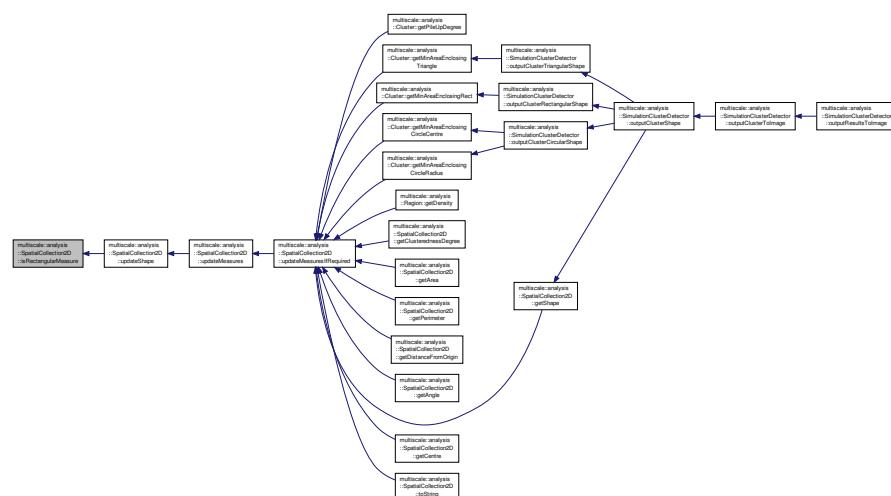
7.44.3.12 virtual double multiscale::analysis::SpatialCollection2D::isRectangularMeasure() [protected], [pure virtual]

Get the measure that the cluster has a rectangular shape.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by `updateShape()`.

Here is the caller graph for this function:



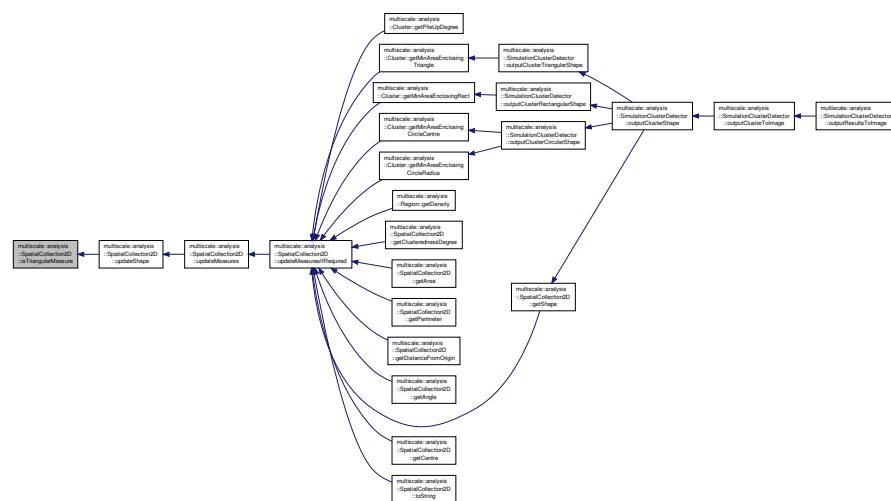
7.44.3.13 virtual double multiscale::analysis::SpatialCollection2D::isTriangularMeasure() [protected], [pure virtual]

Get the measure that the cluster has a triangular shape.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by [updateShape\(\)](#).

Here is the caller graph for this function:



7.44.3.14 string SpatialCollection2D::shapeAsString() [protected]

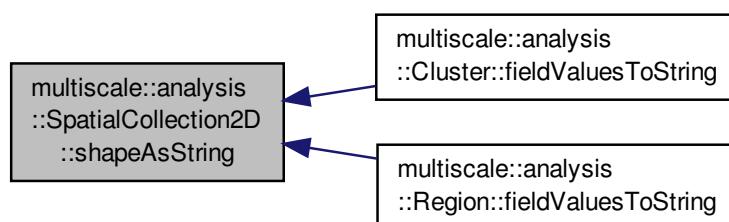
Return the shape of the cluster as a string.

Definition at line 97 of file [SpatialCollection2D.cpp](#).

References [multiscale::analysis::Circle](#), [multiscale::analysis::Rectangle](#), [shape](#), [STR_CIRCLE](#), [STR_RECTANGLE](#), [STR_TRIANGLE](#), [STR_UNDEFINED](#), [multiscale::analysis::Triangle](#), and [multiscale::analysis::Undefined](#).

Referenced by [multiscale::analysis::Cluster::fieldValuesToString\(\)](#), and [multiscale::analysis::Region::fieldValuesToString\(\)](#).

Here is the caller graph for this function:



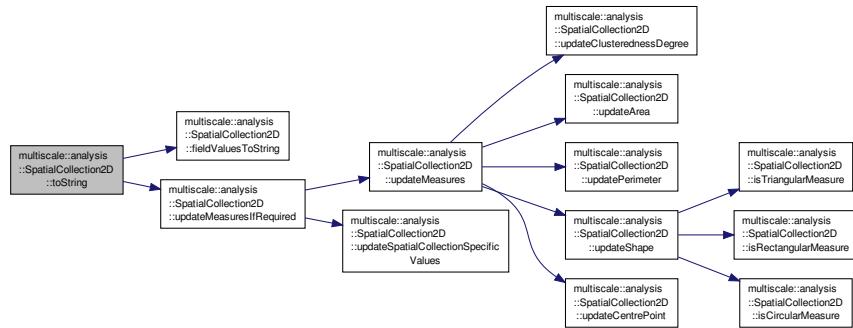
7.44.3.15 string SpatialCollection2D::toString ()

Get the string representation of all field values.

Definition at line 54 of file SpatialCollection2D.cpp.

References `fieldValuesToString()`, and `updateMeasuresIfRequired()`.

Here is the call graph for this function:



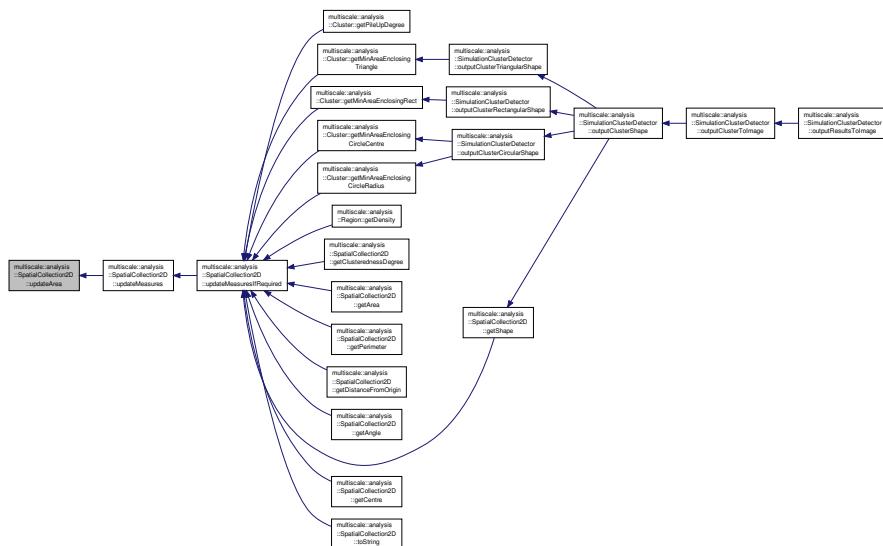
7.44.3.16 virtual void multiscale::analysis::SpatialCollection2D::updateArea () [protected], [pure virtual]

Update the value of the area.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by `updateMeasures()`.

Here is the caller graph for this function:



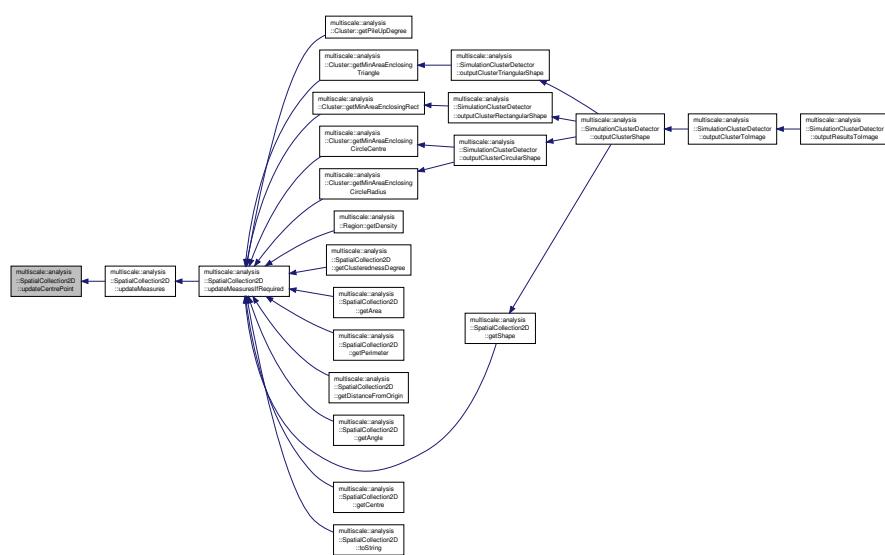
7.44.3.17 virtual void multiscale::analysis::SpatialCollection2D::updateCentrePoint() [protected], [pure virtual]

Update the point defining the centre of the cluster.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by updateMeasures().

Here is the caller graph for this function:



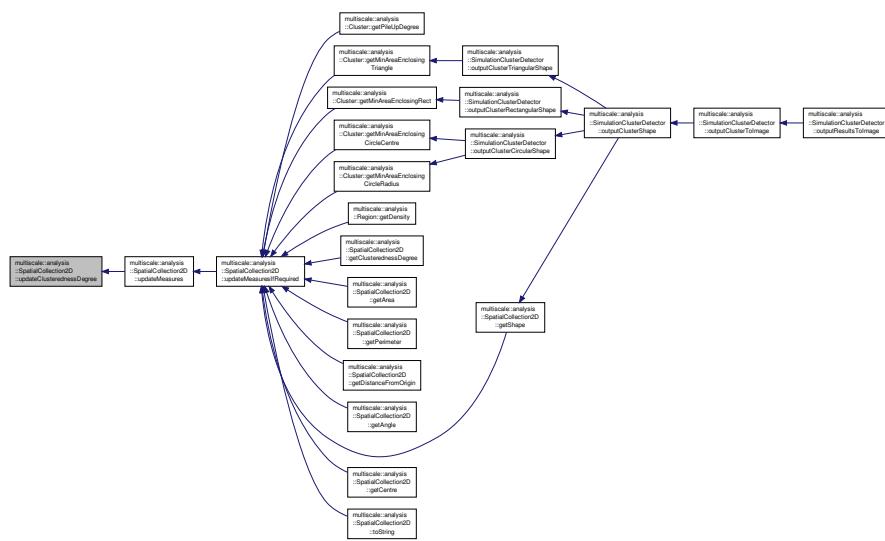
7.44.3.18 virtual void multiscale::analysis::SpatialCollection2D::updateClusterednessDegree() [protected], [pure virtual]

Update the value of the clusteredness degree.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by updateMeasures().

Here is the caller graph for this function:



7.44.3.19 void SpatialCollection2D::updateMeasures() [protected]

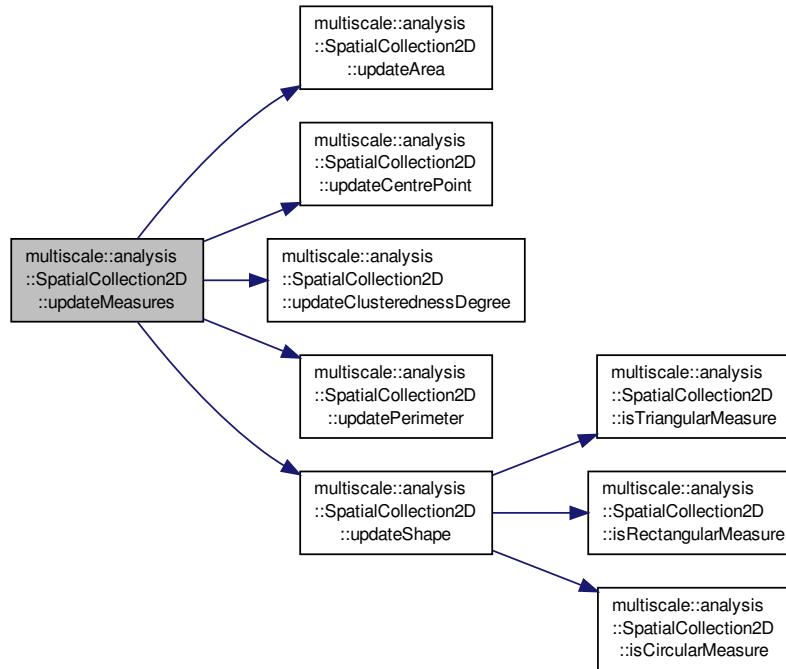
Update the values of all measures.

Definition at line 69 of file SpatialCollection2D.cpp.

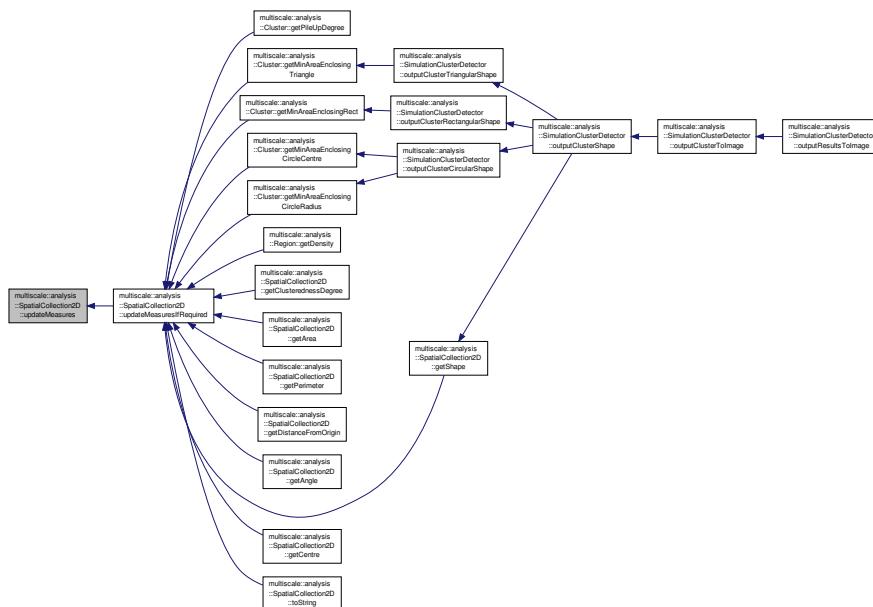
References updateArea(), updateCentrePoint(), updateClusterednessDegree(), updatePerimeter(), and updateShape().

Referenced by updateMeasuresIfRequired().

Here is the call graph for this function:



Here is the caller graph for this function:



7.44.3.20 void SpatialCollection2D::updateMeasuresIfRequired() [protected]

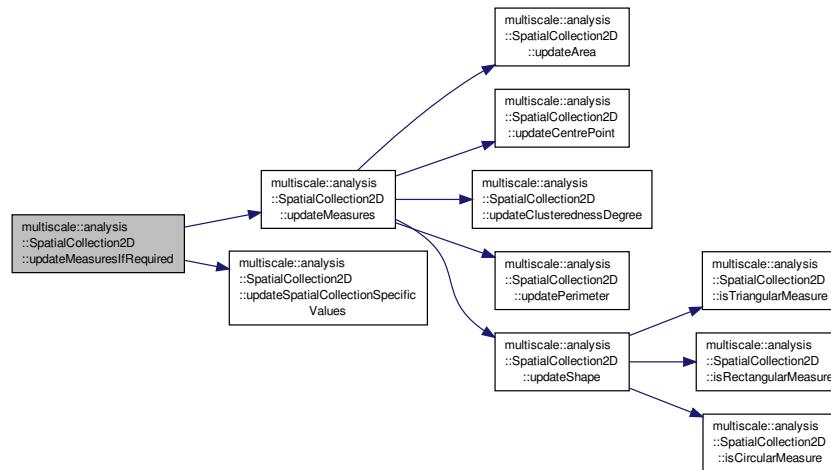
Update the values of all measures if required.

Definition at line 60 of file SpatialCollection2D.cpp.

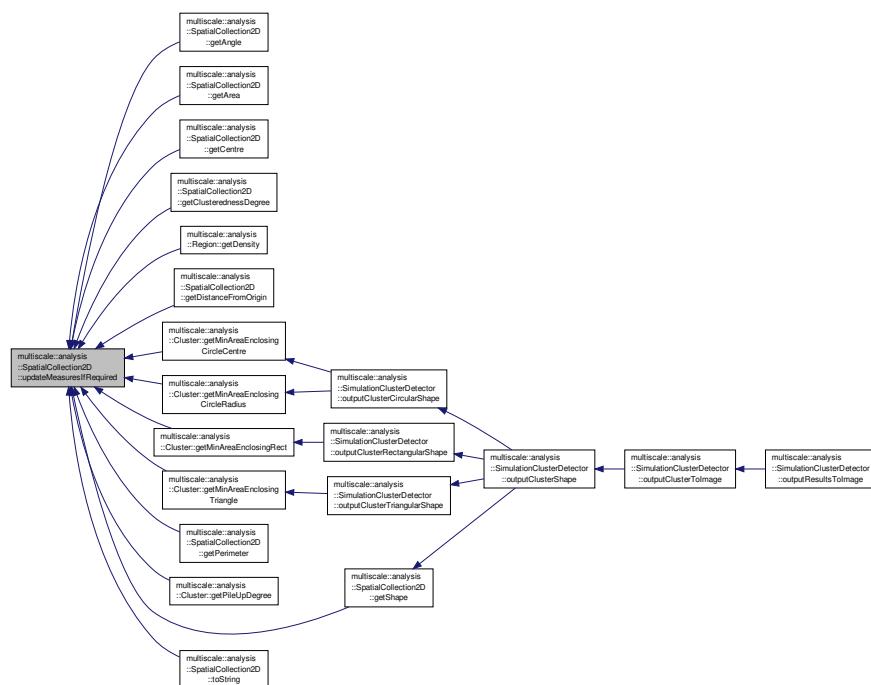
References updateFlag, updateMeasures(), and updateSpatialCollectionSpecificValues().

Referenced by getAngle(), getArea(), getCentre(), getClusterednessDegree(), multiscale::analysis::Region::getDensity(), getDistanceFromOrigin(), multiscale::analysis::Cluster::getMinAreaEnclosingCircleCentre(), multiscale::analysis::Cluster::getMinAreaEnclosingCircleRadius(), multiscale::analysis::Cluster::getMinAreaEnclosingRect(), multiscale::analysis::Cluster::getMinAreaEnclosingTriangle(), getPerimeter(), multiscale::analysis::Cluster::getPileUpDegree(), getShape(), and toString().

Here is the call graph for this function:



Here is the caller graph for this function:



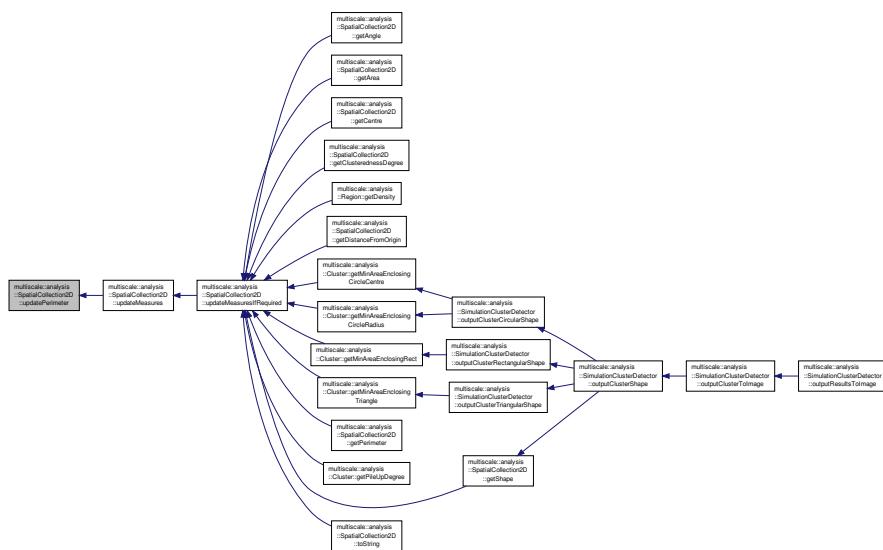
7.44.3.21 virtual void multiscale::analysis::SpatialCollection2D::updatePerimeter() [protected], [pure virtual]

Update the value of the perimeter.

Implemented in [multiscale::analysis::Cluster](#), and [multiscale::analysis::Region](#).

Referenced by updateMeasures().

Here is the caller graph for this function:



7.44.3.22 void SpatialCollection2D::updateShape() [protected]

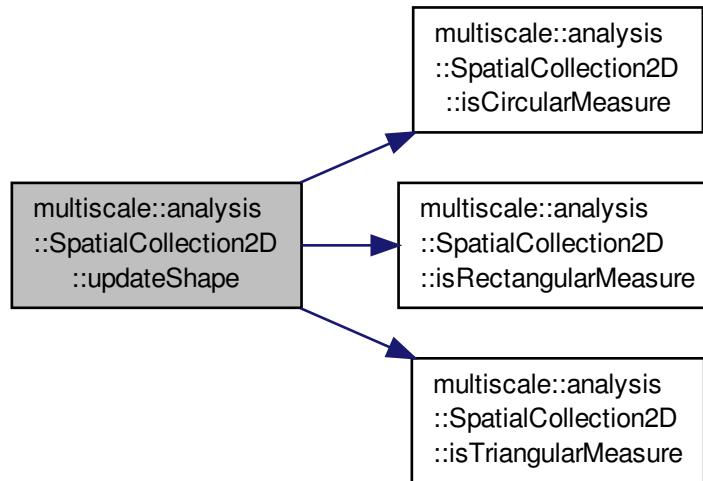
Update the shape of the cluster.

Definition at line 77 of file `SpatialCollection2D.cpp`.

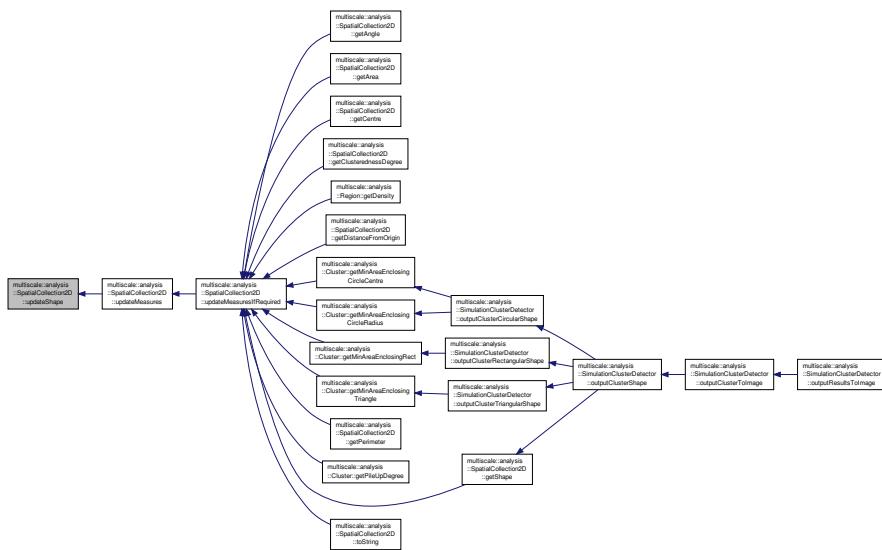
References `multiscale::analysis::Circle`, `circularMeasure`, `isCircularMeasure()`, `isRectangularMeasure()`, `isTriangularMeasure()`, `multiscale::analysis::Rectangle`, `rectangularMeasure`, `shape`, `multiscale::analysis::Triangle`, and `triangularMeasure`.

Referenced by updateMeasures().

Here is the call graph for this function:



Here is the caller graph for this function:



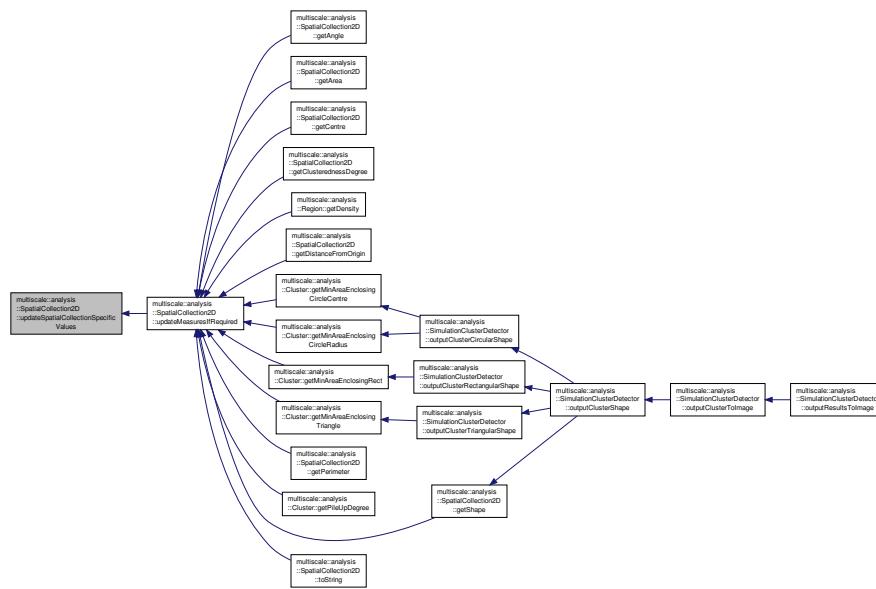
7.44.3.23 virtual void multiscale::analysis::SpatialCollection2D::updateSpatialCollectionSpecificValues () [protected], [pure virtual]

Update the values of all measures specific to the derived classes.

Implemented in [multiscale::analysis::Region](#), and [multiscale::analysis::Cluster](#).

Referenced by `updateMeasuresIfRequired()`.

Here is the caller graph for this function:



7.44.4 Member Data Documentation

7.44.4.1 double multiscale::analysis::SpatialCollection2D::angle [protected]

Angle of the region wrt the origin

Definition at line 39 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getAngle(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::setOriginDependentMembers().

7.44.4.2 double multiscale::analysis::SpatialCollection2D::area [protected]

Area of the spatial collection

Definition at line 35 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getArea(), initialise(), multiscale::analysis::Region::isCircularMeasure(), multiscale::analysis::Cluster::isCircularMeasure(), multiscale::analysis::Region::isRectangularMeasure(), multiscale::analysis::Cluster::isRectangularMeasure(), multiscale::analysis::Region::isTriangularMeasure(), multiscale::analysis::Cluster::isTriangularMeasure(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::updateArea().

7.44.4.3 Point2f multiscale::analysis::SpatialCollection2D::centre [protected]

Point defining the centre of the spatial collection

Definition at line 46 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getCentre(), multiscale::analysis::Cluster::updateCentrePoint(), and multiscale::analysis::Region::updateCentrePoint().

7.44.4.4 double multiscale::analysis::SpatialCollection2D::circularMeasure [protected]

Measure ([0, 1]) indicating that the shape of the spatial collection is circular

Definition at line 43 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), initialise(), and updateShape().

7.44.4.5 double multiscale::analysis::SpatialCollection2D::clusterednessDegree [protected]

Degree of clusteredness

Definition at line 33 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getClusterednessDegree(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::updateClusterednessDegree().

7.44.4.6 double multiscale::analysis::SpatialCollection2D::distanceFromOrigin [protected]

Distance from the origin

Definition at line 38 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getDistanceFromOrigin(), multiscale::analysis::Cluster::initialise(), multiscale::analysis::Region::Region(), and multiscale::analysis::Cluster::setOriginDependentMembers().

7.44.4.7 double multiscale::analysis::SpatialCollection2D::perimeter [protected]

Perimeter of the spatial collection

Definition at line 36 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), getPerimeter(), initialise(), multiscale::analysis::Cluster::updatePerimeter(), and multiscale::analysis::Region::updatePerimeter().

7.44.4.8 double multiscale::analysis::SpatialCollection2D::rectangularMeasure [protected]

Measure ([0, 1]) indicating that the shape of the spatial collection is rectangular

Definition at line 42 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), initialise(), and updateShape().

7.44.4.9 Shape2D multiscale::analysis::SpatialCollection2D::shape [protected]

Shape of the spatial collection

Definition at line 45 of file SpatialCollection2D.hpp.

Referenced by getShape(), initialise(), shapeAsString(), and updateShape().

7.44.4.10 double multiscale::analysis::SpatialCollection2D::triangularMeasure [protected]

Measure ([0, 1]) indicating that the shape of the spatial collection is triangular

Definition at line 41 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), initialise(), and updateShape().

7.44.4.11 bool multiscale::analysis::SpatialCollection2D::updateFlag [protected]

Flag indicating if the field values dependent on the

collection of entities need to be updated. This flag is used for lazy evaluation purposes, such that new field values are computed only when required

Definition at line 48 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::Cluster::addEntity(), multiscale::analysis::Cluster::initialise(), initialise(), and updateMeasuresIfRequired().

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

- include/multiscale/analysis/spatial/[SpatialCollection2D.hpp](#)

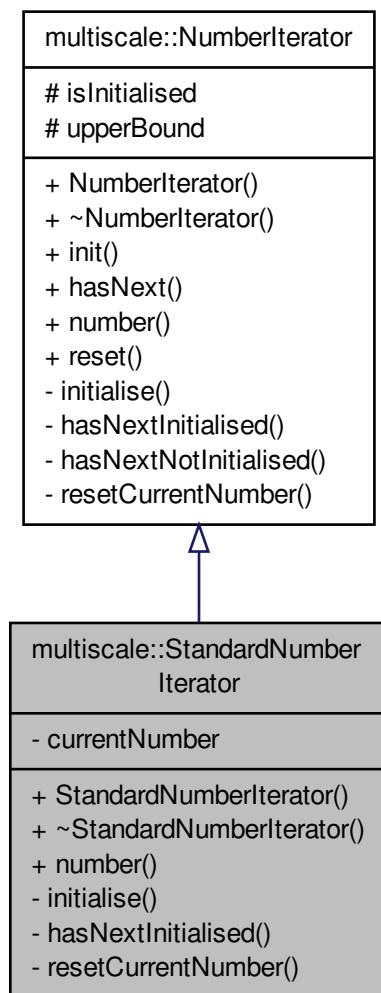
- src/multiscale/analysis/spatial/[SpatialCollection2D.cpp](#)

7.45 multiscale::StandardNumberIterator Class Reference

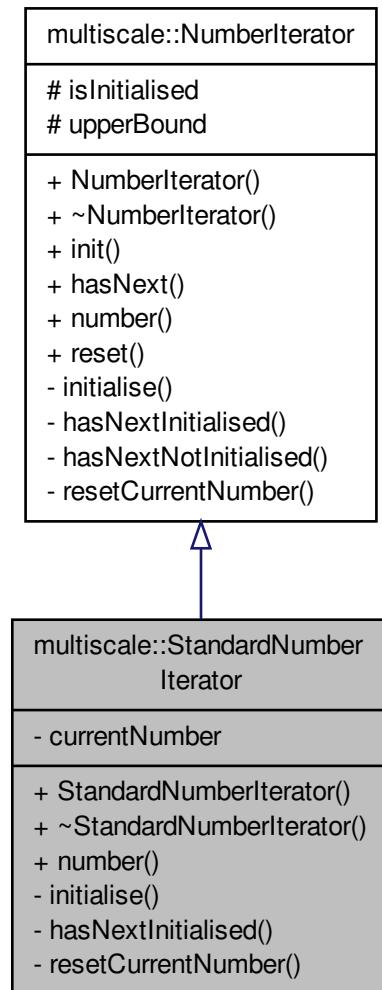
Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.

```
#include <StandardNumberIterator.hpp>
```

Inheritance diagram for multiscale::StandardNumberIterator:



Collaboration diagram for multiscale::StandardNumberIterator:



Public Member Functions

- `StandardNumberIterator (unsigned int upperBound)`
- `~StandardNumberIterator ()`
- `unsigned int number ()`

Get the number pointed by the iterator.

Private Member Functions

- `void initialise ()`
Initialise the value of the current number.
- `bool hasNextInitialised ()`
Check if there is a next number when in initialised state.
- `void resetCurrentNumber ()`
Reset the current number to the initial value.

Private Attributes

- unsigned int [currentNumber](#)

Additional Inherited Members

7.45.1 Detailed Description

Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.

Definition at line 10 of file [StandardNumberIterator.hpp](#).

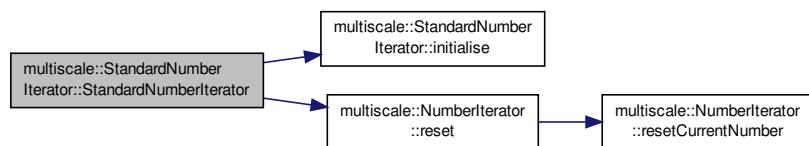
7.45.2 Constructor & Destructor Documentation

7.45.2.1 StandardNumberIterator::StandardNumberIterator (`unsigned int upperBound`)

Definition at line 6 of file [StandardNumberIterator.cpp](#).

References [initialise\(\)](#), and [multiscale::NumberIterator::reset\(\)](#).

Here is the call graph for this function:



7.45.2.2 StandardNumberIterator::~StandardNumberIterator ()

Definition at line 11 of file [StandardNumberIterator.cpp](#).

7.45.3 Member Function Documentation

7.45.3.1 bool StandardNumberIterator::hasNextInitialised () [private], [virtual]

Check if there is a next number when in initialised state.

Implements [multiscale::NumberIterator](#).

Definition at line 19 of file [StandardNumberIterator.cpp](#).

References [currentNumber](#), and [multiscale::NumberIterator::upperBound](#).

7.45.3.2 void StandardNumberIterator::initialise () [private], [virtual]

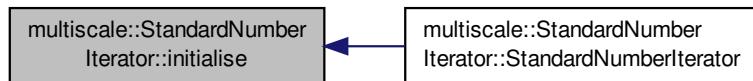
Initialise the value of the current number.

Implements [multiscale::NumberIterator](#).

Definition at line 17 of file [StandardNumberIterator.cpp](#).

Referenced by [StandardNumberIterator\(\)](#).

Here is the caller graph for this function:



7.45.3.3 unsigned int StandardNumberIterator::number() [virtual]

Get the number pointed by the iterator.

Implements [multiscale::NumberIterator](#).

Definition at line 13 of file StandardNumberIterator.cpp.

References currentNumber.

7.45.3.4 void StandardNumberIterator::resetCurrentNumber() [private], [virtual]

Reset the current number to the initial value.

Implements [multiscale::NumberIterator](#).

Definition at line 29 of file StandardNumberIterator.cpp.

References currentNumber.

7.45.4 Member Data Documentation

7.45.4.1 unsigned int multiscale::StandardNumberIterator::currentNumber [private]

The current number

Definition at line 14 of file StandardNumberIterator.hpp.

Referenced by [hasNextInitialised\(\)](#), [number\(\)](#), and [resetCurrentNumber\(\)](#).

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

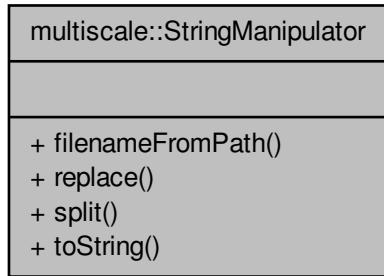
- include/multiscale/util/iterator/[StandardNumberIterator.hpp](#)
- src/multiscale/util/iterator/[StandardNumberIterator.cpp](#)

7.46 multiscale::StringManipulator Class Reference

Class for manipulating strings.

```
#include <StringManipulator.hpp>
```

Collaboration diagram for multiscale::StringManipulator:



Static Public Member Functions

- static string [filenameFromPath](#) (const string &filepath)
Obtain the file name from the given file path.
- static string [replace](#) (const string &initialString, const string &replaceWhat, const string &replaceTo)
Replace a substring of the given string with another string.
- static vector< string > [split](#) (const string &initialString, const string &delimiter)
Split the given string into a vector of strings considering the given delimiter.
- template<class T>
 static string [toString](#) (T variable)
Convert the variable to a string.

7.46.1 Detailed Description

Class for manipulating strings.

Definition at line 15 of file StringManipulator.hpp.

7.46.2 Member Function Documentation

7.46.2.1 string StringManipulator::filenameFromPath (const string & filepath) [static]

Obtain the file name from the given file path.

Parameters

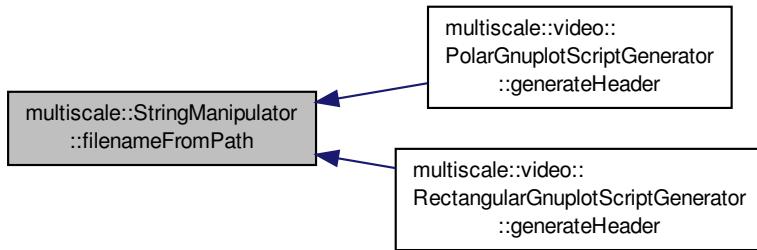
<i>filepath</i>	File path
-----------------	-----------

Definition at line 9 of file StringManipulator.cpp.

References DIR_SEPARATOR.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::generateHeader(), and multiscale::video::RectangularGnuplotScriptGenerator::generateHeader().

Here is the caller graph for this function:



7.46.2.2 string StringManipulator::replace (const string & *initialString*, const string & *replaceWhat*, const string & *replaceTo*) [static]

Replace a substring of the given string with another string.

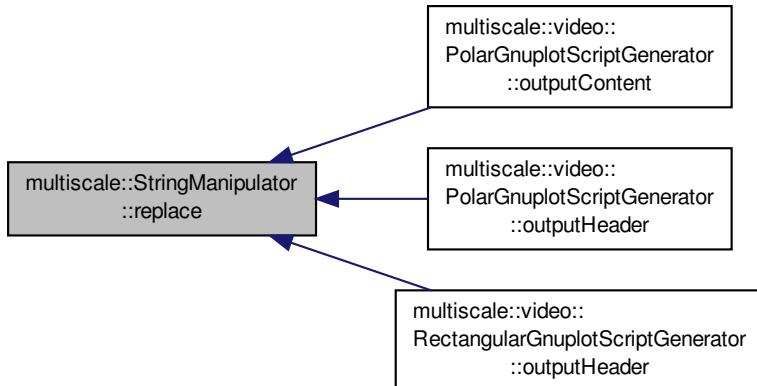
Parameters

<i>initialString</i>	Initial string
<i>replaceWhat</i>	Substring which will be replaced
<i>replaceTo</i>	String which will be inserted instead of the replaceWhat string

Definition at line 19 of file StringManipulator.cpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent(), multiscale::video::PolarGnuplotScriptGenerator::outputHeader(), and multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

Here is the caller graph for this function:



7.46.2.3 `vector< string > StringManipulator::split (const string & initialString, const string & delimiter) [static]`

Split the given string into a vector of strings considering the given delimiter.

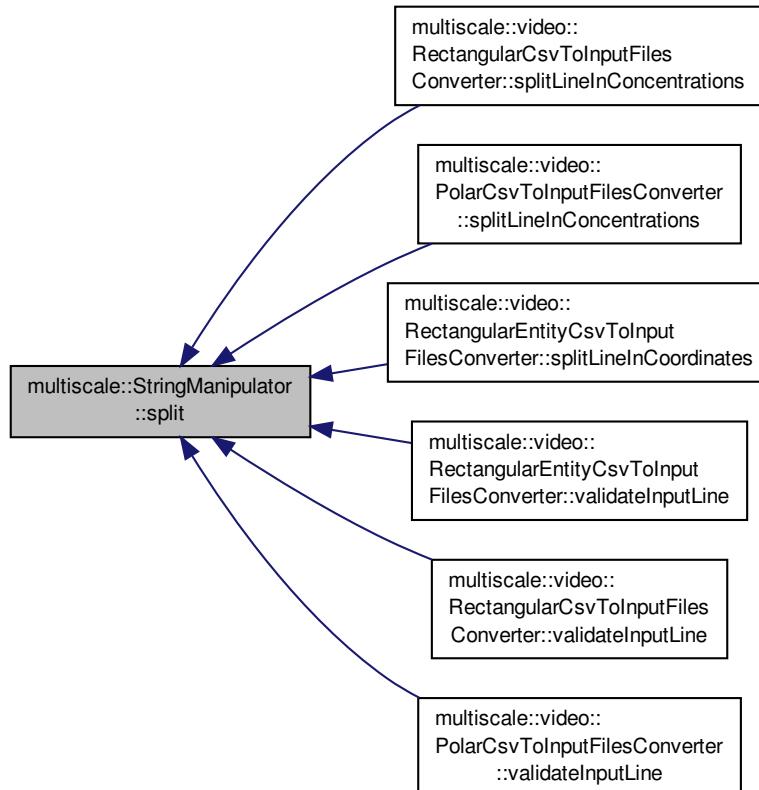
Parameters

<i>initialString</i>	Initial string
<i>delimiter</i>	Delimiter

Definition at line 32 of file StringManipulator.cpp.

Referenced by multiscale::video::RectangularCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::video::PolarCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::video::RectangularEntityCsvToInputFilesConverter::splitLineInCoordinates(), multiscale::video::RectangularEntityCsvToInputFilesConverter::validateInputLine(), multiscale::video::RectangularCsvToInputFilesConverter::validateInputLine(), and multiscale::video::PolarCsvToInputFilesConverter::validateInputLine().

Here is the caller graph for this function:



7.46.2.4 `template<class T > static string multiscale::StringManipulator::toString (T variable) [inline], [static]`

Convert the variable to a string.

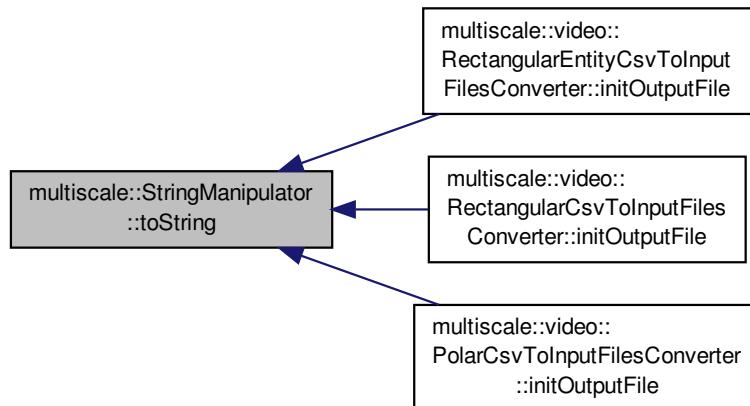
Parameters

<i>variable</i>	Variable
-----------------	----------

Definition at line 47 of file StringManipulator.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::initOutputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initOutputFile(), and multiscale::video::PolarCsvToInputFilesConverter::initOutputFile().

Here is the caller graph for this function:



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

- include/multiscale/util/[StringManipulator.hpp](#)
- src/multiscale/util/[StringManipulator.cpp](#)

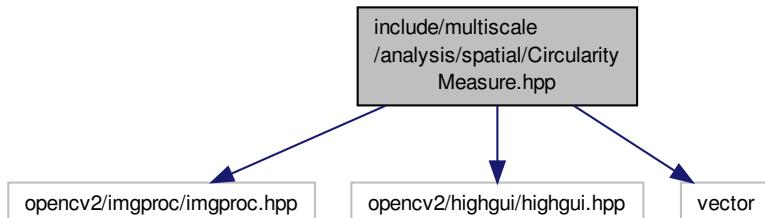
Chapter 8

File Documentation

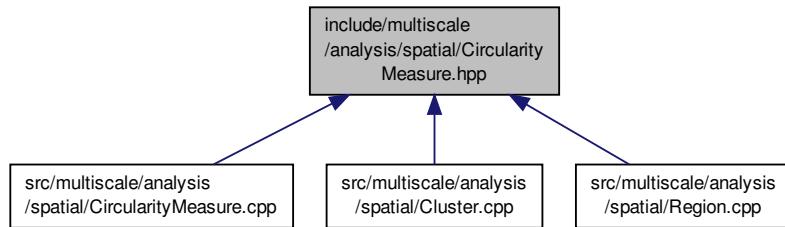
8.1 doc/mainpage.dox File Reference

8.2 include/multiscale/analysis/spatial/CircularityMeasure.hpp File Reference

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <vector>
Include dependency graph for CircularityMeasure.hpp:
```



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



Classes

- class [multiscale::analysis::CircularityMeasure](#)

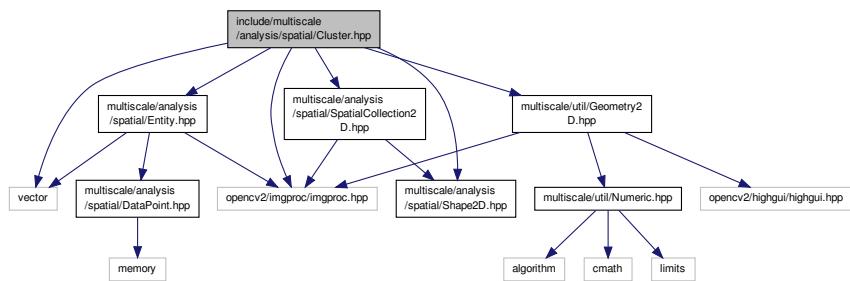
Class for computing the circularity measure for the given collection of points.

Namespaces

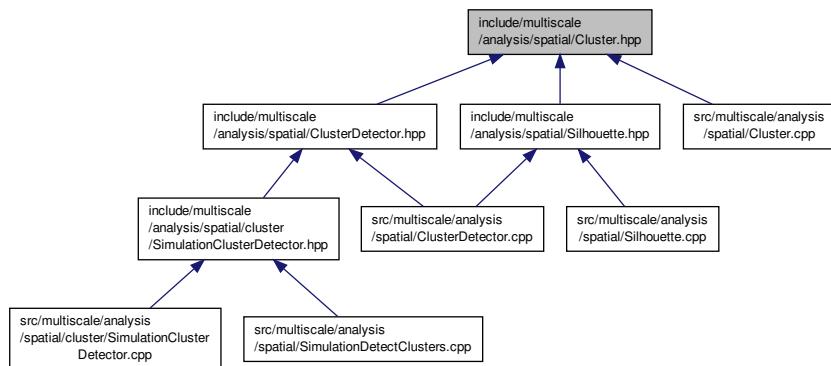
- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

8.3 include/multiscale/analysis/spatial/Cluster.hpp File Reference

```
#include "opencv2/imgproc/imgproc.hpp"
#include "multiscale/analysis/spatial/Entity.hpp"
#include "multiscale/analysis/spatial/Shape2D.hpp"
#include "multiscale/analysis/spatial/SpatialCollection2D.hpp"
#include "multiscale/util/Geometry2D.hpp"
#include <vector>
Include dependency graph for Cluster.hpp:
```



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



Classes

- class [multiscale::analysis::Cluster](#)

Class for representing a cluster of entities in an image.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- `#define ERR_ORIGIN_DEPENDENT_VALUES "The origin dependent values are invalid (i.e. negative)."`

8.3.1 Macro Definition Documentation

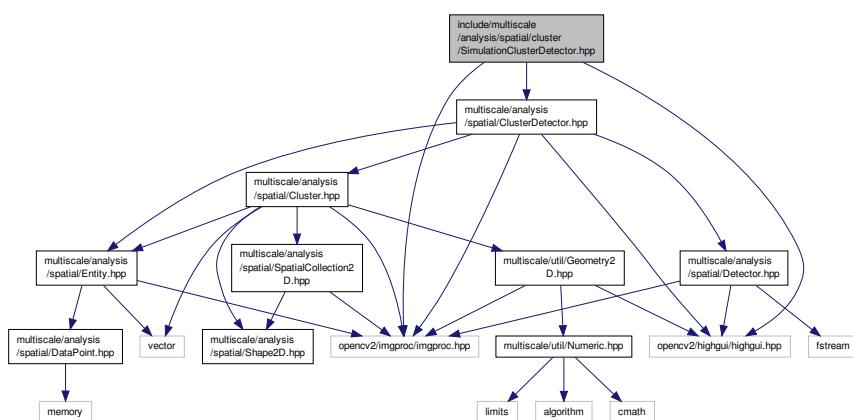
8.3.1.1 `#define ERR_ORIGIN_DEPENDENT_VALUES "The origin dependent values are invalid (i.e. negative)."`

Definition at line 15 of file Cluster.hpp.

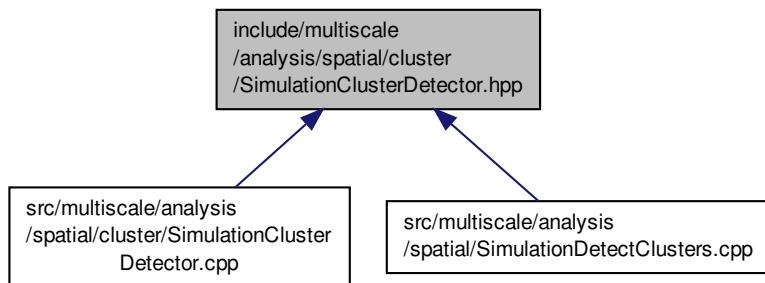
Referenced by `multiscale::analysis::Cluster::validateOriginDependentValues()`.

8.4 include/multiscale/analysis/spatial/cluster/SimulationClusterDetector.hpp File Reference

```
#include "multiscale/analysis/spatial/ClusterDetector.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
Include dependency graph for SimulationClusterDetector.hpp:
```



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



Classes

- class [multiscale::analysis::SimulationClusterDetector](#)
Class for detecting clusters in 2D images obtained from simulations.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define [THRESHOLD](#) 1
- #define [THRESHOLD_MAX](#) 255
- #define [ENTITY_THRESH](#) 200
- #define [DATAPOINT_WIDTH](#) 10
- #define [DATAPOINT_THICKNESS](#) -1

8.4.1 Macro Definition Documentation

8.4.1.1 #define DATAPOINT_THICKNESS -1

Definition at line 18 of file `SimulationClusterDetector.hpp`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputClusterToImage()`.

8.4.1.2 #define DATAPOINT_WIDTH 10

Definition at line 17 of file `SimulationClusterDetector.hpp`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputClusterCircularShape()`, `multiscale::analysis::SimulationClusterDetector::outputClusterRectangularShape()`, `multiscale::analysis::SimulationClusterDetector::outputClusterToImage()`, and `multiscale::analysis::SimulationClusterDetector::outputClusterTriangularShape()`.

8.4.1.3 #define ENTITY_THRESH 200

Definition at line 15 of file SimulationClusterDetector.hpp.

Referenced by multiscale::analysis::SimulationClusterDetector::isEntityAtPosition().

8.4.1.4 #define THRESHOLD 1

Definition at line 12 of file SimulationClusterDetector.hpp.

Referenced by multiscale::analysis::SimulationClusterDetector::initialiseThresholdedImage().

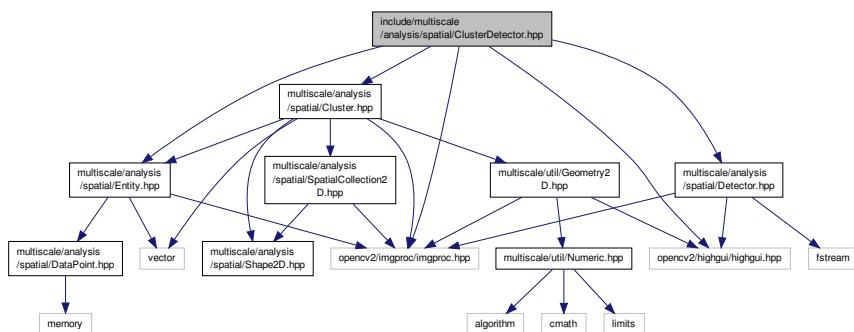
8.4.1.5 #define THRESHOLD_MAX 255

Definition at line 13 of file SimulationClusterDetector.hpp.

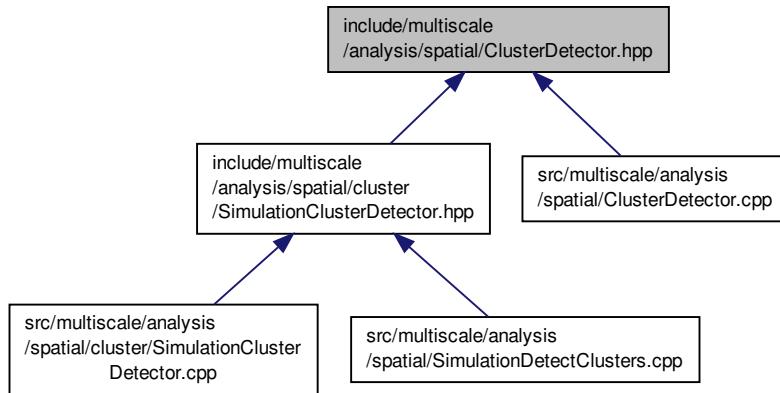
Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars(), multiscale::analysis::SimulationClusterDetector::initialiseThresholdedImage(), multiscale::analysis::RegionDetector::regionArea(), multiscale::analysis::RegionDetector::regionClusterednessDegree(), and multiscale::analysis::RegionDetector::thresholdImage().

8.5 include/multiscale/analysis/spatial/ClusterDetector.hpp File Reference

```
#include "multiscale/analysis/spatial/Cluster.hpp"
#include "multiscale/analysis/spatial/Detector.hpp"
#include "multiscale/analysis/spatial/Entity.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
Include dependency graph for ClusterDetector.hpp:
```



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



Classes

- class [multiscale::analysis::ClusterDetector](#)
Class for detecting clusters in 2D images.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define OUTPUT_CLUSTEREDNESS "Clusteredness index: "
- #define OUTPUT_PILE_UP "Average pile up degree: "
- #define TRACKBAR_EPS "Eps (Multiplied by 10)"
- #define TRACKBAR_MINPOINTS "Minimum number of points"
- #define MIN_POINTS_MIN 0
- #define MIN_POINTS_MAX 100
- #define EPS_MIN 0
- #define EPS_MAX 10000
- #define EPS_REAL_MIN 0
- #define EPS_REAL_MAX 1000

8.5.1 Macro Definition Documentation

8.5.1.1 #define EPS_MAX 10000

Definition at line 24 of file ClusterDetector.hpp.

Referenced by [multiscale::analysis::ClusterDetector::convertEpsValue\(\)](#), [multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars\(\)](#), and [multiscale::analysis::ClusterDetector::setEps\(\)](#).

8.5.1.2 #define EPS_MIN 0

Definition at line 23 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::convertEpsValue(), and multiscale::analysis::ClusterDetector::setEps().

8.5.1.3 #define EPS_REAL_MAX 1000

Definition at line 26 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::convertEpsValue(), and multiscale::analysis::ClusterDetector::setEps().

8.5.1.4 #define EPS_REAL_MIN 0

Definition at line 25 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::convertEpsValue(), and multiscale::analysis::ClusterDetector::setEps().

8.5.1.5 #define MIN_POINTS_MAX 100

Definition at line 21 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars().

8.5.1.6 #define MIN_POINTS_MIN 0

Definition at line 20 of file ClusterDetector.hpp.

8.5.1.7 #define OUTPUT_CLUSTEREDNESS "Clusteredness index: "

Definition at line 14 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::outputAveragedMeasuresToCsvFile(), and multiscale::analysis::ClusterDetector::outputResultsToCsvFile().

8.5.1.8 #define OUTPUT_PILE_UP "Average pile up degree: "

Definition at line 15 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::outputResultsToCsvFile().

8.5.1.9 #define TRACKBAR_EPS "Eps (Multiplied by 10)"

Definition at line 17 of file ClusterDetector.hpp.

Referenced by multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars().

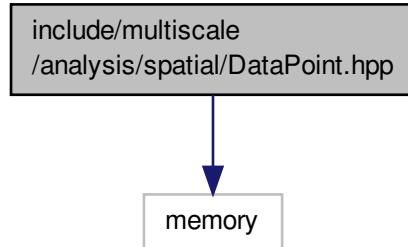
8.5.1.10 #define TRACKBAR_MINPOINTS "Minimum number of points"

Definition at line 18 of file ClusterDetector.hpp.

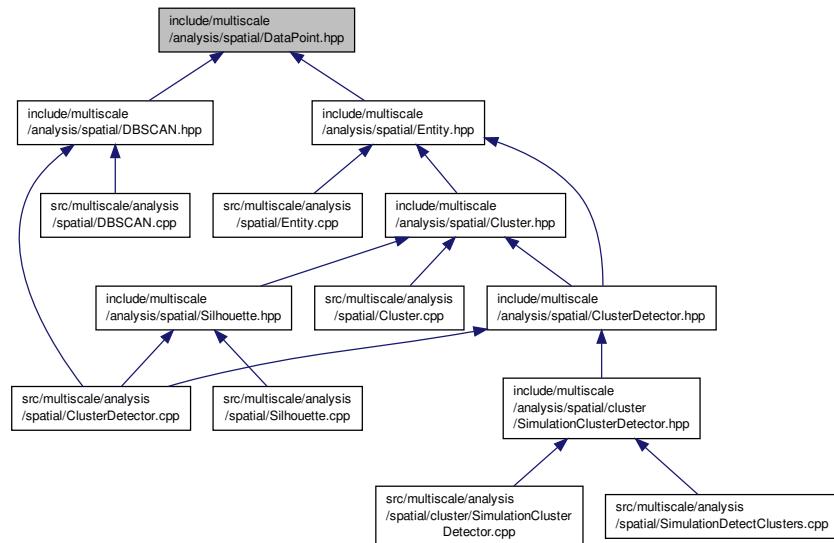
Referenced by multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars().

8.6 include/multiscale/analysis/spatial/DataPoint.hpp File Reference

```
#include <memory>
Include dependency graph for DataPoint.hpp:
```



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



Classes

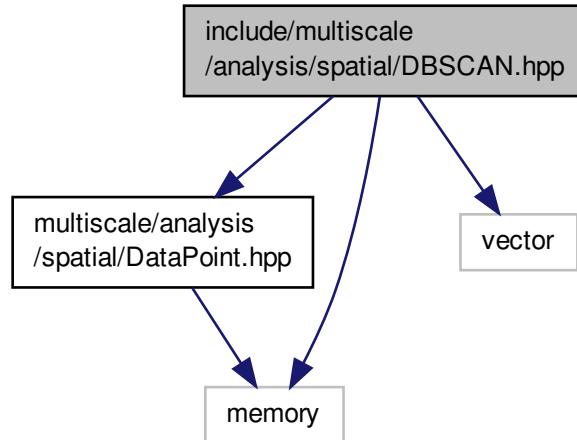
- class [multiscale::analysis::DataPoint](#)

Namespaces

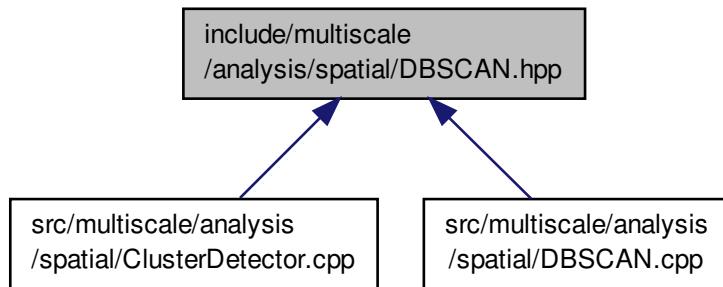
- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

8.7 include/multiscale/analysis/spatial/DBSCAN.hpp File Reference

```
#include "multiscale/analysis/spatial/DataPoint.hpp"
#include <memory>
#include <vector>
Include dependency graph for DBSCAN.hpp:
```



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



Classes

- class [multiscale::analysis::DBSCAN](#)

Namespaces

- namespace [multiscale](#)

- namespace `multiscale::analysis`

Macros

- `#define CLUSTERING_UNCLASSIFIED -2`
- `#define CLUSTERING_BORDER -1`
- `#define CLUSTERING_NOISE 0`

8.7.1 Macro Definition Documentation

8.7.1.1 `#define CLUSTERING_BORDER -1`

Definition at line 12 of file DBSCAN.hpp.

Referenced by `multiscale::analysis::DBSCAN::assignBorderNodesToClusters()`, and `multiscale::analysis::DBSCAN::labelUnclassifiedAndNoiseAsBorder()`.

8.7.1.2 `#define CLUSTERING_NOISE 0`

Definition at line 13 of file DBSCAN.hpp.

Referenced by `multiscale::analysis::DBSCAN::expandCoreCluster()`, and `multiscale::analysis::DBSCAN::labelUnclassifiedAndNoiseAsBorder()`.

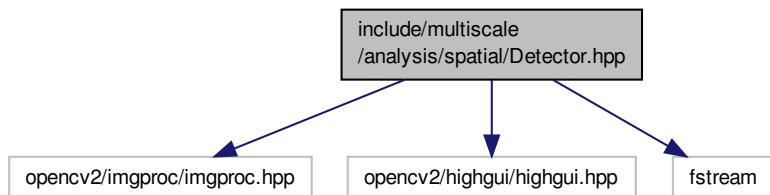
8.7.1.3 `#define CLUSTERING_UNCLASSIFIED -2`

Definition at line 11 of file DBSCAN.hpp.

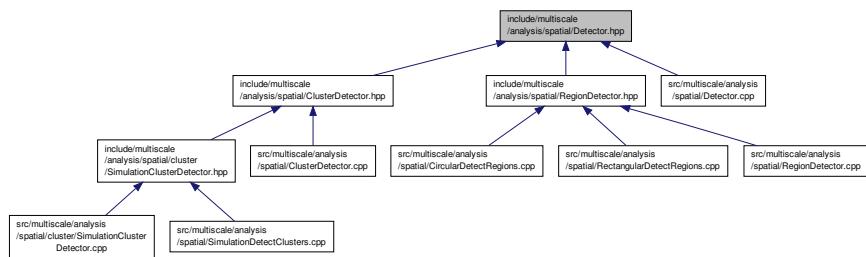
Referenced by `multiscale::analysis::DBSCAN::addUnclassifiedNodesToSeedsList()`, `multiscale::analysis::ClusterDetector::detectAndAnalyseClusters()`, `multiscale::analysis::DBSCAN::labelUnclassifiedAndNoiseAsBorder()`, and `multiscale::analysis::DBSCAN::runAlgorithm()`.

8.8 include/multiscale/analysis/spatial/Detector.hpp File Reference

```
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
#include <fstream>
Include dependency graph for Detector.hpp:
```



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



Classes

- class [multiscale::analysis::Detector](#)

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define [ERR_OUTPUT_WITHOUT_DETECT](#) "Unable to output results if the detect method was not called previously."
- #define [ERR_OUTPUT_FILE](#) "Unable to create output file."
- #define [ERR_INVALID_IMAGE](#) "The input image is invalid."
- #define [OUTPUT_EXTENSION](#) ".out"
- #define [IMG_EXTENSION](#) ".png"
- #define [WIN_OUTPUT_IMAGE](#) "Output image"
- #define [KEY_ESC](#) 27
- #define [KEY_SAVE](#) 115

8.8.1 Macro Definition Documentation

8.8.1.1 #define [ERR_INVALID_IMAGE](#) "The input image is invalid."

Definition at line 14 of file Detector.hpp.

Referenced by multiscale::analysis::Detector::detect().

8.8.1.2 #define [ERR_OUTPUT_FILE](#) "Unable to create output file."

Definition at line 13 of file Detector.hpp.

Referenced by multiscale::analysis::Detector::outputResultsToCsvFile().

8.8.1.3 #define [ERR_OUTPUT_WITHOUT_DETECT](#) "Unable to output results if the detect method was not called previously."

Definition at line 12 of file Detector.hpp.

Referenced by multiscale::analysis::Detector::printOutputErrorMessage().

8.8.1.4 #define IMG_EXTENSION ".png"

Definition at line 17 of file Detector.hpp.

Referenced by multiscale::analysis::Detector::storeOutputImageOnDisk().

8.8.1.5 #define KEY_ESC 27

Definition at line 21 of file Detector.hpp.

Referenced by multiscale::analysis::Detector::detectInDebugMode().

8.8.1.6 #define KEY_SAVE 115

Definition at line 22 of file Detector.hpp.

8.8.1.7 #define OUTPUT_EXTENSION ".out"

Definition at line 16 of file Detector.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::initOutputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initOutputFile(), multiscale::video::PolarCsvToInputFilesConverter::initOutputFile(), and multiscale::analysis::Detector::outputResultsToCsvFile().

8.8.1.8 #define WIN_OUTPUT_IMAGE "Output image"

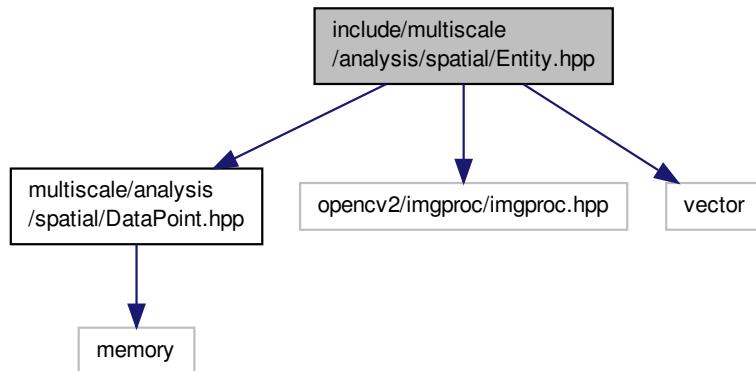
Definition at line 19 of file Detector.hpp.

Referenced by multiscale::analysis::ClusterDetector::createDetectorSpecificTrackbars(), multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars(), multiscale::analysis::Detector::createTrackbarsWindow(), and multiscale::analysis::Detector::displayResultsInWindow().

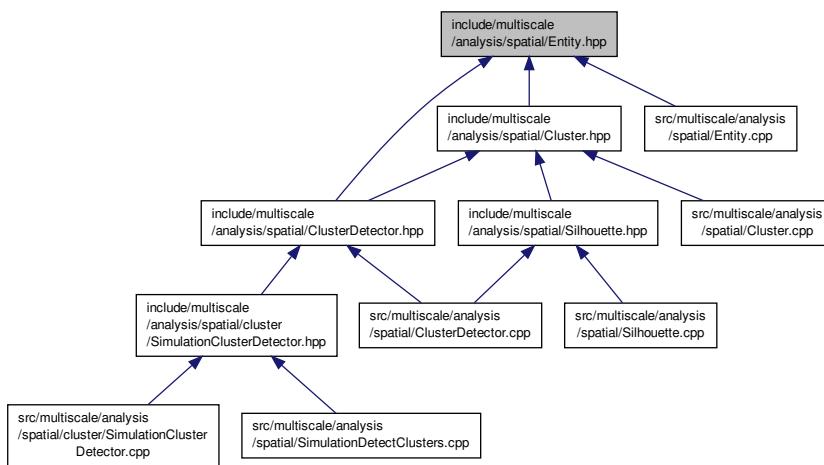
8.9 include/multiscale/analysis/spatial/Entity.hpp File Reference

```
#include "multiscale/analysis/spatial/DataPoint.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <vector>
```

Include dependency graph for Entity.hpp:



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



Classes

- class [multiscale::analysis::Entity](#)

Class for representing an entity in an image (e.g. cell, organism etc.)

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- `#define ERR_INPUT "Invalid input parameters were provided to the constructor."`

- `#define ERR_DISTANCE "The distance to an object of a different type cannot be computed."`
- `#define OUTPUT_SEPARATOR ","`

8.9.1 Macro Definition Documentation

8.9.1.1 `#define ERR_DISTANCE "The distance to an object of a different type cannot be computed."`

Definition at line 14 of file Entity.hpp.

8.9.1.2 `#define ERR_INPUT "Invalid input parameters were provided to the constructor."`

Definition at line 13 of file Entity.hpp.

Referenced by multiscale::analysis::Region::validateInputValues(), and multiscale::analysis::Entity::validateInputValues().

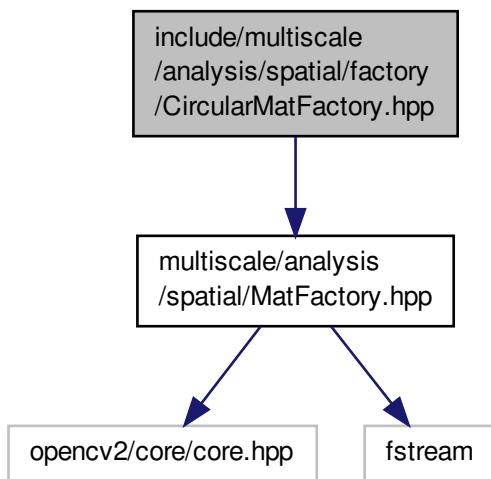
8.9.1.3 `#define OUTPUT_SEPARATOR ","`

Definition at line 16 of file Entity.hpp.

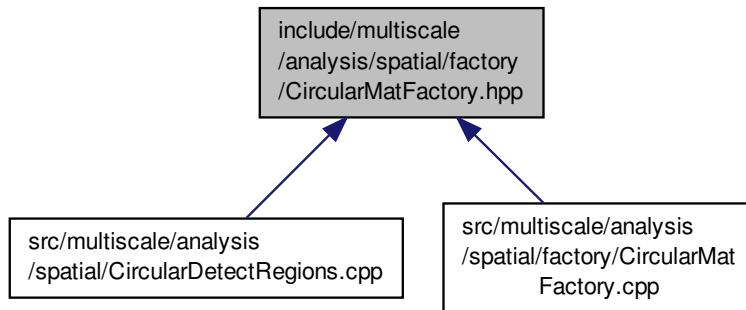
Referenced by multiscale::analysis::Cluster::fieldValuesToString(), multiscale::analysis::Region::fieldValuesToString(), multiscale::video::RectangularEntityCsvToInputFilesConverter::initOutputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initOutputFile(), multiscale::video::PolarCsvToInputFilesConverter::initOutputFile(), multiscale::video::RectangularGnuplotScriptGenerator::outputContent(), multiscale::video::RectangularEntityCsvToInputFilesConverter::processLine(), multiscale::video::RectangularCsvToInputFilesConverter::processLine(), multiscale::video::PolarCsvToInputFilesConverter::processLine(), and multiscale::analysis::Entity::toString().

8.10 include/multiscale/analysis/spatial/factory/CircularMatFactory.hpp File Reference

```
#include "multiscale/analysis/spatial/MatFactory.hpp"
Include dependency graph for CircularMatFactory.hpp:
```



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



Classes

- class [multiscale::analysis::CircularMatFactory](#)
Class for creating a Mat object considering a circular grid.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define [ERR_UNIMPLEMENTED_METHOD](#) "The method you called is not implemented."
- #define [INTENSITY_MAX](#) 255
- #define [ROI_START_X](#) 1024
- #define [ROI_START_Y](#) 786
- #define [ROI_RADIUS](#) 615
- #define [INPUT_IMG_WIDTH](#) 2048
- #define [INPUT_IMG_HEIGHT](#) 1572
- #define [COLOURBAR_MAX_X](#) 1775
- #define [COLOURBAR_MAX_Y](#) 56

8.10.1 Macro Definition Documentation

8.10.1.1 #define COLOURBAR_MAX_X 1775

Definition at line 20 of file CircularMatFactory.hpp.

Referenced by `multiscale::analysis::RectangularMatFactory::maxColourBarIntensityFromViewerImage()`, and `multiscale::analysis::CircularMatFactory::maxColourBarIntensityFromViewerImage()`.

8.10.1.2 #define COLOURBAR_MAX_Y 56

Definition at line 21 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::RectangularMatFactory::maxColourBarIntensityFromViewerImage(), and multiscale::analysis::CircularMatFactory::maxColourBarIntensityFromViewerImage().

8.10.1.3 #define ERR_UNIMPLEMENTED_METHOD "The method you called is not implemented."

Definition at line 9 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::processConcentrations().

8.10.1.4 #define INPUT_IMG_HEIGHT 1572

Definition at line 18 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::isValidViewerImage(), and multiscale::analysis::RectangularMatFactory::isValidViewerImage().

8.10.1.5 #define INPUT_IMG_WIDTH 2048

Definition at line 17 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::isValidViewerImage(), and multiscale::analysis::RectangularMatFactory::isValidViewerImage().

8.10.1.6 #define INTENSITY_MAX 255

Definition at line 11 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::createCircularMask(), multiscale::analysis::RegionDetector::outputResultsToImage(), multiscale::analysis::RegionDetector::regionArea(), multiscale::analysis::RegionDetector::regionClusterednessDegree(), and multiscale::analysis::RegionDetector::regionDensity().

8.10.1.7 #define ROI_RADIUS 615

Definition at line 15 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::createFromViewerImage().

8.10.1.8 #define ROI_START_X 1024

Definition at line 13 of file CircularMatFactory.hpp.

Referenced by multiscale::analysis::RectangularMatFactory::createFromViewerImage(), and multiscale::analysis::CircularMatFactory::createFromViewerImage().

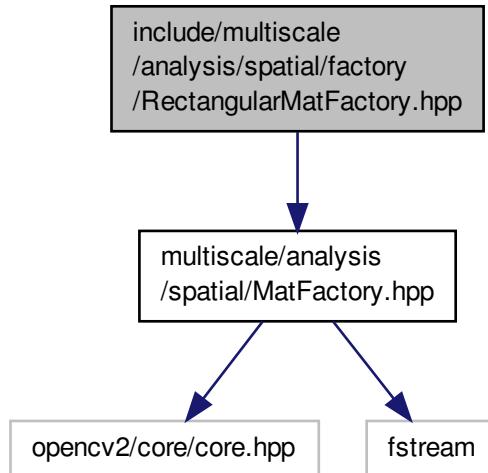
8.10.1.9 #define ROI_START_Y 786

Definition at line 14 of file CircularMatFactory.hpp.

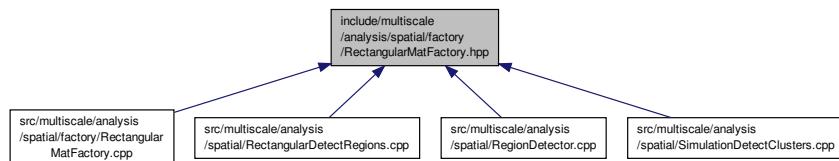
Referenced by multiscale::analysis::RectangularMatFactory::createFromViewerImage(), and multiscale::analysis::CircularMatFactory::createFromViewerImage().

8.11 include/multiscale/analysis/spatial/factory/RectangularMatFactory.hpp File Reference

```
#include "multiscale/analysis/spatial/MatFactory.hpp"
Include dependency graph for RectangularMatFactory.hpp:
```



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



Classes

- class [multiscale::analysis::RectangularMatFactory](#)
Class for creating a Mat object considering a rectangular grid.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define [ERR_CONC](#) "All concentrations have to be between 0 and 1."

- `#define ROI_START_X 321`
- `#define ROI_START_Y 318`
- `#define ROI_WIDTH 1407`
- `#define ROI_HEIGHT 1358`
- `#define INPUT_IMG_WIDTH 2048`
- `#define INPUT_IMG_HEIGHT 2048`
- `#define COLOURBAR_MAX_X 1799`
- `#define COLOURBAR_MAX_Y 320`

8.11.1 Macro Definition Documentation

8.11.1.1 `#define COLOURBAR_MAX_X 1799`

Definition at line 18 of file RectangularMatFactory.hpp.

8.11.1.2 `#define COLOURBAR_MAX_Y 320`

Definition at line 19 of file RectangularMatFactory.hpp.

8.11.1.3 `#define ERR_CONC "All concentrations have to be between 0 and 1."`

Definition at line 8 of file RectangularMatFactory.hpp.

Referenced by multiscale::analysis::RectangularMatFactory::processConcentrations(), multiscale::video::CartesianToConcentrationsConverter::readConcentrations(), and multiscale::video::CartesianToPolarConverter::readConcentrations().

8.11.1.4 `#define INPUT_IMG_HEIGHT 2048`

Definition at line 16 of file RectangularMatFactory.hpp.

8.11.1.5 `#define INPUT_IMG_WIDTH 2048`

Definition at line 15 of file RectangularMatFactory.hpp.

8.11.1.6 `#define ROI_HEIGHT 1358`

Definition at line 13 of file RectangularMatFactory.hpp.

Referenced by multiscale::analysis::RectangularMatFactory::createFromViewerImage().

8.11.1.7 `#define ROI_START_X 321`

Definition at line 10 of file RectangularMatFactory.hpp.

8.11.1.8 `#define ROI_START_Y 318`

Definition at line 11 of file RectangularMatFactory.hpp.

8.11.1.9 #define ROI_WIDTH 1407

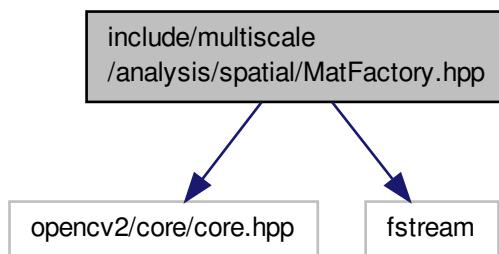
Definition at line 12 of file RectangularMatFactory.hpp.

Referenced by multiscale::analysis::RectangularMatFactory::createFromViewerImage().

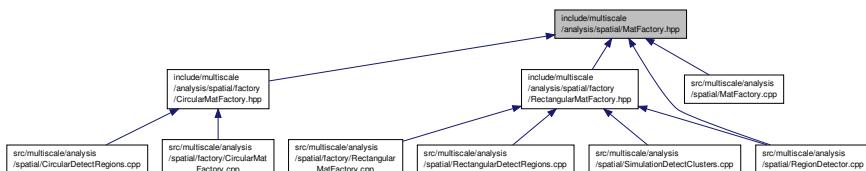
8.12 include/multiscale/analysis/spatial/MatFactory.hpp File Reference

```
#include "opencv2/core/core.hpp"
#include <fstream>
```

Include dependency graph for MatFactory.hpp:



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



Classes

- class [multiscale::analysis::MatFactory](#)

Class for creating a Mat object.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- `#define ERR_INPUT_OPEN "The input file could not be opened."`
- `#define ERR_IMG_RESOLUTION "The resolution of the input image is not the expected one."`
- `#define ERR_IN_EXTRA_DATA "The input file contains more data than required."`

8.12.1 Macro Definition Documentation

8.12.1.1 `#define ERR_IMG_RESOLUTION "The resolution of the input image is not the expected one."`

Definition at line 11 of file MatFactory.hpp.

Referenced by multiscale::analysis::CircularMatFactory::isValidViewerImage(), and multiscale::analysis::RectangularMatFactory::isValidViewerImage().

8.12.1.2 `#define ERR_IN_EXTRA_DATA "The input file contains more data than required."`

Definition at line 12 of file MatFactory.hpp.

Referenced by multiscale::analysis::MatFactory::create(), multiscale::video::CartesianToConcentrationsConverter::readInputData(), and multiscale::video::CartesianToPolarConverter::readInputData().

8.12.1.3 `#define ERR_INPUT_OPEN "The input file could not be opened."`

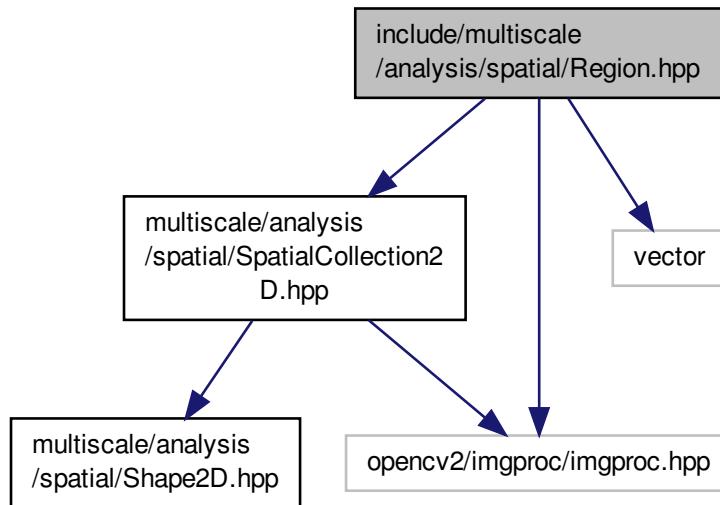
Definition at line 10 of file MatFactory.hpp.

Referenced by multiscale::analysis::MatFactory::initInputFile(), multiscale::video::RectangularEntityCsvToInputFilesConverter::initInputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initInputFile(), multiscale::video::PolarCsvToInputFilesConverter::initInputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initMaximumConcentration(), multiscale::video::PolarCsvToInputFilesConverter::initMaximumConcentration(), multiscale::analysis::CircularMatFactory::isValidViewerImage(), multiscale::analysis::RectangularMatFactory::isValidViewerImage(), multiscale::video::CartesianToConcentrationsConverter::readInputData(), multiscale::video::CartesianToPolarConverter::readInputData(), multiscale::video::RectangularEntityCsvToInputFilesConverter::validateInput(), multiscale::video::RectangularCsvToInputFilesConverter::validateInput(), and multiscale::video::PolarCsvToInputFilesConverter::validateInput().

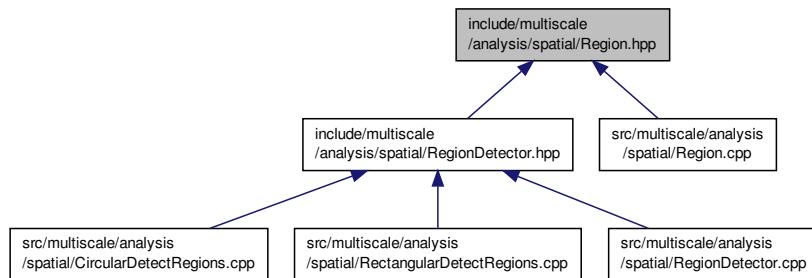
8.13 include/multiscale/analysis/spatial/Region.hpp File Reference

```
#include "multiscale/analysis/spatial/SpatialCollection2D.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include <vector>
```

Include dependency graph for Region.hpp:



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



Classes

- class [multiscale::analysis::Region](#)
Class for representing a region.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- `#define CONTOUR_ORIENTED false`

- `#define CONTOUR_CLOSED true`

8.13.1 Macro Definition Documentation

8.13.1.1 `#define CONTOUR_CLOSED true`

Definition at line 14 of file Region.hpp.

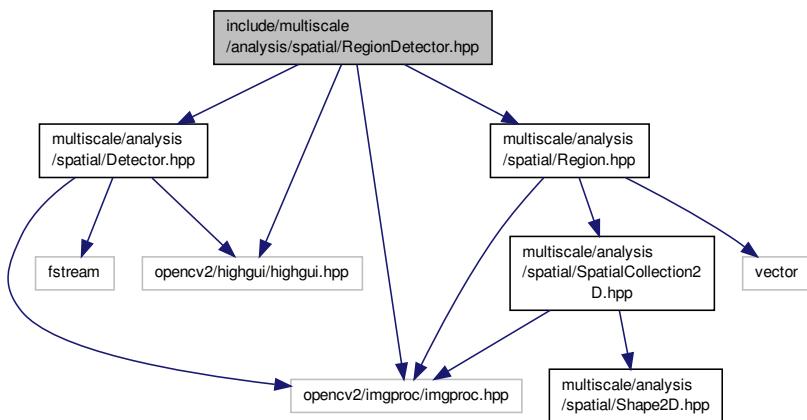
Referenced by multiscale::analysis::Region::updatePerimeter().

8.13.1.2 `#define CONTOUR_ORIENTED false`

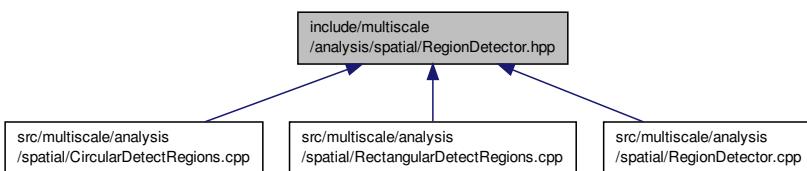
Definition at line 13 of file Region.hpp.

8.14 include/multiscale/analysis/spatial/RegionDetector.hpp File Reference

```
#include "multiscale/analysis/spatial/Detector.hpp"
#include "multiscale/analysis/spatial/Region.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
Include dependency graph for RegionDetector.hpp:
```



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



Classes

- class [multiscale::analysis::RegionDetector](#)

Class for detecting regions of high intensity in grayscale images.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- #define OUTPUT_CLUSTEREDNESS "Average clusteredness degree: "
- #define OUTPUT_DENSITY "Average density: "
- #define TRACKBAR_ALPHA "Alpha"
- #define TRACKBAR_BETA "Beta"
- #define TRACKBAR_KERNEL "Gaussian blur kernel size"
- #define TRACKBAR_MORPH "Morphological open, number of iterations"
- #define TRACKBAR_CANNY "Canny lower threshold"
- #define TRACKBAR_EPSILON "Epsilon"
- #define TRACKBAR_REGION_AREA_THRESH "Region area threshold"
- #define TRACKBAR_THRESHOLD "Threshold value"
- #define USE_CANNY_L2 true
- #define CONTOUR_AREA_ORIENTED false
- #define ALPHA_REAL_MIN 1.0
- #define ALPHA_REAL_MAX 3.0
- #define BETA_REAL_MIN -100
- #define BETA_REAL_MAX 100
- #define ALPHA_MAX 1000
- #define BETA_MAX 200
- #define KERNEL_MAX 2000
- #define MORPH_ITER_MAX 100
- #define CANNY_THRESH_MAX 100
- #define EPSILON_MAX 100
- #define REGION_AREA_THRESH_MAX 200000
- #define THRESHOLD_MAX 255
- #define THRESHOLD_CLUSTEREDNESS 0
- #define INTENSITY_MAX 255
- #define POLYGON_CLOSED true
- #define DISPLAY_LINE_THICKNESS 10
- #define ENCLOSING_RECT_VERTICES 4

8.14.1 Macro Definition Documentation

8.14.1.1 #define ALPHA_MAX 1000

Definition at line 33 of file RegionDetector.hpp.

Referenced by [multiscale::analysis::RegionDetector::convertAlpha\(\)](#), and [multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars\(\)](#).

8.14.1.2 #define ALPHA_REAL_MAX 3.0

Definition at line 29 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::convertAlpha().

8.14.1.3 #define ALPHA_REAL_MIN 1.0

Definition at line 28 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::convertAlpha().

8.14.1.4 #define BETA_MAX 200

Definition at line 34 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::convertBeta(), and multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.5 #define BETA_REAL_MAX 100

Definition at line 31 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::convertBeta().

8.14.1.6 #define BETA_REAL_MIN -100

Definition at line 30 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::convertBeta().

8.14.1.7 #define CANNY_THRESH_MAX 100

Definition at line 37 of file RegionDetector.hpp.

8.14.1.8 #define CONTOUR_AREA_ORIENTED false

Definition at line 26 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::isValidRegion(), multiscale::analysis::RegionDetector::regionArea(), and multiscale::analysis::RegionDetector::regionHolesArea().

8.14.1.9 #define DISPLAY_LINE_THICKNESS 10

Definition at line 46 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::outputResultsToImage().

8.14.1.10 #define ENCLOSING_RECT_VERTICES 4

Definition at line 48 of file RegionDetector.hpp.

Referenced by convertVertices().

8.14.1.11 `#define EPSILON_MAX 100`

Definition at line 38 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.12 `#define INTENSITY_MAX 255`

Definition at line 42 of file RegionDetector.hpp.

8.14.1.13 `#define KERNEL_MAX 2000`

Definition at line 35 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.14 `#define MORPH_ITER_MAX 100`

Definition at line 36 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.15 `#define OUTPUT_CLUSTEREDNESS "Average clusteredness degree: "`

Definition at line 13 of file RegionDetector.hpp.

8.14.1.16 `#define OUTPUT_DENSITY "Average density: "`

Definition at line 14 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::outputAveragedMeasuresToCsvFile().

8.14.1.17 `#define POLYGON_CLOSED true`

Definition at line 44 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::outputResultsToImage().

8.14.1.18 `#define REGION_AREA_THRESH_MAX 200000`

Definition at line 39 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.19 `#define THRESHOLD_CLUSTEREDNESS 0`

Definition at line 41 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::regionArea(), and multiscale::analysis::RegionDetector::regionClusterednessDegree().

8.14.1.20 `#define THRESHOLD_MAX 255`

Definition at line 40 of file RegionDetector.hpp.

8.14.1.21 `#define TRACKBAR_ALPHA "Alpha"`

Definition at line 16 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.22 `#define TRACKBAR_BETA "Beta"`

Definition at line 17 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.23 `#define TRACKBAR_CANNY "Canny lower threshold"`

Definition at line 20 of file RegionDetector.hpp.

8.14.1.24 `#define TRACKBAR_EPSILON "Epsilon"`

Definition at line 21 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.25 `#define TRACKBAR_KERNEL "Gaussian blur kernel size"`

Definition at line 18 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.26 `#define TRACKBAR_MORPH "Morphological open, number of iterations"`

Definition at line 19 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.27 `#define TRACKBAR_REGION_AREA_THRESH "Region area threshold"`

Definition at line 22 of file RegionDetector.hpp.

Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.28 `#define TRACKBAR_THRESHOLD "Threshold value"`

Definition at line 23 of file RegionDetector.hpp.

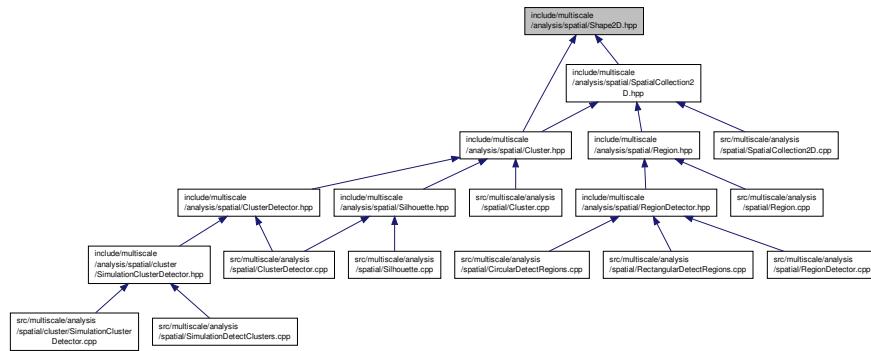
Referenced by multiscale::analysis::RegionDetector::createDetectorSpecificTrackbars().

8.14.1.29 `#define USE_CANNY_L2 true`

Definition at line 25 of file RegionDetector.hpp.

8.15 include/multiscale/analysis/spatial/Shape2D.hpp File Reference

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



Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

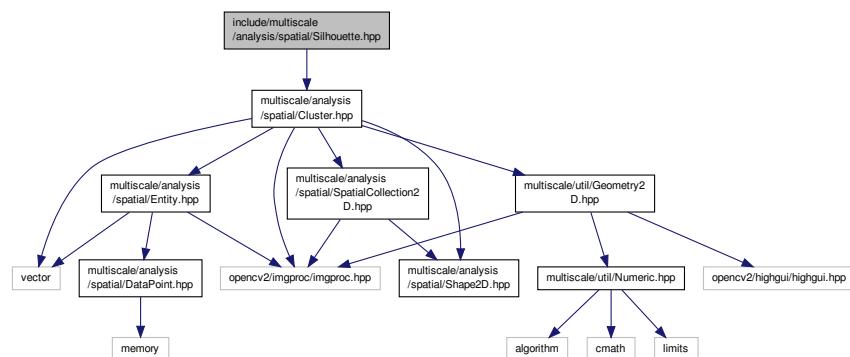
Enumerations

- enum [multiscale::analysis::Shape2D](#) { [multiscale::analysis::Triangle](#) = 1, [multiscale::analysis::Rectangle](#) = 2, [multiscale::analysis::Circle](#) = 3, [multiscale::analysis::Undefined](#) = 4 }

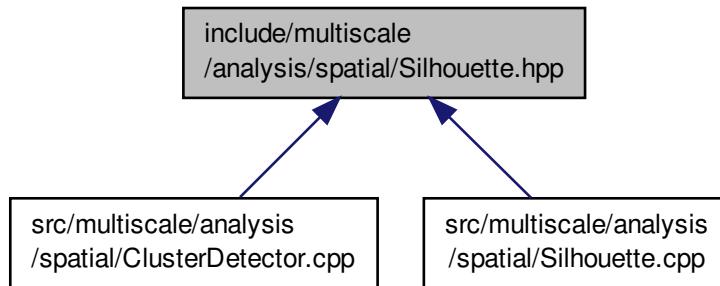
Enumeration for determining the type of a 2D shape.

8.16 include/multiscale/analysis/spatial/Silhouette.hpp File Reference

```
#include "multiscale/analysis/spatial/Cluster.hpp"
Include dependency graph for Silhouette.hpp:
```



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



Classes

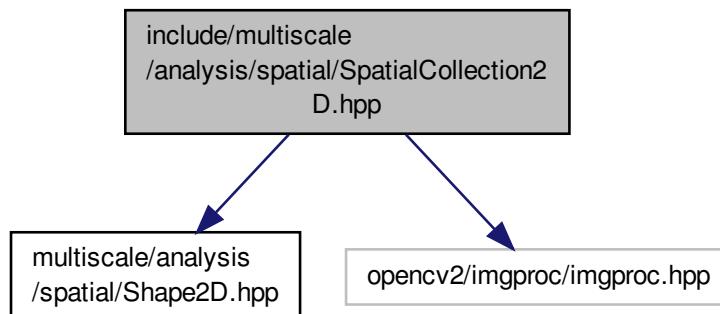
- class [multiscale::analysis::Silhouette](#)

Namespaces

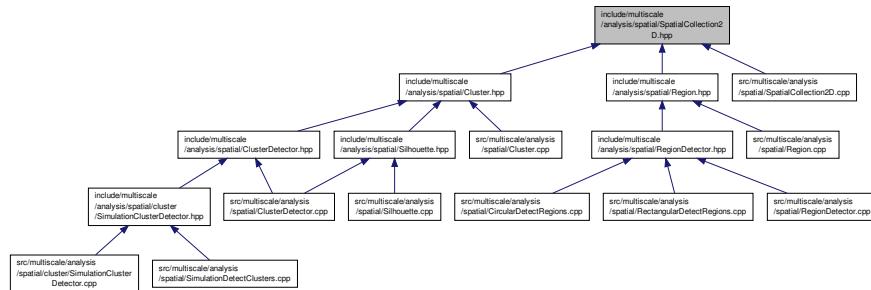
- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

8.17 include/multiscale/analysis/spatial/SpatialCollection2D.hpp File Reference

```
#include "multiscale/analysis/spatial/Shape2D.hpp"
#include "opencv2/imgproc/imgproc.hpp"
Include dependency graph for SpatialCollection2D.hpp:
```



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



Classes

- class [multiscale::analysis::SpatialCollection2D](#)
Class for representing a collection of objects in 2D space.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::analysis](#)

Macros

- `#define STR_TRIANGLE "triangle"`
- `#define STR_RECTANGLE "rectangle"`
- `#define STR_CIRCLE "circle"`
- `#define STR_UNDEFINED "undefined"`
- `#define OUTPUT_SEPARATOR ","`
- `#define ERR_INPUT "Invalid input parameters were provided to the constructor."`
- `#define ERR_UNDEFINED_SHAPE "The shape of the given cluster is undefined."`
- `#define CONVEX_HULL_CLOCKWISE true`

8.17.1 Macro Definition Documentation

8.17.1.1 `#define CONVEX_HULL_CLOCKWISE true`

Definition at line 21 of file `SpatialCollection2D.hpp`.

Referenced by `multiscale::analysis::Cluster::getEntitiesConvexHull()`, `multiscale::MinEnclosingTriangleFinder::initialiseConvexPolygon()`, and `multiscale::analysis::Region::isTriangularMeasure()`.

8.17.1.2 `#define ERR_INPUT "Invalid input parameters were provided to the constructor."`

Definition at line 18 of file `SpatialCollection2D.hpp`.

8.17.1.3 `#define ERR_UNDEFINED_SHAPE "The shape of the given cluster is undefined."`

Definition at line 19 of file `SpatialCollection2D.hpp`.

Referenced by `multiscale::analysis::SimulationClusterDetector::outputClusterShape()`.

8.17.1.4 `#define OUTPUT_SEPARATOR ","`

Definition at line 16 of file SpatialCollection2D.hpp.

8.17.1.5 `#define STR_CIRCLE "circle"`

Definition at line 13 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::SpatialCollection2D::shapeAsString().

8.17.1.6 `#define STR_RECTANGLE "rectangle"`

Definition at line 12 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::SpatialCollection2D::shapeAsString().

8.17.1.7 `#define STR_TRIANGLE "triangle"`

Definition at line 11 of file SpatialCollection2D.hpp.

Referenced by multiscale::analysis::SpatialCollection2D::shapeAsString().

8.17.1.8 `#define STR_UNDEFINED "undefined"`

Definition at line 14 of file SpatialCollection2D.hpp.

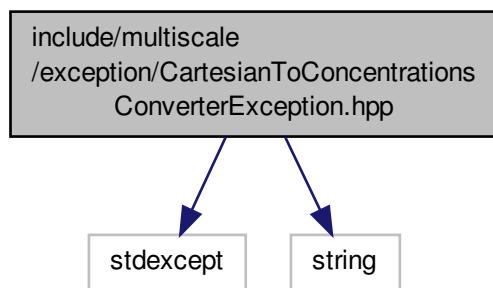
Referenced by multiscale::analysis::SpatialCollection2D::shapeAsString().

8.18 include/multiscale/exception/CartesianToConcentrationsConverterException.hpp

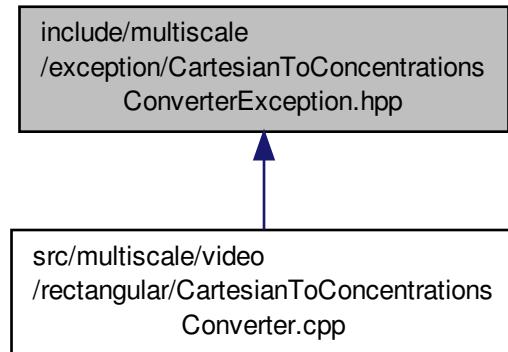
File Reference

```
#include <stdexcept>
#include <string>
```

Include dependency graph for CartesianToConcentrationsConverterException.hpp:



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



Classes

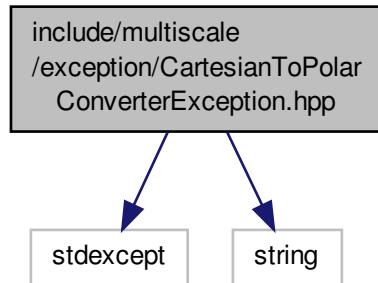
- class [multiscale::CartesianToConcentrationsConverterException](#)
Exception class for the CartesianToConcentrationsConverter class.

Namespaces

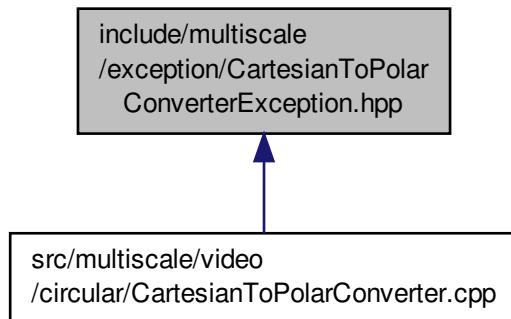
- namespace [multiscale](#)

8.19 include/multiscale/exception/CartesianToPolarConverterException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for CartesianToPolarConverterException.hpp:
```



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



Classes

- class [multiscale::CartesianToPolarConverterException](#)

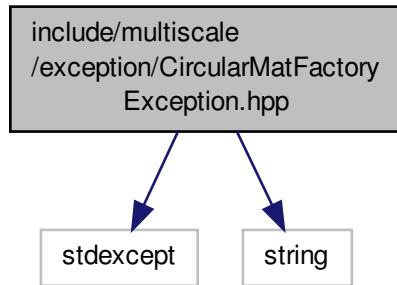
Exception class for the CartesianToPolarConverter class.

Namespaces

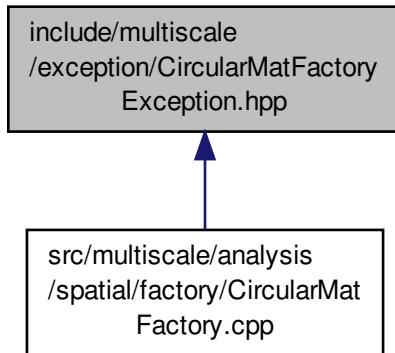
- namespace [multiscale](#)

8.20 include/multiscale/exception/CircularMatFactoryException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for CircularMatFactoryException.hpp:
```



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



Classes

- class [multiscale::CircularMatFactoryException](#)

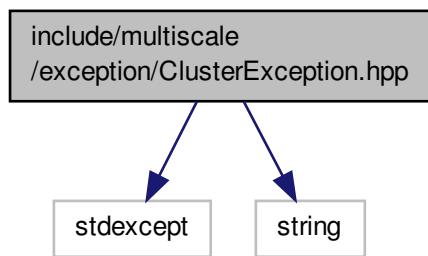
Exception class for the CircularMatFactory instances.

Namespaces

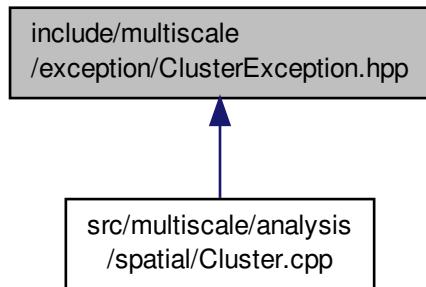
- namespace [multiscale](#)

8.21 include/multiscale/exception/ClusterException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for ClusterException.hpp:
```



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



Classes

- class [multiscale::ClusterException](#)

Exception class for the Cluster instances.

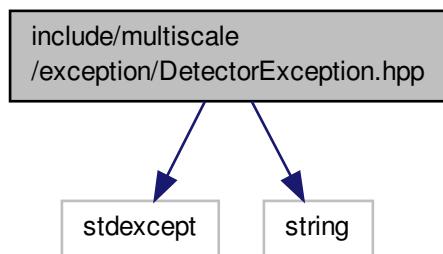
Namespaces

- namespace [multiscale](#)

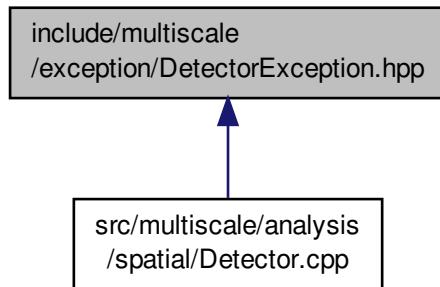
8.22 include/multiscale/exception/DetectorException.hpp File Reference

```
#include <stdexcept>
#include <string>
```

Include dependency graph for DetectorException.hpp:



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



Classes

- class [multiscale::DetectorException](#)

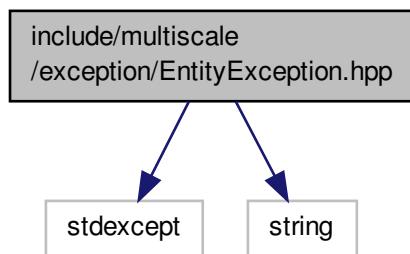
Exception class for the Detector class.

Namespaces

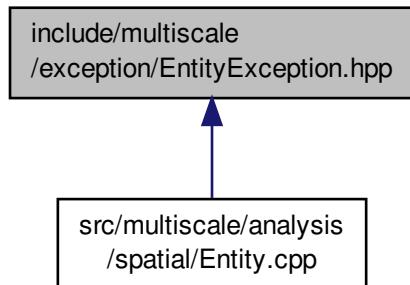
- namespace [multiscale](#)

8.23 include/multiscale/exception/EntityException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for EntityException.hpp:
```



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



Classes

- class [multiscale::EntityException](#)

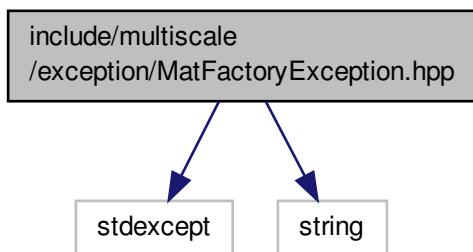
Exception class for the Entity instances.

Namespaces

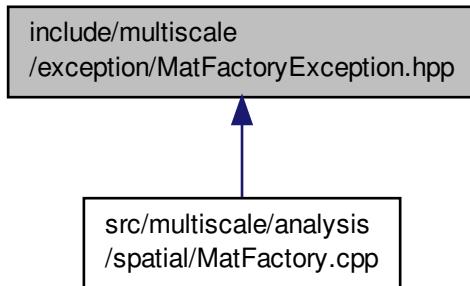
- namespace [multiscale](#)

8.24 include/multiscale/exception/MatFactoryException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for MatFactoryException.hpp:
```



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



Classes

- class [multiscale::MatFactoryException](#)

Exception class for the MatFactory class.

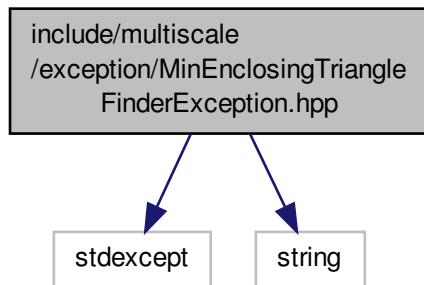
Namespaces

- namespace [multiscale](#)

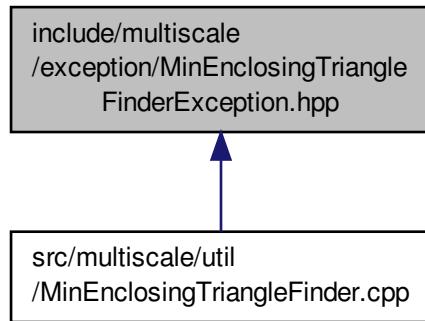
8.25 include/multiscale/exception/MinEnclosingTriangleFinderException.hpp File Reference

```
#include <stdexcept>
#include <string>
```

Include dependency graph for MinEnclosingTriangleFinderException.hpp:



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



Classes

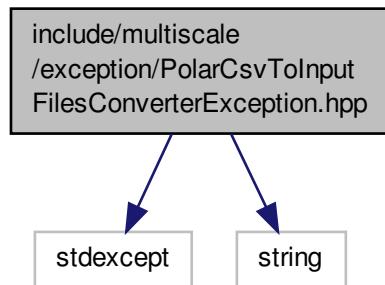
- class [multiscale::MinEnclosingTriangleFinderException](#)
Exception class for the minimum area enclosing triangle module.

Namespaces

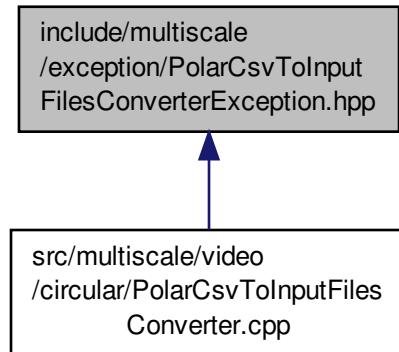
- namespace [multiscale](#)

8.26 include/multiscale/exception/PolarCsvToInputFilesConverterException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for PolarCsvToInputFilesConverterException.hpp:
```



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



Classes

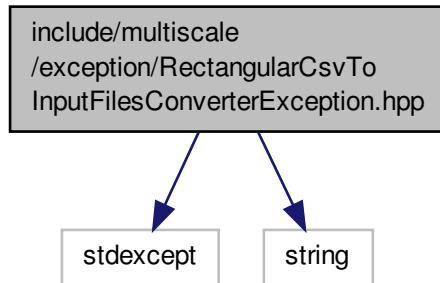
- class [multiscale::PolarCsvToInputFilesConverterException](#)
Exception class for the PolarCsvToInputFilesConverter class.

Namespaces

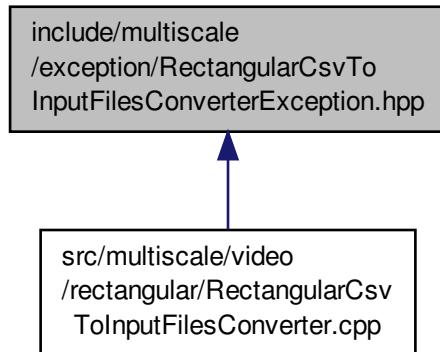
- namespace [multiscale](#)

8.27 include/multiscale/exception/RectangularCsvToInputFilesConverterException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for RectangularCsvToInputFilesConverterException.hpp:
```



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



Classes

- class [multiscale::RectangularCsvToInputFilesConverterException](#)
Exception class for the RectangularCsvToInputFilesConverter class.

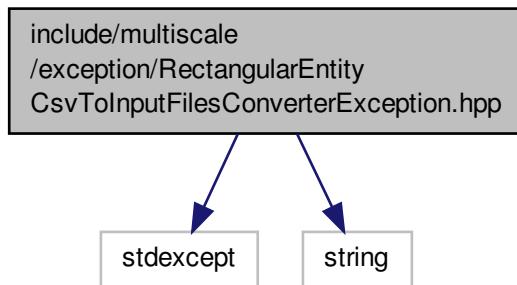
Namespaces

- namespace [multiscale](#)

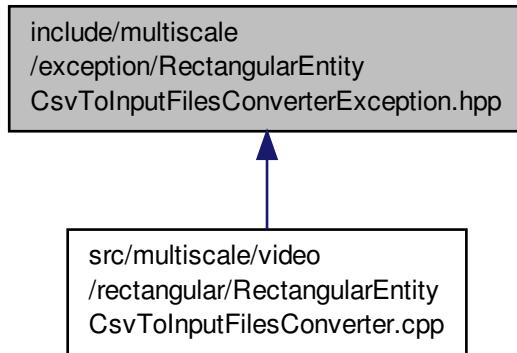
8.28 include/multiscale/exception/RectangularEntityCsvToInputFilesConverterException.hpp

File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for RectangularEntityCsvToInputFilesConverterException.hpp:
```



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



Classes

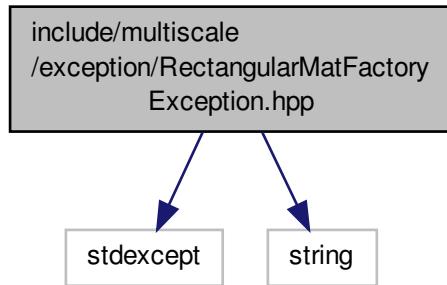
- class [multiscale::RectangularEntityCsvToInputFilesConverterException](#)
Exception class for the RectangularEntityCsvToInputFilesConverter class.

Namespaces

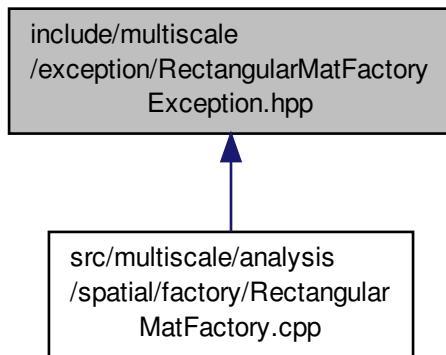
- namespace [multiscale](#)

8.29 include/multiscale/exception/RectangularMatFactoryException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for RectangularMatFactoryException.hpp:
```



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



Classes

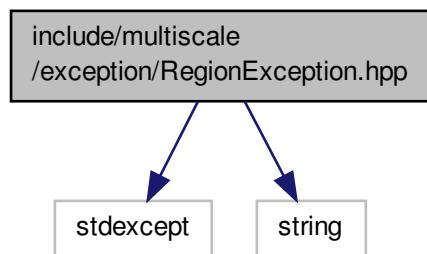
- class [multiscale::RectangularMatFactoryException](#)
Exception class for the RectangularMatFactory instances.

Namespaces

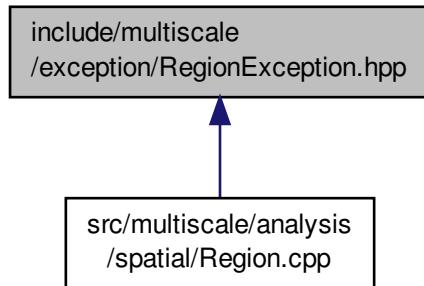
- namespace [multiscale](#)

8.30 include/multiscale/exception/RegionException.hpp File Reference

```
#include <stdexcept>
#include <string>
Include dependency graph for RegionException.hpp:
```



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



Classes

- class [multiscale::RegionException](#)

Exception class for the Region instances.

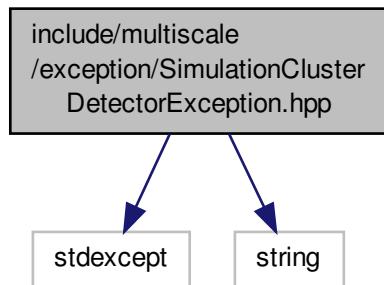
Namespaces

- namespace [multiscale](#)

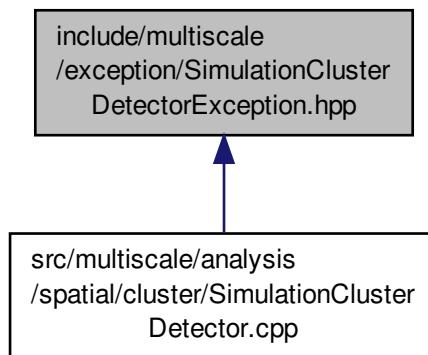
8.31 include/multiscale/exception/SimulationClusterDetectorException.hpp File Reference

```
#include <stdexcept>
#include <string>
```

Include dependency graph for SimulationClusterDetectorException.hpp:



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



Classes

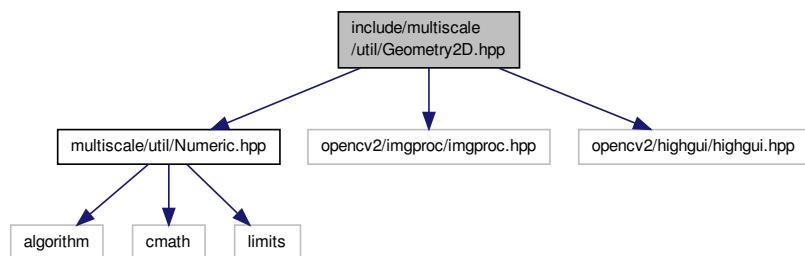
- class [multiscale::SimulationClusterDetectorException](#)
Exception class for the SimulationClusterDetector instances.

Namespaces

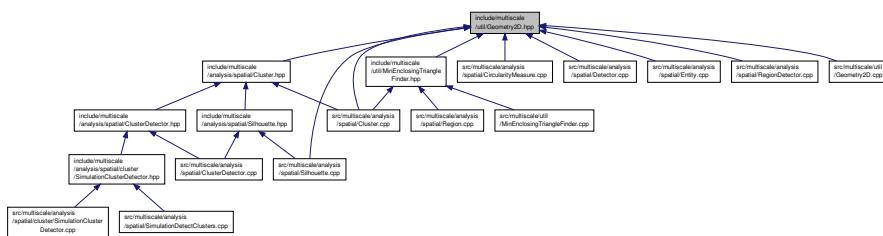
- namespace [multiscale](#)

8.32 include/multiscale/util/Geometry2D.hpp File Reference

```
#include "multiscale/util/Numeric.hpp"
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
Include dependency graph for Geometry2D.hpp:
```



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



Classes

- class [multiscale::Geometry2D](#)

Two-dimensional geometric operations.

Namespaces

- namespace [multiscale](#)

Macros

- `#define PI 3.14159265358979323846264338327950288419716939937510`
- `#define MATRIX_START_INDEX 1`

8.32.1 Macro Definition Documentation

8.32.1.1 #define MATRIX_START_INDEX 1

Definition at line 13 of file Geometry2D.hpp.

Referenced by multiscale::Geometry2D::isPointOnEdge().

8.32.1.2 #define PI 3.14159265358979323846264338327950288419716939937510

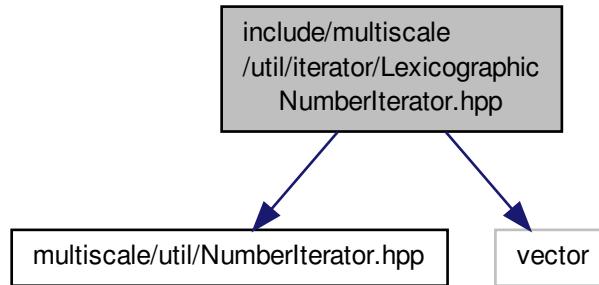
Definition at line 12 of file Geometry2D.hpp.

Referenced by multiscale::Geometry2D::angleBtwPoints(), multiscale::Geometry2D::angleOfLineWrtOxAxis(), multiscale::analysis::CircularityMeasure::compute(), multiscale::analysis::Region::isCircularMeasure(), and multiscale::analysis::Cluster::isCircularMeasure().

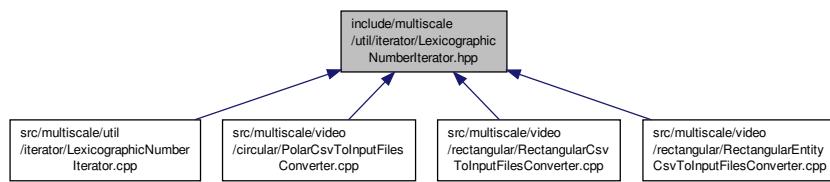
8.33 include/multiscale/util/iterator/LexicographicNumberIterator.hpp File Reference

```
#include "multiscale/util/NumberIterator.hpp"
#include <vector>
```

Include dependency graph for LexicographicNumberIterator.hpp:



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



Classes

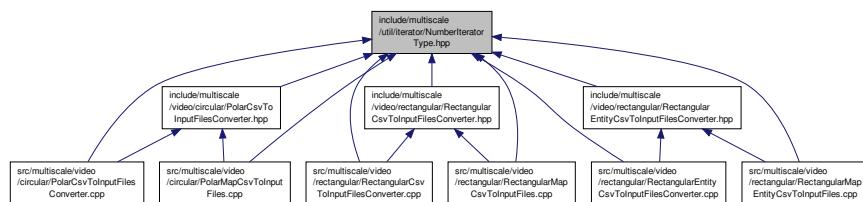
- class [multiscale::LexicographicNumberIterator](#)
Iterator class starting at 1 and ending at the provided upper bound considering that each number is followed by an " _ ".

Namespaces

- namespace [multiscale](#)

8.34 include/multiscale/util/iterator/NumberIteratorType.hpp File Reference

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



Namespaces

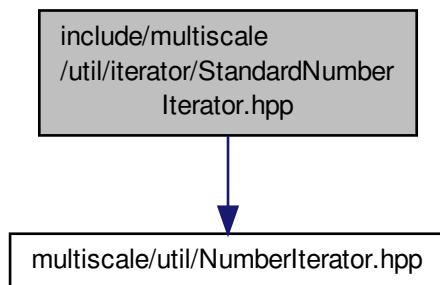
- namespace [multiscale](#)

Enumerations

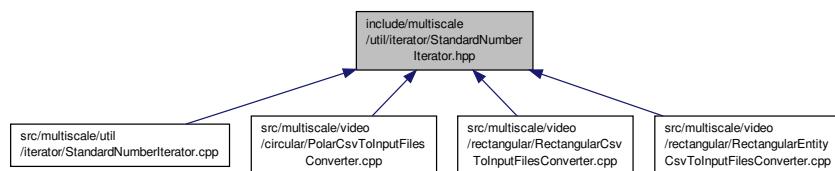
- enum [multiscale::NumberIteratorType](#) { [multiscale::STANDARD](#) = 1, [multiscale::LEXICOGRAPHIC](#) = 2 }
- The type of the number iterator.*

8.35 include/multiscale/util/iterator/StandardNumberIterator.hpp File Reference

```
#include "multiscale/util/NumberIterator.hpp"
Include dependency graph for StandardNumberIterator.hpp:
```



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



Classes

- class [multiscale::StandardNumberIterator](#)

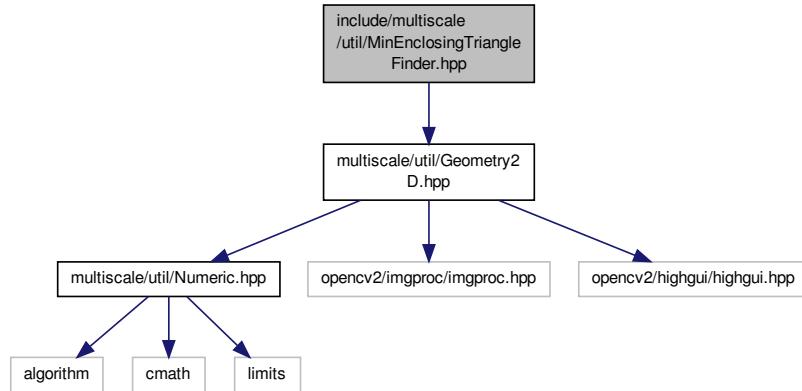
Iterator class starting at 1 and iterating over all natural numbers until the provided upper bound is reached.

Namespaces

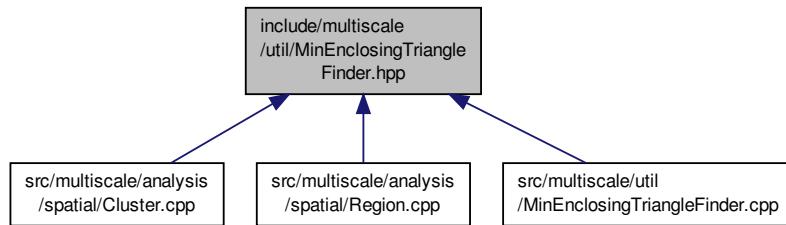
- namespace [multiscale](#)

8.36 include/multiscale/util/MinEnclosingTriangleFinder.hpp File Reference

```
#include "multiscale/util/Geometry2D.hpp"
Include dependency graph for MinEnclosingTriangleFinder.hpp:
```



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



Classes

- class [multiscale::MinEnclosingTriangleFinder](#)
Class for computing the minimum area enclosing triangle for a given polygon.

Namespaces

- namespace [multiscale](#)

Macros

- `#define CONVEX_HULL_CLOCKWISE true`
- `#define INTERSECTS_BELOW 1`
- `#define INTERSECTS_ABOVE 2`
- `#define INTERSECTS_CRITICAL 3`
- `#define INTERSECTS_LIMIT 4`

- #define `ERR_MIDPOINT_SIDE_B` "The position of the middle point of side B could not be determined."
- #define `ERR_SIDE_B_GAMMA` "The position of side B could not be determined, because gamma(b) could not be computed."
- #define `ERR_VERTEX_C_ON_SIDE_B` "The position of the vertex C on side B could not be determined, because the considered lines do not intersect."
- #define `ERR_TRIANGLE_VERTICES` "The position of the triangle vertices could not be determined, because the sides of the triangle do not intersect."
- #define `VALIDATION_SIDE_A_TANGENT` 0
- #define `VALIDATION_SIDE_B_TANGENT` 1
- #define `VALIDATION_SIDES_FLUSH` 2

8.36.1 Macro Definition Documentation

8.36.1.1 #define CONVEX_HULL_CLOCKWISE true

Definition at line 9 of file MinEnclosingTriangleFinder.hpp.

8.36.1.2 #define ERR_MIDPOINT_SIDE_B "The position of the middle point of side B could not be determined."

Definition at line 16 of file MinEnclosingTriangleFinder.hpp.

8.36.1.3 #define ERR_SIDE_B_GAMMA "The position of side B could not be determined, because gamma(b) could not be computed."

Definition at line 17 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::updateSideB().

8.36.1.4 #define ERR_TRIANGLE_VERTICES "The position of the triangle vertices could not be determined, because the sides of the triangle do not intersect."

Definition at line 19 of file MinEnclosingTriangleFinder.hpp.

8.36.1.5 #define ERR_VERTEX_C_ON_SIDE_B "The position of the vertex C on side B could not be determined, because the considered lines do not intersect."

Definition at line 18 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::findVertexCOnSideB().

8.36.1.6 #define INTERSECTS ABOVE 2

Definition at line 12 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::intersectsAbove(), and multiscale::MinEnclosingTriangleFinder::intersectsAboveOrBelow().

8.36.1.7 #define INTERSECTS BELOW 1

Definition at line 11 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::intersects(), multiscale::MinEnclosingTriangleFinder::intersectsAboveOrBelow(), and multiscale::MinEnclosingTriangleFinder::intersectsBelow().

8.36.1.8 #define INTERSECTS_CRITICAL 3

Definition at line 13 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::intersects().

8.36.1.9 #define INTERSECTS_LIMIT 4

Definition at line 14 of file MinEnclosingTriangleFinder.hpp.

8.36.1.10 #define VALIDATION_SIDE_A_TANGENT 0

Definition at line 21 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and multiscale::MinEnclosingTriangleFinder::updateSidesBA().

8.36.1.11 #define VALIDATION_SIDE_B_TANGENT 1

Definition at line 22 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::isValidMinimalTriangle(), and multiscale::MinEnclosingTriangleFinder::updateSideB().

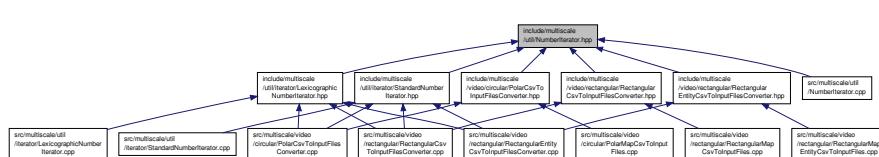
8.36.1.12 #define VALIDATION_SIDES_FLUSH 2

Definition at line 23 of file MinEnclosingTriangleFinder.hpp.

Referenced by multiscale::MinEnclosingTriangleFinder::updateSidesBA().

8.37 include/multiscale/util/NumberIterator.hpp File Reference

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



Classes

- class [multiscale::NumberIterator](#)

Abstract class representing a number iterator.

Namespaces

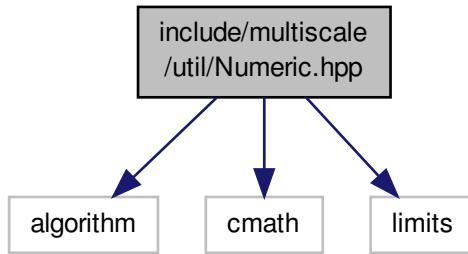
- namespace [multiscale](#)

8.38 include/multiscale/util/Numeric.hpp File Reference

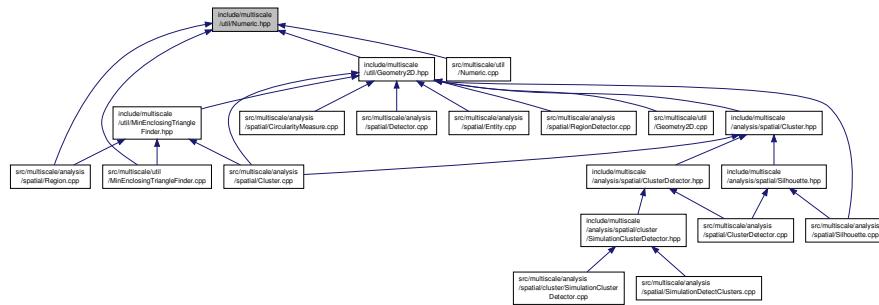
```
#include <algorithm>
#include <cmath>
```

```
#include <limits>
```

Include dependency graph for Numeric.hpp:



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



Classes

- class [multiscale::Numeric](#)

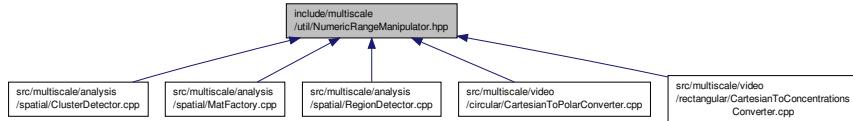
Class for manipulating numbers (shorts, ints, floats, doubles etc.)

Namespaces

- namespace [multiscale](#)

8.39 include/multiscale/util/NumericRangeManipulator.hpp File Reference

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



Classes

- class [multiscale::NumericRangeManipulator](#)

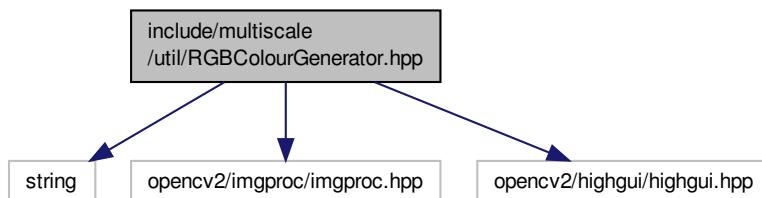
Operations for ranges of numeric values.

Namespaces

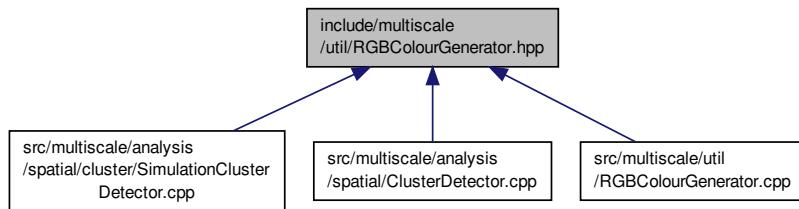
- namespace [multiscale](#)

8.40 include/multiscale/util/RGBColourGenerator.hpp File Reference

```
#include <string>
#include "opencv2/imgproc/imgproc.hpp"
#include "opencv2/highgui/highgui.hpp"
Include dependency graph for RGBColourGenerator.hpp:
```



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



Classes

- class [multiscale::RGBColourGenerator](#)

Generate a RGB colour.

Namespaces

- namespace [multiscale](#)

Macros

- #define HUE_MIN 0
- #define HUE_MAX 120
- #define SATURATION 1
- #define VALUE 1

8.40.1 Macro Definition Documentation

8.40.1.1 #define HUE_MAX 120

Definition at line 12 of file RGBColourGenerator.hpp.

Referenced by multiscale::RGBColourGenerator::generate().

8.40.1.2 #define HUE_MIN 0

Definition at line 11 of file RGBColourGenerator.hpp.

Referenced by multiscale::RGBColourGenerator::generate().

8.40.1.3 #define SATURATION 1

Definition at line 13 of file RGBColourGenerator.hpp.

Referenced by multiscale::RGBColourGenerator::generate().

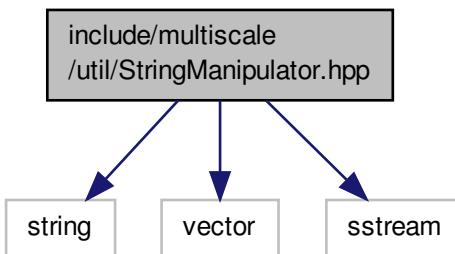
8.40.1.4 #define VALUE 1

Definition at line 14 of file RGBColourGenerator.hpp.

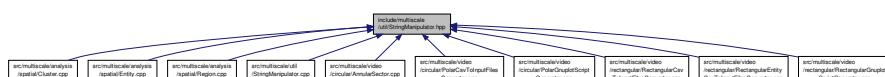
Referenced by multiscale::RGBColourGenerator::generate().

8.41 include/multiscale/util/StringManipulator.hpp File Reference

```
#include <string>
#include <vector>
#include <sstream>
Include dependency graph for StringManipulator.hpp:
```



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



Classes

- class [multiscale::StringManipulator](#)

Class for manipulating strings.

Namespaces

- namespace [multiscale](#)

Macros

- #define [DIR_SEPARATOR](#) '/'

8.41.1 Macro Definition Documentation

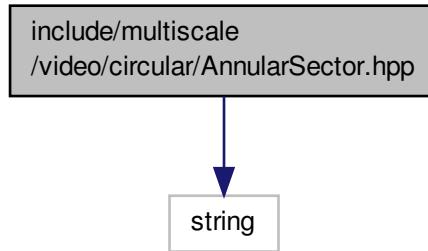
8.41.1.1 #define DIR_SEPARATOR '/'

Definition at line 8 of file StringManipulator.hpp.

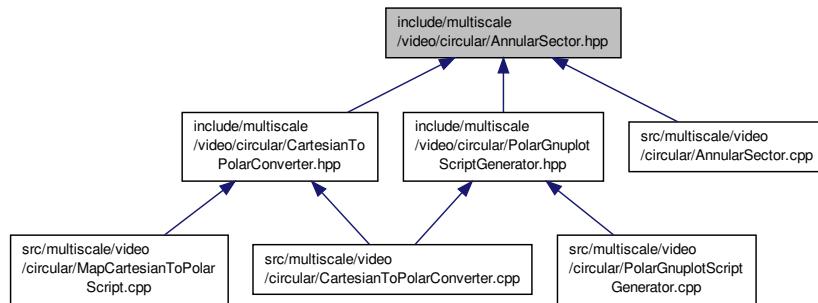
Referenced by multiscale::StringManipulator::filenameFromPath().

8.42 include/multiscale/video/circular/AnnularSector.hpp File Reference

```
#include <string>
Include dependency graph for AnnularSector.hpp:
```



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



Classes

- class [multiscale::video::AnnularSector](#)

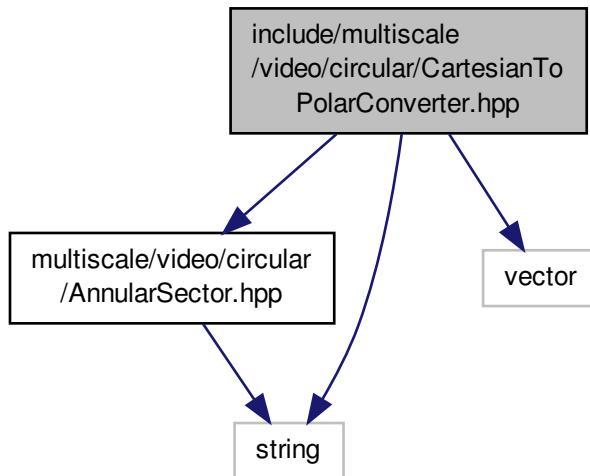
An annular sector is the basic element in the considered circular geometry.

Namespaces

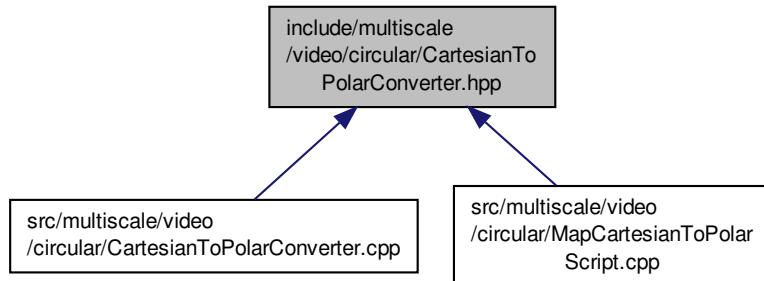
- namespace [multiscale](#)
- namespace [multiscale::video](#)

8.43 include/multiscale/video/circular/CartesianToPolarConverter.hpp File Reference

```
#include "multiscale/video/circular/AnnularSector.hpp"
#include <string>
#include <vector>
Include dependency graph for CartesianToPolarConverter.hpp:
```



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



Classes

- class [multiscale::video::CartesianToPolarConverter](#)
Converter from the rectangular geometry grid cells to annular sectors.

Namespaces

- namespace [multiscale](#)

- namespace `multiscale::video`

Macros

- `#define ERR_CONC "All concentrations have to be between 0 and 1."`
- `#define ERR_NONPOS_DIMENSION "The dimensions N and M must be positive."`
- `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`
- `#define ERR_INPUT_OPEN "The input file could not be opened"`
- `#define ERR_IN_EXTRA_DATA "The input file contains more data than required."`
- `#define OUTPUT_FILE_EXTENSION ".out"`
- `#define RADIUS_MIN 0.001`
- `#define RADIUS_MAX 0.3`

8.43.1 Macro Definition Documentation

8.43.1.1 `#define ERR_CONC "All concentrations have to be between 0 and 1."`

Definition at line 11 of file `CartesianToPolarConverter.hpp`.

8.43.1.2 `#define ERR_IN_EXTRA_DATA "The input file contains more data than required."`

Definition at line 15 of file `CartesianToPolarConverter.hpp`.

8.43.1.3 `#define ERR_INPUT_OPEN "The input file could not be opened"`

Definition at line 14 of file `CartesianToPolarConverter.hpp`.

8.43.1.4 `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`

Definition at line 13 of file `CartesianToPolarConverter.hpp`.

Referenced by `multiscale::video::RectangularEntityCsvToInputFilesConverter::computeSimulationTime()`, `multiscale::video::RectangularCsvToInputFilesConverter::computeSimulationTime()`, `multiscale::video::PolarCsvToInputFilesConverter::computeSimulationTime()`, `multiscale::video::CartesianToConcentrationsConverter::readHeaderLine()`, and `multiscale::video::CartesianToPolarConverter::readHeaderLine()`.

8.43.1.5 `#define ERR_NONPOS_DIMENSION "The dimensions N and M must be positive."`

Definition at line 12 of file `CartesianToPolarConverter.hpp`.

Referenced by `multiscale::video::CartesianToConcentrationsConverter::readHeaderLine()`, and `multiscale::video::CartesianToPolarConverter::readHeaderLine()`.

8.43.1.6 `#define OUTPUT_FILE_EXTENSION ".out"`

Definition at line 17 of file `CartesianToPolarConverter.hpp`.

Referenced by `multiscale::video::CartesianToPolarConverter::outputResultsAsFile()`.

8.43.1.7 `#define RADIUS_MAX 0.3`

Definition at line 20 of file `CartesianToPolarConverter.hpp`.

Referenced by `multiscale::video::CartesianToPolarConverter::transformToAnnularSectors()`.

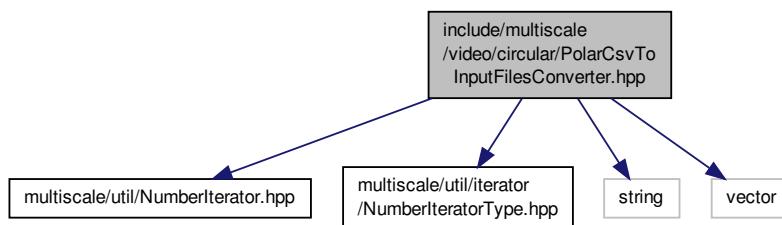
8.43.1.8 #define RADIUS_MIN 0.001

Definition at line 19 of file `CartesianToPolarConverter.hpp`.

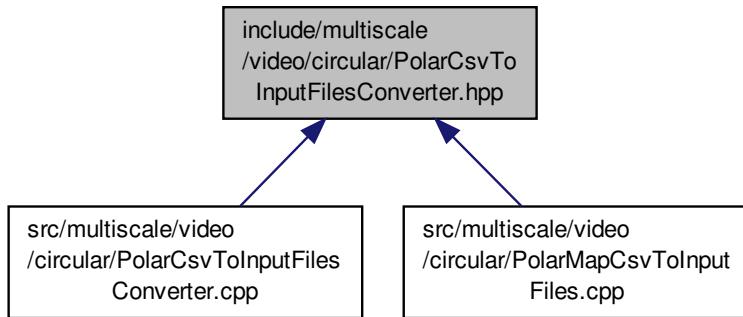
Referenced by `multiscale::video::CartesianToPolarConverter::transformToAnnularSectors()`.

8.44 include/multiscale/video/circular/PolarCsvToInputFilesConverter.hpp File Reference

```
#include "multiscale/util/NumberIterator.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <string>
#include <vector>
Include dependency graph for PolarCsvToInputFilesConverter.hpp:
```



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



Classes

- class [multiscale::video::PolarCsvToInputFilesConverter](#)
Csv file to input file converter considering polar coordinates.

Namespaces

- namespace [multiscale](#)

- namespace `multiscale::video`

Macros

- `#define OUTPUT_EXTENSION ".in"`
- `#define OUTPUT_SEPARATOR " "`
- `#define OUTPUT_FILE_SEPARATOR "_"`
- `#define INPUT_FILE_SEPARATOR ","`
- `#define RADIUS_MIN 1`
- `#define ERR_NEG_CONCENTRATION "All concentrations must be non-negative."`
- `#define ERR_SELECTED_CONCENTRATION_INDEX "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."`
- `#define ERR_NR_CONCENTRATIONS "The number of concentrations in the input file does not match the values of the input parameters height and width."`
- `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`
- `#define ERR_INPUT_OPEN "The input file could not be opened."`
- `#define ERR_INVALID_VALUE_LINE "Invalid value on line: "`
- `#define ERR_INVALID_VALUE_TOKEN ", value: "`

8.44.1 Macro Definition Documentation

8.44.1.1 `#define ERR_INPUT_OPEN "The input file could not be opened."`

Definition at line 21 of file PolarCsvToInputFilesConverter.hpp.

8.44.1.2 `#define ERR_INVALID_VALUE_LINE "Invalid value on line: "`

Definition at line 22 of file PolarCsvToInputFilesConverter.hpp.

Referenced by `multiscale::video::RectangularEntityCsvToInputFilesConverter::validateCoordinate()`, `multiscale::video::RectangularCsvToInputFilesConverter::validateInputLine()`, `multiscale::video::PolarCsvToInputFilesConverter::validateInputLine()`, and `multiscale::video::RectangularEntityCsvToInputFilesConverter::validateSimulationTime()`.

8.44.1.3 `#define ERR_INVALID_VALUE_TOKEN ", value: "`

Definition at line 23 of file PolarCsvToInputFilesConverter.hpp.

Referenced by `multiscale::video::RectangularEntityCsvToInputFilesConverter::validateCoordinate()`, `multiscale::video::RectangularCsvToInputFilesConverter::validateInputLine()`, `multiscale::video::PolarCsvToInputFilesConverter::validateInputLine()`, and `multiscale::video::RectangularEntityCsvToInputFilesConverter::validateSimulationTime()`.

8.44.1.4 `#define ERR_NEG_CONCENTRATION "All concentrations must be non-negative."`

Definition at line 17 of file PolarCsvToInputFilesConverter.hpp.

8.44.1.5 `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`

Definition at line 20 of file PolarCsvToInputFilesConverter.hpp.

8.44.1.6 #define ERR_NR_CONCENTRATIONS "The number of concentrations in the input file does not match the values of the input parameters height and width."

Definition at line 19 of file PolarCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularCsvToInputFilesConverter::validateInputLine(), and multiscale::video::PolarCsvToInputFilesConverter::validateInputLine().

8.44.1.7 #define ERR_SELECTED_CONCENTRATION_INDEX "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."

Definition at line 18 of file PolarCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularCsvToInputFilesConverter::validateSelectedConcentrationIndex(), and multiscale::video::PolarCsvToInputFilesConverter::validateSelectedConcentrationIndex().

8.44.1.8 #define INPUT_FILE_SEPARATOR ", "

Definition at line 13 of file PolarCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::video::PolarCsvToInputFilesConverter::splitLineInConcentrations(), multiscale::video::RectangularEntityCsvToInputFilesConverter::splitLineInCoordinates(), multiscale::video::RectangularEntityCsvToInputFilesConverter::validateInputLine(), multiscale::video::RectangularCsvToInputFilesConverter::validateInputLine(), and multiscale::video::PolarCsvToInputFilesConverter::validateInputLine().

8.44.1.9 #define OUTPUT_EXTENSION ".in"

Definition at line 10 of file PolarCsvToInputFilesConverter.hpp.

8.44.1.10 #define OUTPUT_FILE_SEPARATOR "_"

Definition at line 12 of file PolarCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::initOutputFile(), multiscale::video::RectangularCsvToInputFilesConverter::initOutputFile(), and multiscale::video::PolarCsvToInputFilesConverter::initOutputFile().

8.44.1.11 #define OUTPUT_SEPARATOR " "

Definition at line 11 of file PolarCsvToInputFilesConverter.hpp.

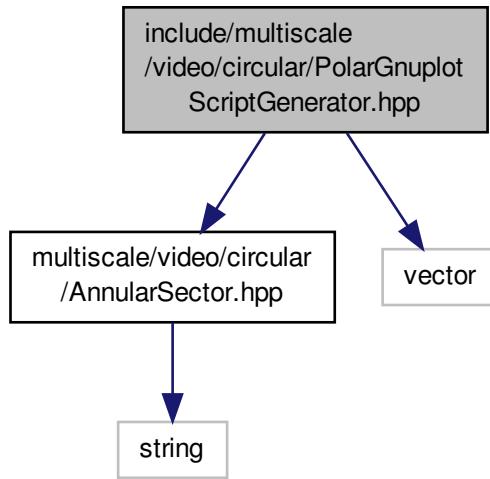
8.44.1.12 #define RADIUS_MIN 1

Definition at line 15 of file PolarCsvToInputFilesConverter.hpp.

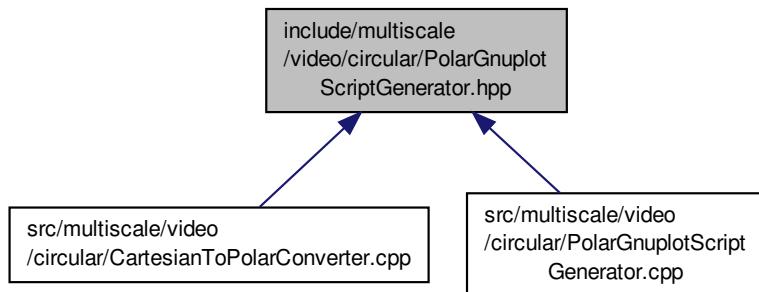
8.45 include/multiscale/video/circular/PolarGnuplotScriptGenerator.hpp File Reference

```
#include "multiscale/video/circular/AnnularSector.hpp"
#include <vector>
```

Include dependency graph for PolarGnuplotScriptGenerator.hpp:



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



Classes

- class [multiscale::video::PolarGnuplotScriptGenerator](#)
Gnuplot script generator from the provided annular sectors.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::video](#)

Macros

- `#define HEADER_IN "config/video/circular/header.in"`
- `#define CONTENT_IN "config/video/circular/content.in"`
- `#define FOOTER_IN "config/video/circular/footer.in"`
- `#define REPLACE_HEADER_FILENAME "OUTPUT_FILENAME"`
- `#define REPLACE_HEADER_SIM_TIME "OUTPUT_SIM_TIME"`
- `#define REPLACE_CONTENT_INDEX "OBJ_INDEX"`
- `#define REPLACE_CONTENT_RADIUS "OBJ_END_RADIUS"`
- `#define REPLACE_CONTENT_START_ANGLE "OBJ_START_ANGLE"`
- `#define REPLACE_CONTENT_END_ANGLE "OBJ_END_ANGLE"`
- `#define REPLACE_CONTENT_CONCENTRATION "OBJ_CONCENTRATION"`
- `#define GNUPLOT_EXTENSION ".plt"`

8.45.1 Macro Definition Documentation

8.45.1.1 `#define CONTENT_IN "config/video/circular/content.in"`

Definition at line 11 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::generateBody(), and multiscale::video::RectangularGnuplotScriptGenerator::generateBody().

8.45.1.2 `#define FOOTER_IN "config/video/circular/footer.in"`

Definition at line 12 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::generateFooter(), and multiscale::video::RectangularGnuplotScriptGenerator::generateFooter().

8.45.1.3 `#define GNUPLOT_EXTENSION ".plt"`

Definition at line 23 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::generateScript(), and multiscale::video::RectangularGnuplotScriptGenerator::generateScript().

8.45.1.4 `#define HEADER_IN "config/video/circular/header.in"`

Definition at line 10 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::generateHeader(), and multiscale::video::RectangularGnuplotScriptGenerator::generateHeader().

8.45.1.5 `#define REPLACE_CONTENT_CONCENTRATION "OBJ_CONCENTRATION"`

Definition at line 21 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent().

8.45.1.6 `#define REPLACE_CONTENT_END_ANGLE "OBJ_END_ANGLE"`

Definition at line 20 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent().

8.45.1.7 `#define REPLACE_CONTENT_INDEX "OBJ_INDEX"`

Definition at line 17 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent().

8.45.1.8 `#define REPLACE_CONTENT_RADIUS "OBJ_END_RADIUS"`

Definition at line 18 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent().

8.45.1.9 `#define REPLACE_CONTENT_START_ANGLE "OBJ_START_ANGLE"`

Definition at line 19 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputContent().

8.45.1.10 `#define REPLACE_HEADER_FILENAME "OUTPUT_FILENAME"`

Definition at line 14 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputHeader(), and multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

8.45.1.11 `#define REPLACE_HEADER_SIM_TIME "OUTPUT_SIM_TIME"`

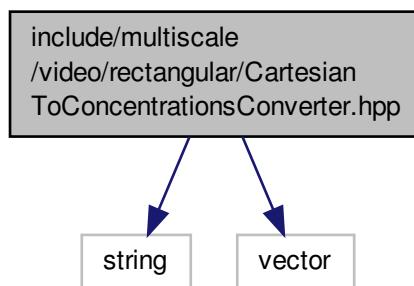
Definition at line 15 of file PolarGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::PolarGnuplotScriptGenerator::outputHeader(), and multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

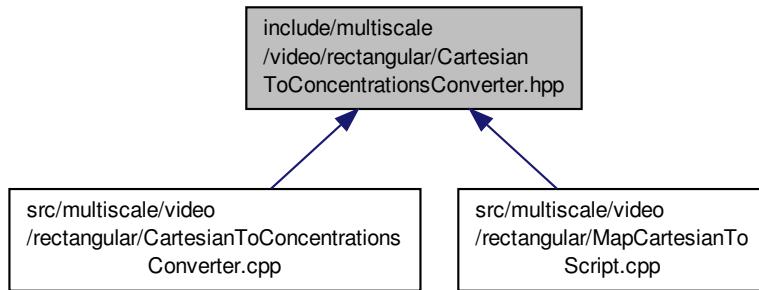
8.46 include/multiscale/video/rectangular/CartesianToConcentrationsConverter.hpp File Reference

```
#include <string>
#include <vector>
```

Include dependency graph for CartesianToConcentrationsConverter.hpp:



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



Classes

- class [multiscale::video::CartesianToConcentrationsConverter](#)
Scale the values of the rectangular geometry grid cells.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::video](#)

Macros

- #define [ERR_CONC](#) "All concentrations have to be between 0 and 1."
- #define [ERR_NONPOS_DIMENSION](#) "The dimensions N and M must be positive."
- #define [ERR_NEG_SIM_TIME](#) "The simulation time must be non-negative."
- #define [ERR_INPUT_OPEN](#) "The input file could not be opened"
- #define [ERR_IN_EXTRA_DATA](#) "The input file contains more data than required."
- #define [OUTPUT_FILE_EXTENSION](#) ".out"
- #define [RADIUS_MIN](#) 0.001
- #define [RADIUS_MAX](#) 0.3

8.46.1 Macro Definition Documentation

8.46.1.1 #define ERR_CONC "All concentrations have to be between 0 and 1."

Definition at line 9 of file `CartesianToConcentrationsConverter.hpp`.

8.46.1.2 #define ERR_IN_EXTRA_DATA "The input file contains more data than required."

Definition at line 13 of file `CartesianToConcentrationsConverter.hpp`.

8.46.1.3 #define ERR_INPUT_OPEN "The input file could not be opened"

Definition at line 12 of file `CartesianToConcentrationsConverter.hpp`.

8.46.1.4 #define ERR_NEG_SIM_TIME "The simulation time must be non-negative."

Definition at line 11 of file CartesianToConcentrationsConverter.hpp.

8.46.1.5 #define ERR_NONPOS_DIMENSION "The dimensions N and M must be positive."

Definition at line 10 of file CartesianToConcentrationsConverter.hpp.

8.46.1.6 #define OUTPUT_FILE_EXTENSION ".out"

Definition at line 15 of file CartesianToConcentrationsConverter.hpp.

8.46.1.7 #define RADIUS_MAX 0.3

Definition at line 18 of file CartesianToConcentrationsConverter.hpp.

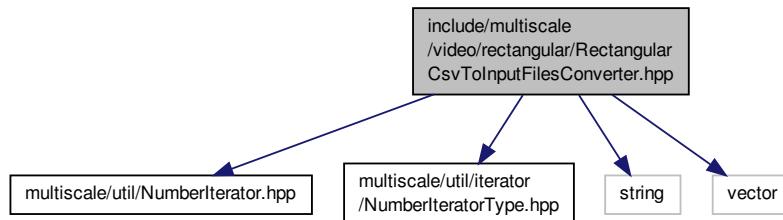
8.46.1.8 #define RADIUS_MIN 0.001

Definition at line 17 of file CartesianToConcentrationsConverter.hpp.

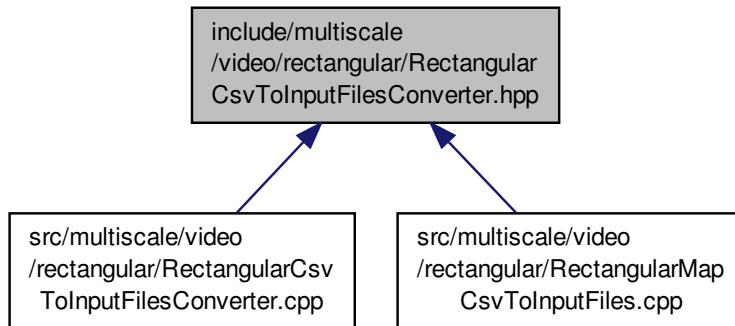
8.47 include/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.hpp File Reference

```
#include "multiscale/util/NumberIterator.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <string>
#include <vector>
```

Include dependency graph for RectangularCsvToInputFilesConverter.hpp:



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



Classes

- class [multiscale::video::RectangularCsvToInputFilesConverter](#)
Csv file to input file converter considering cartesian coordinates.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::video](#)

Macros

- `#define OUTPUT_EXTENSION ".in"`
- `#define OUTPUT_SEPARATOR " "`
- `#define OUTPUT_FILE_SEPARATOR "_"`
- `#define INPUT_FILE_SEPARATOR ","`
- `#define ERR_NEG_CONCENTRATION "All concentrations must be non-negative."`
- `#define ERR_SELECTED_CONCENTRATION_INDEX "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."`
- `#define ERR_NR_CONCENTRATIONS "The number of concentrations in the input file does not match the values of the input parameters height and width."`
- `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`
- `#define ERR_INPUT_OPEN "The input file could not be opened."`
- `#define ERR_INVALID_VALUE_LINE "Invalid value on line: "`
- `#define ERR_INVALID_VALUE_TOKEN ", value: "`

8.47.1 Macro Definition Documentation

8.47.1.1 `#define ERR_INPUT_OPEN "The input file could not be opened."`

Definition at line 19 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.2 `#define ERR_INVALID_VALUE_LINE "Invalid value on line: "`

Definition at line 20 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.3 `#define ERR_INVALID_VALUE_TOKEN ", value: "`

Definition at line 21 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.4 `#define ERR_NEG_CONCENTRATION "All concentrations must be non-negative."`

Definition at line 15 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.5 `#define ERR_NEG_SIM_TIME "The simulation time must be non-negative."`

Definition at line 18 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.6 `#define ERR_NR_CONCENTRATIONS "The number of concentrations in the input file does not match the values of the input parameters height and width."`

Definition at line 17 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.7 `#define ERR_SELECTED_CONCENTRATION_INDEX "The selected concentration index (0-based indexing) should be smaller than the number of concentrations."`

Definition at line 16 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.8 `#define INPUT_FILE_SEPARATOR ","`

Definition at line 13 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.9 `#define OUTPUT_EXTENSION ".in"`

Definition at line 10 of file RectangularCsvToInputFilesConverter.hpp.

8.47.1.10 `#define OUTPUT_FILE_SEPARATOR "_"`

Definition at line 12 of file RectangularCsvToInputFilesConverter.hpp.

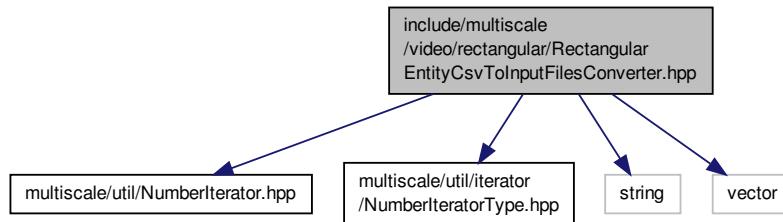
8.47.1.11 `#define OUTPUT_SEPARATOR " "`

Definition at line 11 of file RectangularCsvToInputFilesConverter.hpp.

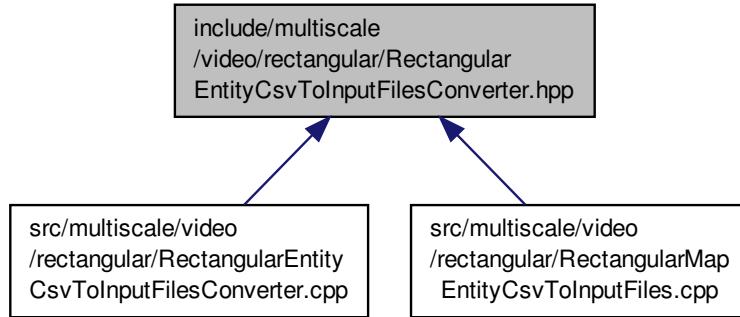
8.48 include/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.hpp File Reference

```
#include "multiscale/util/NumberIterator.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <string>
#include <vector>
```

Include dependency graph for `RectangularEntityCsvToInputFilesConverter.hpp`:



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



Classes

- class `multiscale::video::RectangularEntityCsvToInputFilesConverter`
Csv entity file to input file converter considering cartesian coordinates.

Namespaces

- namespace `multiscale`
- namespace `multiscale::video`

Macros

- `#define OUTPUT_EXTENSION ".in"`
- `#define OUTPUT_SEPARATOR " "`
- `#define OUTPUT_FILE_SEPARATOR "_"`
- `#define INPUT_FILE_SEPARATOR ","`
- `#define ERR_INVALID_NR_ENTITIES "The number of entities at the given position is invalid."`
- `#define ERR_INVALID_OX_COORDINATE "The value of the Ox coordinate is invalid."`
- `#define ERR_INVALID_OY_COORDINATE "The value of the Oy coordinate is invalid."`

- `#define ERR_MAX_NR_ENTITIES` "The maximum number of entities per grid position is equal to zero."
- `#define ERR_NR_COORDINATES` "The number of coordinates in the input file does not match the values of the input parameters height, width and nrOfEntities."
- `#define ERR_NEG_SIM_TIME` "The simulation time must be non-negative."
- `#define ERR_INPUT_OPEN` "The input file could not be opened."
- `#define ERR_INVALID_VALUE_LINE` "Invalid value on line: "
- `#define ERR_INVALID_VALUE_TOKEN` ", value: "

8.48.1 Macro Definition Documentation

8.48.1.1 `#define ERR_INPUT_OPEN` "The input file could not be opened."

Definition at line 21 of file RectangularEntityCsvToInputFilesConverter.hpp.

8.48.1.2 `#define ERR_INVALID_NR_ENTITIES` "The number of entities at the given position is invalid."

Definition at line 15 of file RectangularEntityCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::validateEntitiesGrid().

8.48.1.3 `#define ERR_INVALID_OX_COORDINATE` "The value of the Ox coordinate is invalid."

Definition at line 16 of file RectangularEntityCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::computeCoordinate().

8.48.1.4 `#define ERR_INVALID_OY_COORDINATE` "The value of the Oy coordinate is invalid."

Definition at line 17 of file RectangularEntityCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::computeCoordinate().

8.48.1.5 `#define ERR_INVALID_VALUE_LINE` "Invalid value on line: "

Definition at line 22 of file RectangularEntityCsvToInputFilesConverter.hpp.

8.48.1.6 `#define ERR_INVALID_VALUE_TOKEN` ", value: "

Definition at line 23 of file RectangularEntityCsvToInputFilesConverter.hpp.

8.48.1.7 `#define ERR_MAX_NR_ENTITIES` "The maximum number of entities per grid position is equal to zero."

Definition at line 18 of file RectangularEntityCsvToInputFilesConverter.hpp.

Referenced by multiscale::video::RectangularEntityCsvToInputFilesConverter::validateMaxNrOfEntitiesPerPosition().

8.48.1.8 `#define ERR_NEG_SIM_TIME` "The simulation time must be non-negative."

Definition at line 20 of file RectangularEntityCsvToInputFilesConverter.hpp.

8.48.1.9 `#define ERR_NR_COORDINATES "The number of coordinates in the input file does not match the values of the input parameters height, width and nrOfEntities."`

Definition at line 19 of file `RectangularEntityCsvToInputFilesConverter.hpp`.

Referenced by `multiscale::video::RectangularEntityCsvToInputFilesConverter::validateInputLine()`.

8.48.1.10 `#define INPUT_FILE_SEPARATOR ","`

Definition at line 13 of file `RectangularEntityCsvToInputFilesConverter.hpp`.

8.48.1.11 `#define OUTPUT_EXTENSION ".in"`

Definition at line 10 of file `RectangularEntityCsvToInputFilesConverter.hpp`.

8.48.1.12 `#define OUTPUT_FILE_SEPARATOR "_"`

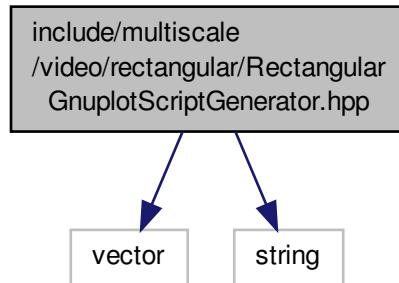
Definition at line 12 of file `RectangularEntityCsvToInputFilesConverter.hpp`.

8.48.1.13 `#define OUTPUT_SEPARATOR " "`

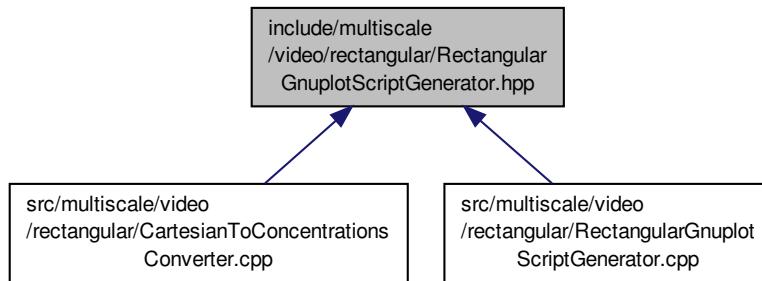
Definition at line 11 of file `RectangularEntityCsvToInputFilesConverter.hpp`.

8.49 include/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.hpp File Reference

```
#include <vector>
#include <string>
Include dependency graph for RectangularGnuplotScriptGenerator.hpp:
```



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



Classes

- class [multiscale::video::RectangularGnuplotScriptGenerator](#)

Gnuplot script generator from the provided concentrations considering a rectangular geometry.

Namespaces

- namespace [multiscale](#)
- namespace [multiscale::video](#)

Macros

- #define HEADER_IN "config/video/rectangular/header.in"
- #define CONTENT_IN "config/video/rectangular/content.in"
- #define FOOTER_IN "config/video/rectangular/footer.in"
- #define REPLACE_HEADER_FILENAME "OUTPUT_FILENAME"
- #define REPLACE_HEADER_HEIGHT "OUTPUT_DIMENSION1"
- #define REPLACE_HEADER_WIDTH "OUTPUT_DIMENSION2"
- #define REPLACE_HEADER_SIM_TIME "OUTPUT_SIM_TIME"
- #define REPLACE_DIMENSION_EXTRA 0.5
- #define OUTPUT_SEPARATOR " "
- #define GNUPLOT_EXTENSION ".plt"

8.49.1 Macro Definition Documentation

8.49.1.1 #define CONTENT_IN "config/video/rectangular/content.in"

Definition at line 10 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.2 #define FOOTER_IN "config/video/rectangular/footer.in"

Definition at line 11 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.3 #define GNUPLOT_EXTENSION ".plt"

Definition at line 22 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.4 #define HEADER_IN "config/video/rectangular/header.in"

Definition at line 9 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.5 #define OUTPUT_SEPARATOR " "

Definition at line 20 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.6 #define REPLACE_DIMENSION_EXTRA 0.5

Definition at line 18 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

8.49.1.7 #define REPLACE_HEADER_FILENAME "OUTPUT_FILENAME"

Definition at line 13 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.8 #define REPLACE_HEADER_HEIGHT "OUTPUT_DIMENSION1"

Definition at line 14 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

8.49.1.9 #define REPLACE_HEADER_SIM_TIME "OUTPUT_SIM_TIME"

Definition at line 16 of file RectangularGnuplotScriptGenerator.hpp.

8.49.1.10 #define REPLACE_HEADER_WIDTH "OUTPUT_DIMENSION2"

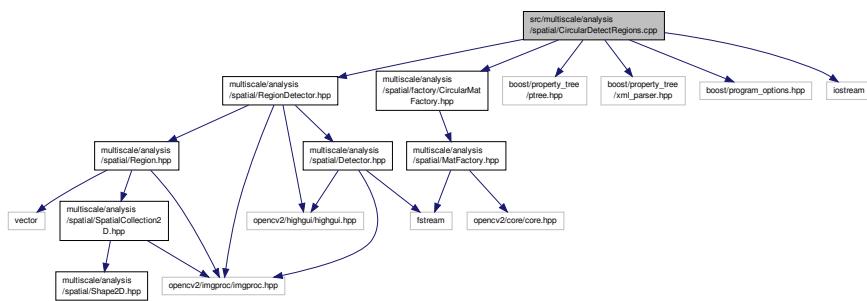
Definition at line 15 of file RectangularGnuplotScriptGenerator.hpp.

Referenced by multiscale::video::RectangularGnuplotScriptGenerator::outputHeader().

8.50 src/multiscale/analysis/spatial/CircularDetectRegions.cpp File Reference

```
#include "multiscale/analysis/spatial/RegionDetector.hpp"
#include "multiscale/analysis/spatial/factory/CircularMatFactory.hpp"
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/xml_parser.hpp>
#include <boost/program_options.hpp>
#include <iostream>
```

Include dependency graph for CircularDetectRegions.cpp:



Macros

- #define CONFIG_FILE "config/analysis/spatial/circular_region_detector.xml"
- #define LABEL_ROOT_COMMENT "<xmlcomment>"
- #define LABEL_ALPHA "detector.alpha"
- #define LABEL_BETA "detector.beta"
- #define LABEL_BLUR_KERNEL_SIZE "detector.blurKernelSize"
- #define LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS "detector.morphologicalCloseIterations"
- #define LABEL_EPSILON "detector.epsilon"
- #define LABEL_REGION_AREA_THRESH "detector.regionAreaThresh"
- #define LABEL_THRESHOLD_VALUE "detector.thresholdValue"
- #define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."
- #define ERR_CODE 1
- #define ERR_MSG "An error occurred: "

Functions

- po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)
- void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)
- void printWrongParameters ()
- bool areValidParameters (string &inputFilepath, string &outputFilename, bool &debugFlag, int argc, char **argv)
- void loadDetectorParameterValues (RegionDetector &detector)
- void saveDetectorParameterValues (RegionDetector &detector)
- void loadDetectorParameterValues (RegionDetector &detector, bool debugMode)
- void saveDetectorParameterValues (RegionDetector &detector, bool debugMode)
- int main (int argc, char **argv)

8.50.1 Macro Definition Documentation

8.50.1.1 #define CONFIG_FILE "config/analysis/spatial/circular_region_detector.xml"

Definition at line 30 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.2 #define ERR_CODE 1

Definition at line 43 of file CircularDetectRegions.cpp.

Referenced by main().

8.50.1.3 #define ERR_MSG "An error occurred: "

Definition at line 44 of file CircularDetectRegions.cpp.

Referenced by main(), and printWrongParameters().

8.50.1.4 #define LABEL_ALPHA "detector.alpha"

Definition at line 33 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.5 #define LABEL_BETA "detector.beta"

Definition at line 34 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.6 #define LABEL_BLUR_KERNEL_SIZE "detector.blurKernelSize"

Definition at line 35 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.7 #define LABEL_EPSILON "detector.epsilon"

Definition at line 37 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.8 #define LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS "detector.morphologicalCloseIterations"

Definition at line 36 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.9 #define LABEL_REGION_AREA_THRESH "detector.regionAreaThresh"

Definition at line 38 of file CircularDetectRegions.cpp.

Referenced by loadDetectorParameterValues(), and saveDetectorParameterValues().

8.50.1.10 #define LABEL_ROOT_COMMENT "<xmlcomment>"

Definition at line 32 of file CircularDetectRegions.cpp.

Referenced by saveDetectorParameterValues().

8.50.1.11 `#define LABEL_THRESHOLD_VALUE "detector.thresholdValue"`

Definition at line 39 of file CircularDetectRegions.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.50.1.12 `#define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."`

Definition at line 41 of file CircularDetectRegions.cpp.

Referenced by `saveDetectorParameterValues()`.

8.50.2 Function Documentation

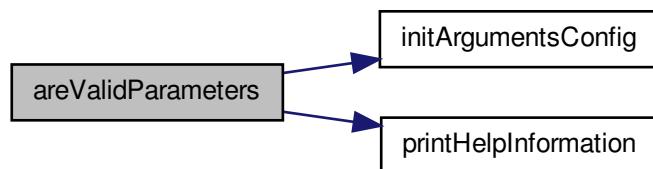
8.50.2.1 `bool areValidParameters (string & inputFilepath, string & outputFilename, bool & debugFlag, int argc, char ** argv)`

Definition at line 72 of file CircularDetectRegions.cpp.

References `initArgumentsConfig()`, and `printHelpInformation()`.

Referenced by `main()`.

Here is the call graph for this function:



Here is the caller graph for this function:

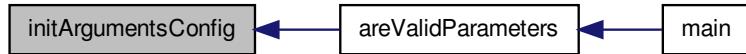


8.50.2.2 `po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)`

Definition at line 48 of file CircularDetectRegions.cpp.

Referenced by `areValidParameters()`.

Here is the caller graph for this function:



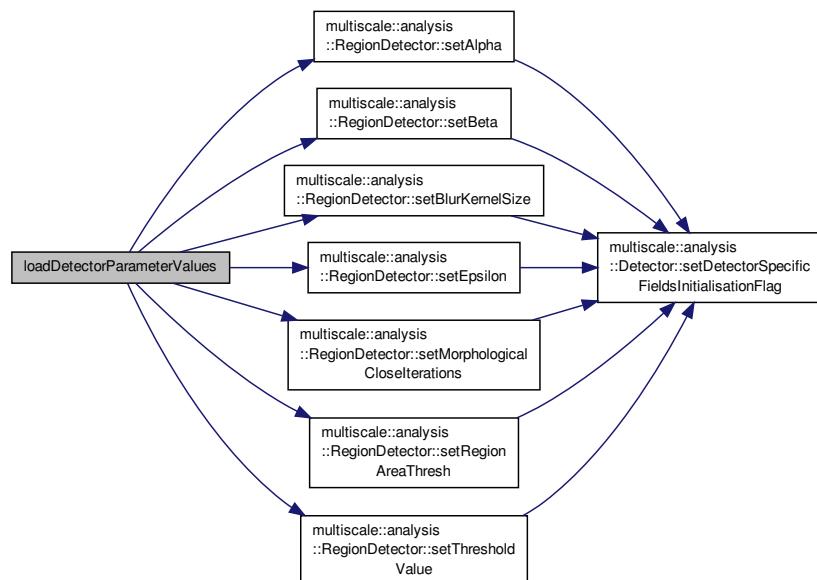
8.50.2.3 void loadDetectorParameterValues (RegionDetector & detector)

Definition at line 100 of file CircularDetectRegions.cpp.

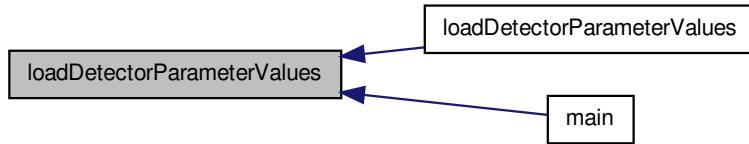
References CONFIG_FILE, LABEL_ALPHA, LABEL_BETA, LABEL_BLUR_KERNEL_SIZE, LABEL_EPSILON, LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS, LABEL_REGION_AREA_THRESH, LABEL_THRESHOLD_VALUE, multiscale::analysis::RegionDetector::setAlpha(), multiscale::analysis::RegionDetector::setBeta(), multiscale::analysis::RegionDetector::setBlurKernelSize(), multiscale::analysis::RegionDetector::setEpsilon(), multiscale::analysis::RegionDetector::setMorphologicalCloseliterations(), multiscale::analysis::RegionDetector::setRegionAreaThresh(), and multiscale::analysis::RegionDetector::setThresholdValue().

Referenced by loadDetectorParameterValues(), and main().

Here is the call graph for this function:



Here is the caller graph for this function:

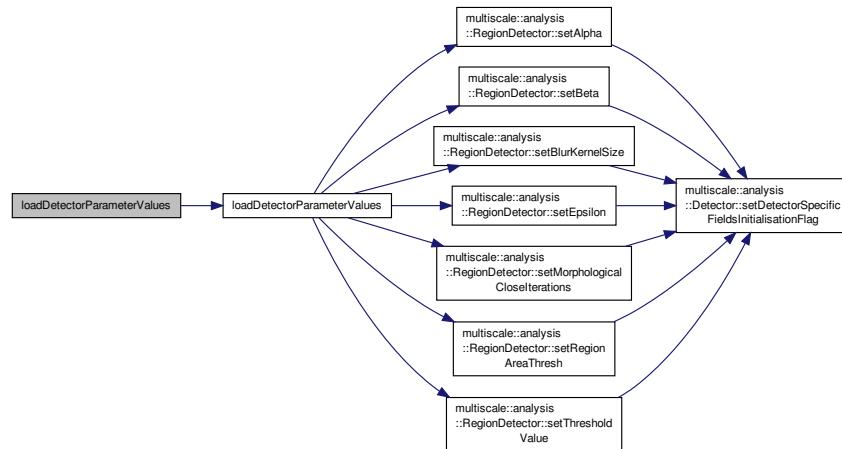


8.50.2.4 void loadDetectorParameterValues (RegionDetector & detector, bool debugMode)

Definition at line 134 of file CircularDetectRegions.cpp.

References loadDetectorParameterValues().

Here is the call graph for this function:

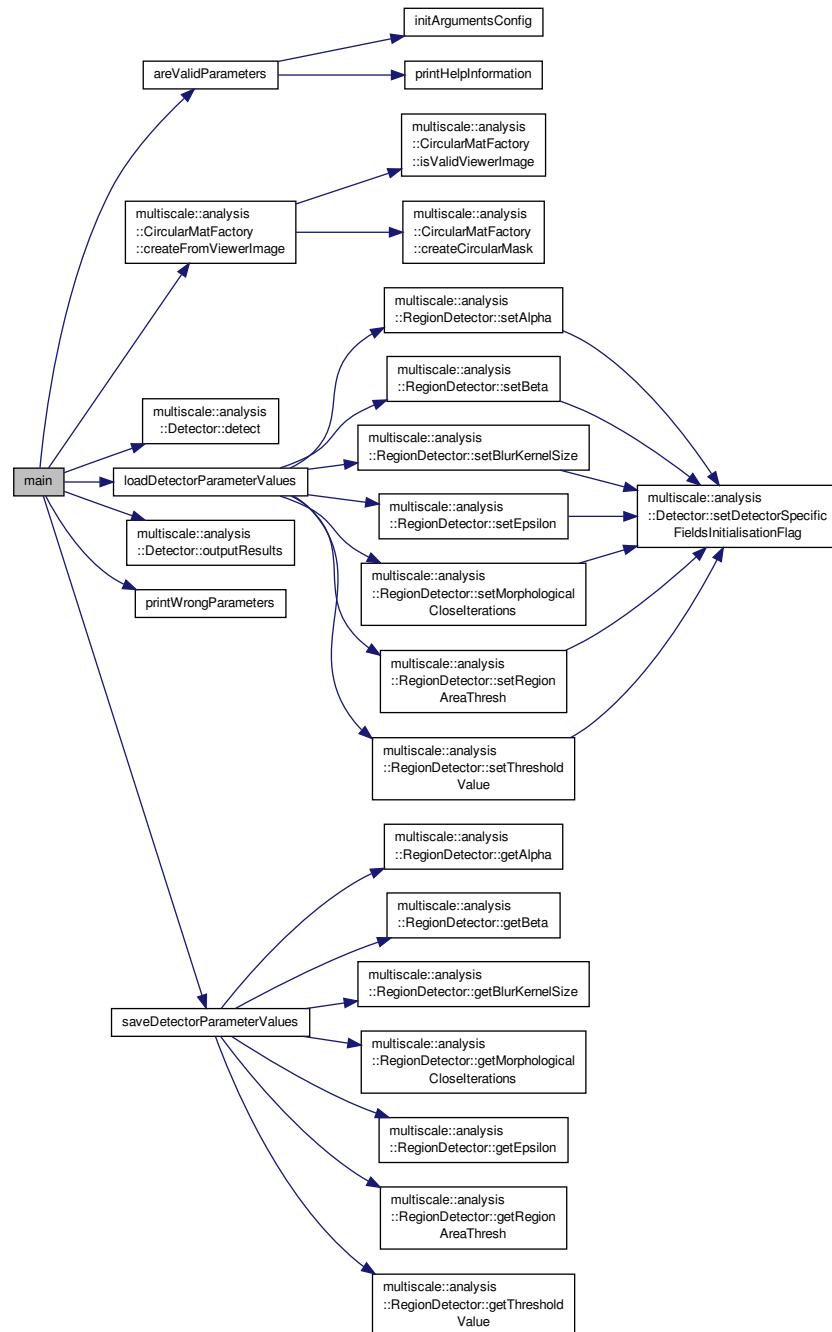


8.50.2.5 int main (int argc, char ** argv)

Definition at line 146 of file CircularDetectRegions.cpp.

References areValidParameters(), multiscale::analysis::CircularMatFactory::createFromViewerImage(), multiscale::analysis::Detector::detect(), ERR_CODE, ERR_MSG, loadDetectorParameterValues(), multiscale::analysis::Detector::outputResults(), printWrongParameters(), and saveDetectorParameterValues().

Here is the call graph for this function:

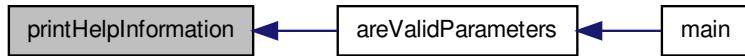


8.50.2.6 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 61 of file CircularDetectRegions.cpp.

Referenced by areValidParameters().

Here is the caller graph for this function:



8.50.2.7 void printWrongParameters ()

Definition at line 66 of file CircularDetectRegions.cpp.

References ERR_MSG.

Referenced by main().

Here is the caller graph for this function:



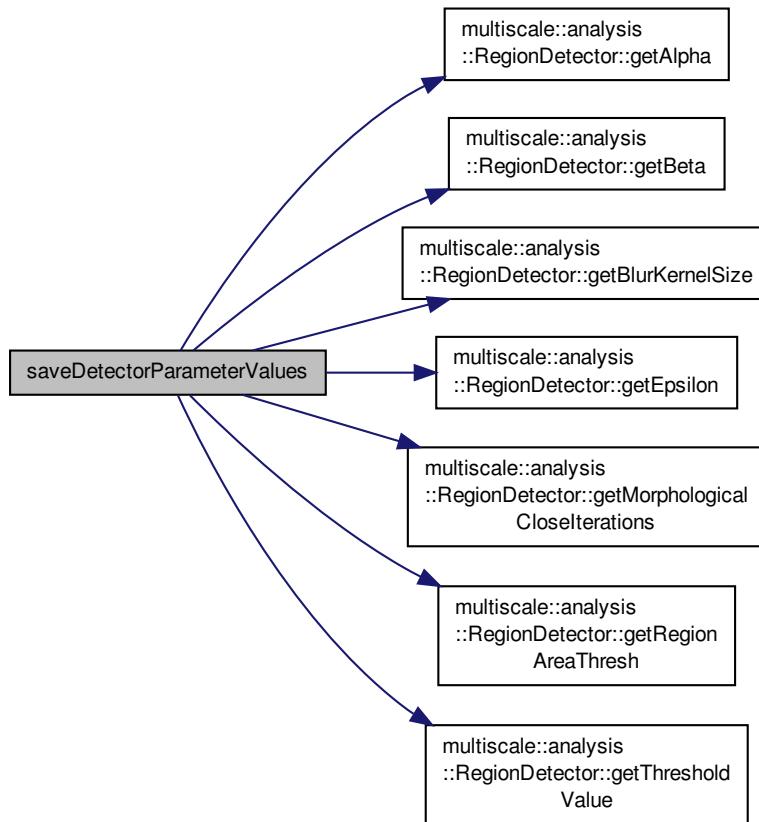
8.50.2.8 void saveDetectorParameterValues (RegionDetector & detector)

Definition at line 115 of file CircularDetectRegions.cpp.

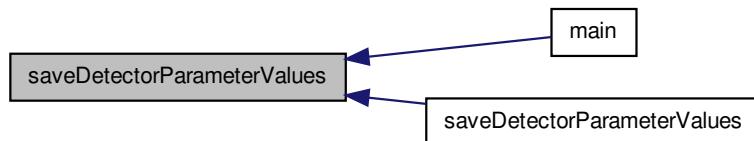
References CONFIG_FILE, multiscale::analysis::RegionDetector::getAlpha(), multiscale::analysis::RegionDetector::getBeta(), multiscale::analysis::RegionDetector::getBlurKernelSize(), multiscale::analysis::RegionDetector::getEpsilon(), multiscale::analysis::RegionDetector::getMorphologicalCloseIterations(), multiscale::analysis::RegionDetector::getRegionAreaThresh(), multiscale::analysis::RegionDetector::getThresholdValue(), LABEL_ALPHA, LABEL_BETA, LABEL_BLUR_KERNEL_SIZE, LABEL_EPSILON, LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS, LABEL_REGION_AREA_THRESH, LABEL_ROOT_COMMENT, LABEL_THRESHOLD_VALUE, and ROOT_COMMENT.

Referenced by main(), and saveDetectorParameterValues().

Here is the call graph for this function:



Here is the caller graph for this function:

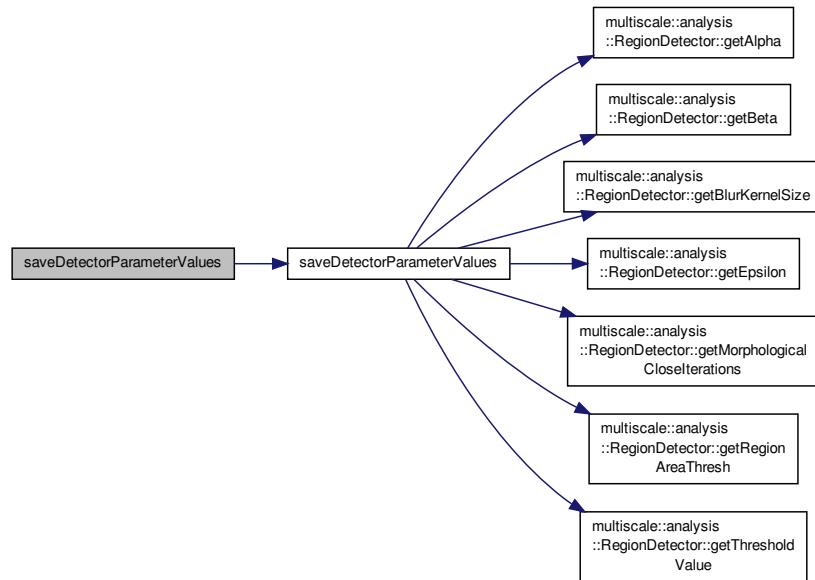


8.50.2.9 void saveDetectorParameterValues (`RegionDetector & detector, bool debugMode`)

Definition at line 139 of file CircularDetectRegions.cpp.

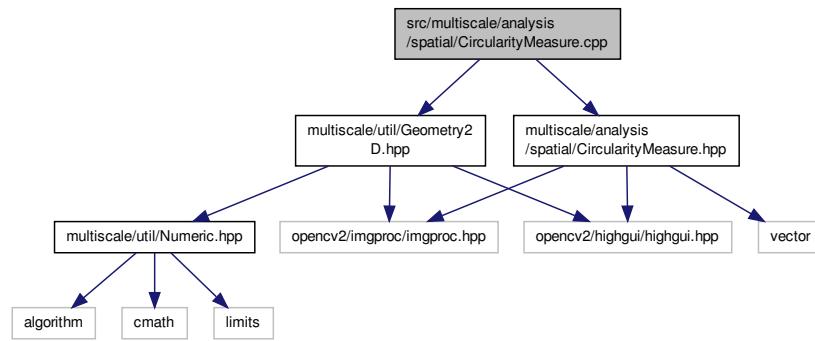
References `saveDetectorParameterValues()`.

Here is the call graph for this function:



8.51 src/multiscale/analysis/spatial/CircularityMeasure.cpp File Reference

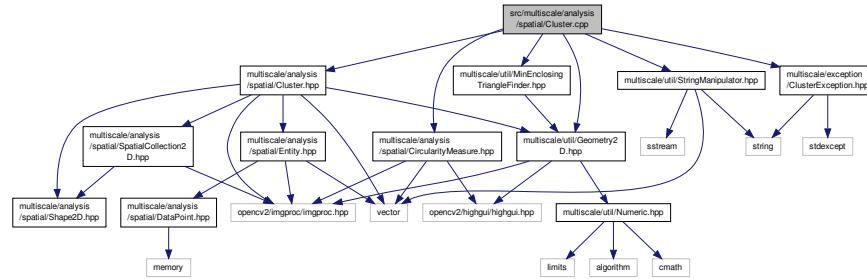
```
#include "multiscale/analysis/spatial/CircularityMeasure.hpp"
#include "multiscale/util/Geometry2D.hpp"
Include dependency graph for CircularityMeasure.cpp:
```



8.52 src/multiscale/analysis/spatial/Cluster.cpp File Reference

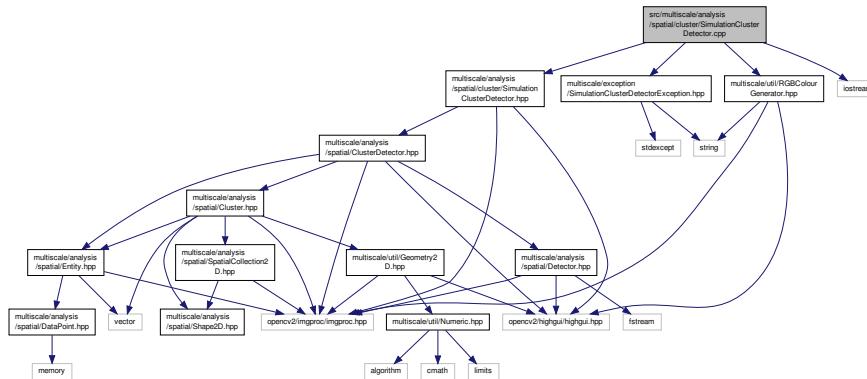
```
#include "multiscale/analysis/spatial/CircularityMeasure.hpp"
```

```
#include "multiscale/analysis/spatial/Cluster.hpp"
#include "multiscale/exception/ClusterException.hpp"
#include "multiscale/util/Geometry2D.hpp"
#include "multiscale/util/MinEnclosingTriangleFinder.hpp"
#include "multiscale/util/StringManipulator.hpp"
Include dependency graph for Cluster.cpp:
```



8.53 src/multiscale/analysis/spatial/cluster/SimulationClusterDetector.cpp File Reference

```
#include "multiscale/analysis/spatial/cluster/SimulationClusterDetector.hpp"
#include "multiscale/exception/SimulationClusterDetectorException.hpp"
#include "multiscale/util/RGBColourGenerator.hpp"
#include <iostream>
Include dependency graph for SimulationClusterDetector.cpp:
```

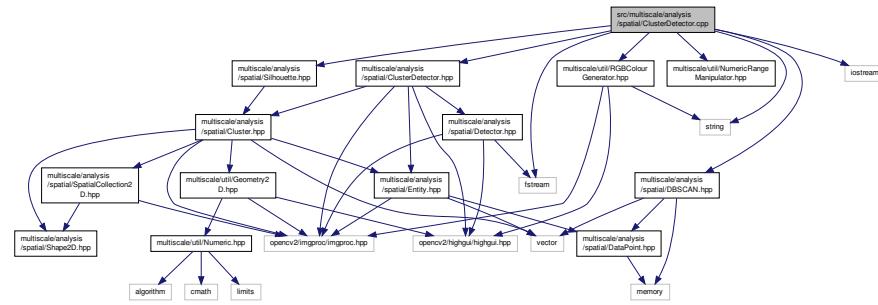


8.54 src/multiscale/analysis/spatial/ClusterDetector.cpp File Reference

```
#include "multiscale/analysis/spatial/ClusterDetector.hpp"
```

```
#include "multiscale/analysis/spatial/DBSCAN.hpp"
#include "multiscale/analysis/spatial/Silhouette.hpp"
#include "multiscale/util/NumericRangeManipulator.hpp"
#include "multiscale/util/RGBColourGenerator.hpp"
#include <iostream>
#include <fstream>
#include <string>
```

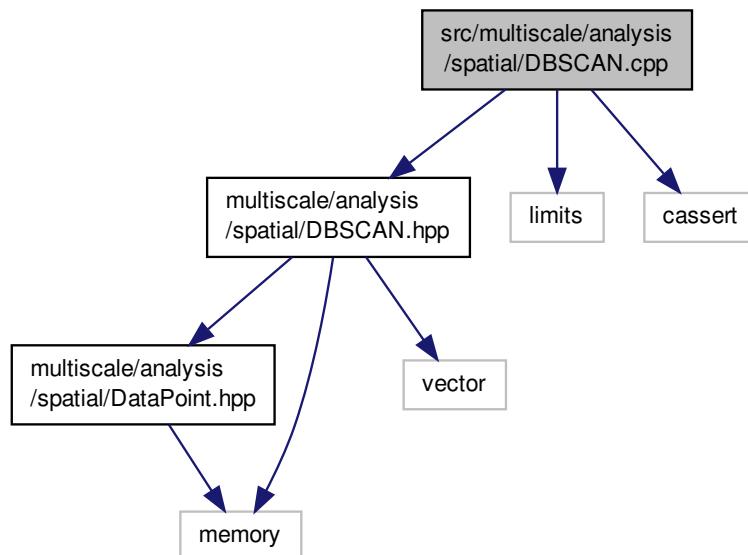
Include dependency graph for ClusterDetector.cpp:



8.55 src/multiscale/analysis/spatial/DBSCAN.cpp File Reference

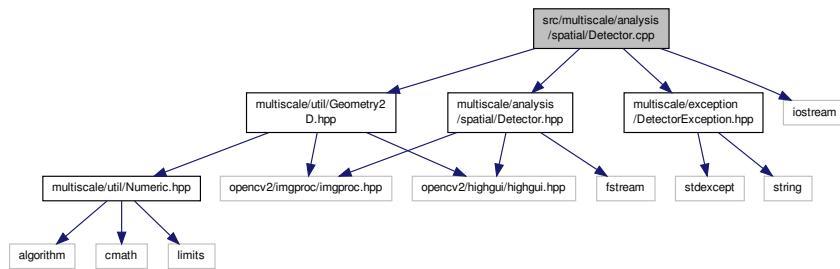
```
#include "multiscale/analysis/spatial/DBSCAN.hpp"
#include <limits>
#include <cassert>
```

Include dependency graph for DBSCAN.cpp:



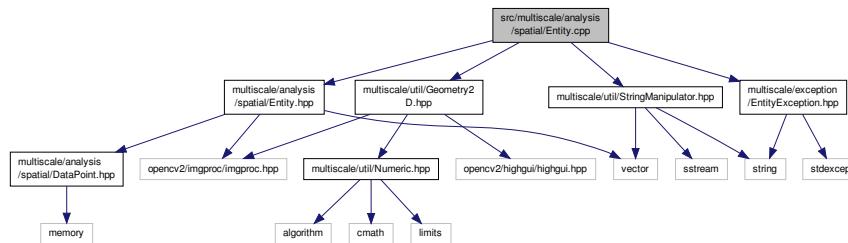
8.56 src/multiscale/analysis/spatial/Detector.cpp File Reference

```
#include "multiscale/analysis/spatial/Detector.hpp"
#include "multiscale/exception/DetectorException.hpp"
#include "multiscale/util/Geometry2D.hpp"
#include <iostream>
Include dependency graph for Detector.cpp:
```



8.57 src/multiscale/analysis/spatial/Entity.cpp File Reference

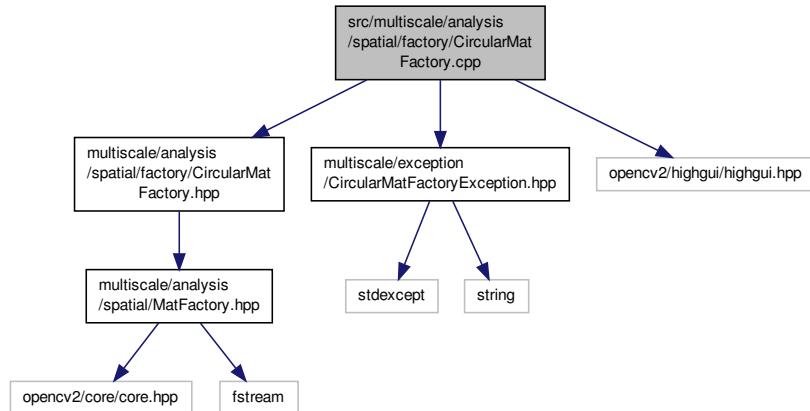
```
#include "multiscale/analysis/spatial/Entity.hpp"
#include "multiscale/exception/EntityException.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include "multiscale/util/Geometry2D.hpp"
Include dependency graph for Entity.cpp:
```



8.58 src/multiscale/analysis/spatial/factory/CircularMatFactory.cpp File Reference

```
#include "multiscale/analysis/spatial/factory/CircularMatFactory.hpp"
#include "multiscale/exception/CircularMatFactoryException.hpp"
#include "opencv2/highgui/highgui.hpp"
```

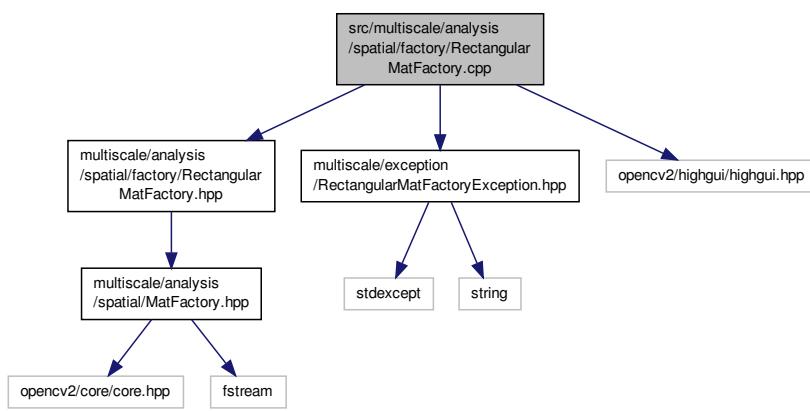
Include dependency graph for CircularMatFactory.cpp:



8.59 src/multiscale/analysis/spatial/factory/RectangularMatFactory.cpp File Reference

```
#include "multiscale/analysis/spatial/factory/RectangularMatFactory.hpp"
#include "multiscale/exception/RectangularMatFactoryException.hpp"
#include "opencv2/highgui/highgui.hpp"
```

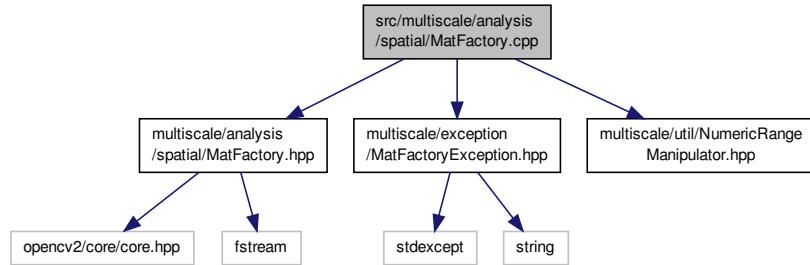
Include dependency graph for RectangularMatFactory.cpp:



8.60 src/multiscale/analysis/spatial/MatFactory.cpp File Reference

```
#include "multiscale/analysis/spatial/MatFactory.hpp"
#include "multiscale/exception/MatFactoryException.hpp"
#include "multiscale/util/NumericRangeManipulator.hpp"
```

Include dependency graph for MatFactory.cpp:

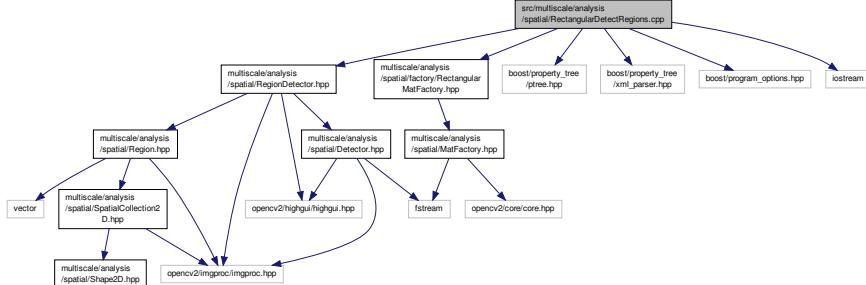


8.61 src/multiscale/analysis/spatial/RectangularDetectRegions.cpp File Reference

```

#include "multiscale/analysis/spatial/RegionDetector.hpp"
#include "multiscale/analysis/spatial/factory/RectangularMatFactory.hpp"
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/xml_parser.hpp>
#include <boost/program_options.hpp>
#include <iostream>
  
```

Include dependency graph for RectangularDetectRegions.cpp:



Macros

- #define CONFIG_FILE "config/analysis/spatial/rectangular_region_detector.xml"
- #define LABEL_ROOT_COMMENT "<xmlcomment>"
- #define LABEL_ALPHA "detector.alpha"
- #define LABEL_BETA "detector.beta"
- #define LABEL_BLUR_KERNEL_SIZE "detector.blurKernelSize"
- #define LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS "detector.morphologicalCloseIterations"
- #define LABEL_EPSILON "detector.epsilon"
- #define LABEL_REGION_AREA_THRESH "detector.regionAreaThresh"
- #define LABEL_THRESHOLD_VALUE "detector.thresholdValue"
- #define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."
- #define ERR_CODE 1
- #define ERR_MSG "An error occurred: "

Functions

- po::variables_map **initArgumentsConfig** (po::options_description &usageDescription, int argc, char **argv)
- void **printHelpInformation** (const po::variables_map &vm, const po::options_description &usageDescription)
- void **printWrongParameters** ()
- bool **isValidParameters** (string &inputFilepath, string &outputFilename, bool &debugFlag, int argc, char **argv)
- void **loadDetectorParameterValues** (RegionDetector &detector)
- void **saveDetectorParameterValues** (RegionDetector &detector)
- void **loadDetectorParameterValues** (RegionDetector &detector, bool debugMode)
- void **saveDetectorParameterValues** (RegionDetector &detector, bool debugMode)
- int **main** (int argc, char **argv)

8.61.1 Macro Definition Documentation

8.61.1.1 **#define CONFIG_FILE "config/analysis/spatial/rectangular_region_detector.xml"**

Definition at line 30 of file RectangularDetectRegions.cpp.

Referenced by **loadDetectorParameterValues()**, and **saveDetectorParameterValues()**.

8.61.1.2 **#define ERR_CODE 1**

Definition at line 43 of file RectangularDetectRegions.cpp.

Referenced by **main()**.

8.61.1.3 **#define ERR_MSG "An error occurred: "**

Definition at line 44 of file RectangularDetectRegions.cpp.

Referenced by **main()**, and **printWrongParameters()**.

8.61.1.4 **#define LABEL_ALPHA "detector.alpha"**

Definition at line 33 of file RectangularDetectRegions.cpp.

Referenced by **loadDetectorParameterValues()**, and **saveDetectorParameterValues()**.

8.61.1.5 **#define LABEL_BETA "detector.beta"**

Definition at line 34 of file RectangularDetectRegions.cpp.

Referenced by **loadDetectorParameterValues()**, and **saveDetectorParameterValues()**.

8.61.1.6 **#define LABEL_BLUR_KERNEL_SIZE "detector.blurKernelSize"**

Definition at line 35 of file RectangularDetectRegions.cpp.

Referenced by **loadDetectorParameterValues()**, and **saveDetectorParameterValues()**.

8.61.1.7 **#define LABEL_EPSILON "detector.epsilon"**

Definition at line 37 of file RectangularDetectRegions.cpp.

Referenced by **loadDetectorParameterValues()**, and **saveDetectorParameterValues()**.

8.61.1.8 `#define LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS "detector.morphologicalCloseIterations"`

Definition at line 36 of file RectangularDetectRegions.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.61.1.9 `#define LABEL_REGION_AREA_THRESH "detector.regionAreaThresh"`

Definition at line 38 of file RectangularDetectRegions.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.61.1.10 `#define LABEL_ROOT_COMMENT "<xmlcomment>"`

Definition at line 32 of file RectangularDetectRegions.cpp.

Referenced by `saveDetectorParameterValues()`.

8.61.1.11 `#define LABEL_THRESHOLD_VALUE "detector.thresholdValue"`

Definition at line 39 of file RectangularDetectRegions.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.61.1.12 `#define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."`

Definition at line 41 of file RectangularDetectRegions.cpp.

Referenced by `saveDetectorParameterValues()`.

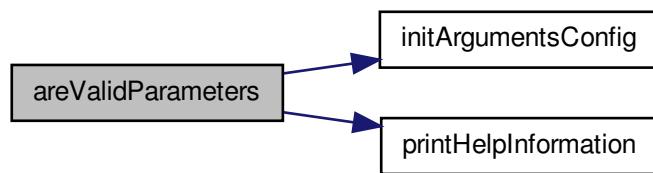
8.61.2 Function Documentation

8.61.2.1 `bool areValidParameters (string & inputfilepath, string & outputfilename, bool & debugFlag, int argc, char ** argv)`

Definition at line 72 of file RectangularDetectRegions.cpp.

References `initArgumentsConfig()`, and `printHelpInformation()`.

Here is the call graph for this function:



8.61.2.2 po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)

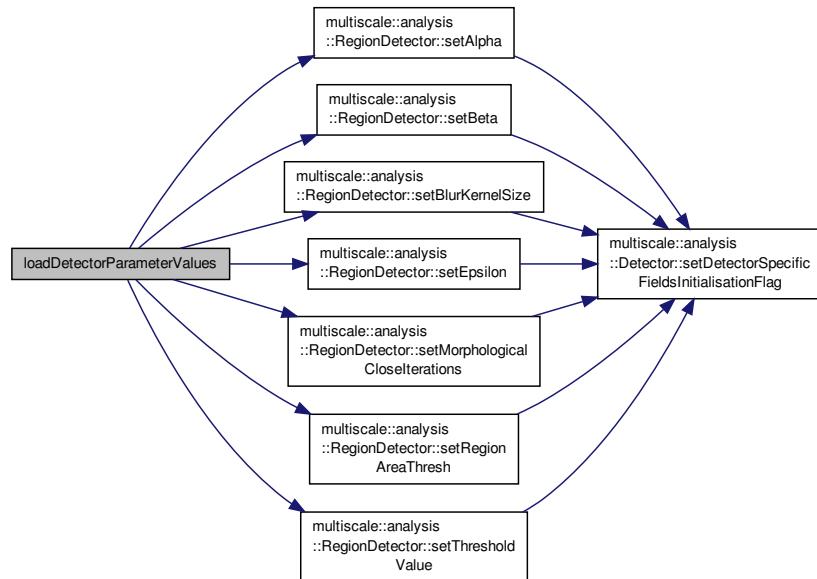
Definition at line 48 of file RectangularDetectRegions.cpp.

8.61.2.3 void loadDetectorParameterValues (RegionDetector & detector)

Definition at line 100 of file RectangularDetectRegions.cpp.

References CONFIG_FILE, LABEL_ALPHA, LABEL_BETA, LABEL_BLUR_KERNEL_SIZE, LABEL_EPSILON, LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS, LABEL_REGION_AREA_THRESH, LABEL_THRESHOLD_VALUE, multiscale::analysis::RegionDetector::setAlpha(), multiscale::analysis::RegionDetector::setBeta(), multiscale::analysis::RegionDetector::setBlurKernelSize(), multiscale::analysis::RegionDetector::setEpsilon(), multiscale::analysis::RegionDetector::setMorphologicalCloselterations(), multiscale::analysis::RegionDetector::setRegionAreaThresh(), and multiscale::analysis::RegionDetector::setThresholdValue().

Here is the call graph for this function:

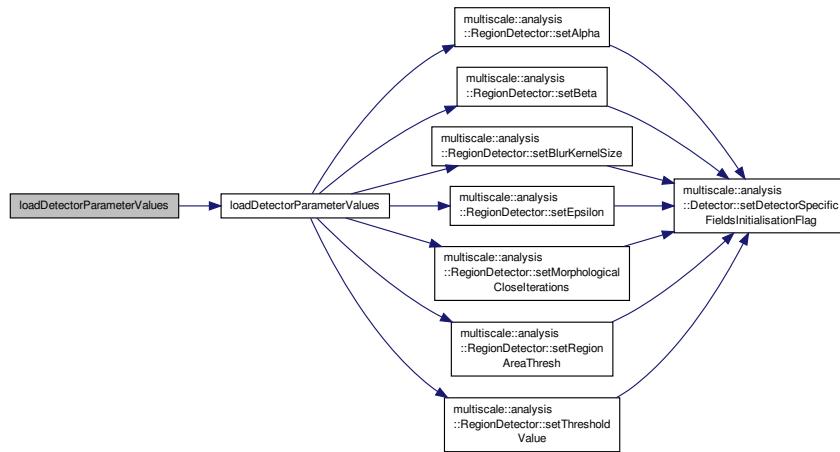


8.61.2.4 void loadDetectorParameterValues (RegionDetector & detector, bool debugMode)

Definition at line 134 of file RectangularDetectRegions.cpp.

References loadDetectorParameterValues().

Here is the call graph for this function:

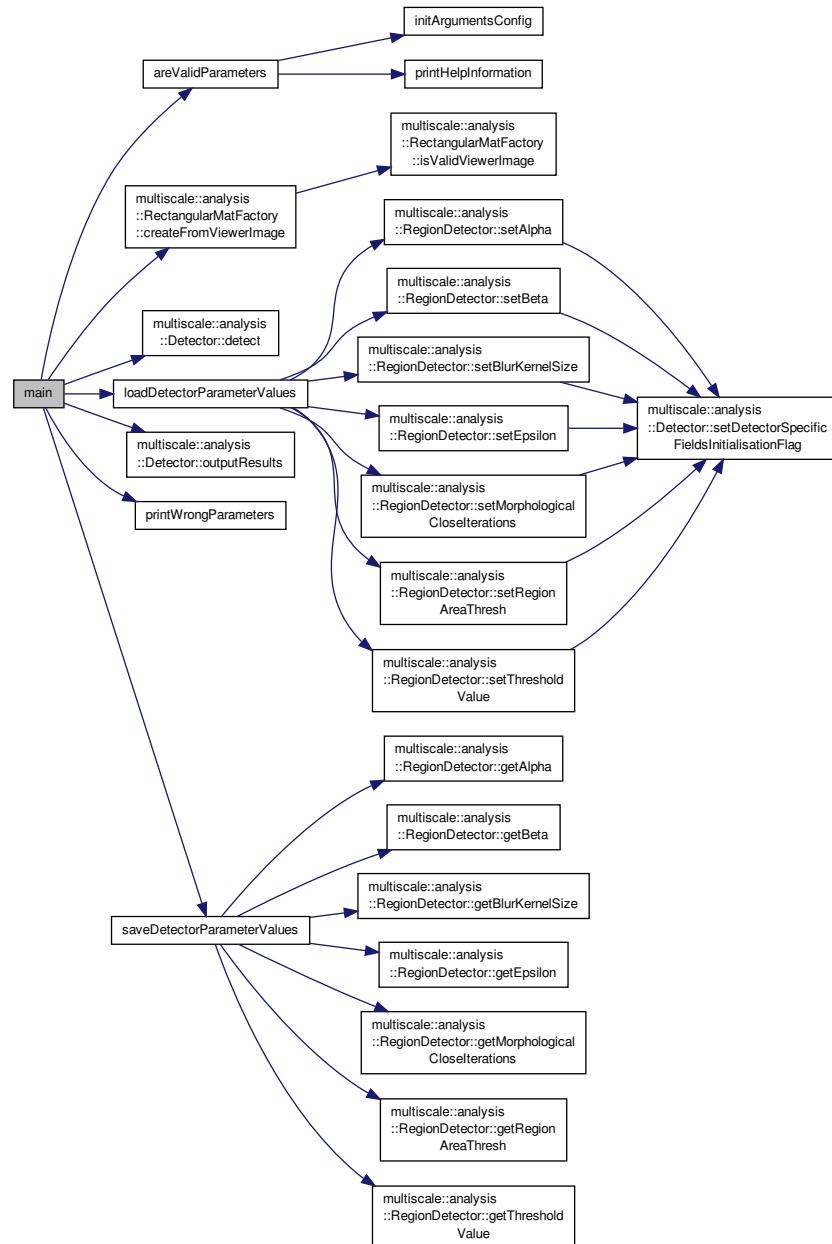


8.61.2.5 int main (int argc, char ** argv)

Definition at line 146 of file RectangularDetectRegions.cpp.

References areValidParameters(), multiscale::analysis::RectangularMatFactory::createFromViewerImage(), multiscale::analysis::Detector::detect(), ERR_CODE, ERR_MSG, loadDetectorParameterValues(), multiscale::analysis::Detector::outputResults(), printWrongParameters(), and saveDetectorParameterValues().

Here is the call graph for this function:



8.61.2.6 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 61 of file RectangularDetectRegions.cpp.

8.61.2.7 void printWrongParameters ()

Definition at line 66 of file RectangularDetectRegions.cpp.

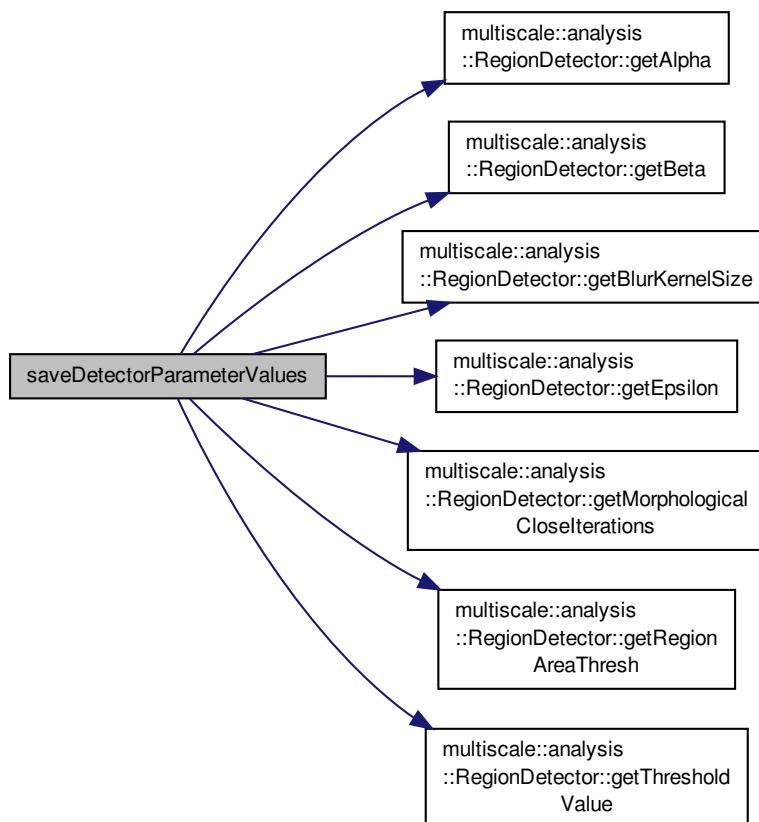
References ERR_MSG.

8.61.2.8 void saveDetectorParameterValues (RegionDetector & detector)

Definition at line 115 of file RectangularDetectRegions.cpp.

References CONFIG_FILE, multiscale::analysis::RegionDetector::getAlpha(), multiscale::analysis::RegionDetector::getBeta(), multiscale::analysis::RegionDetector::getBlurKernelSize(), multiscale::analysis::RegionDetector::getEpsilon(), multiscale::analysis::RegionDetector::getMorphologicalCloselterations(), multiscale::analysis::RegionDetector::getRegionAreaThresh(), multiscale::analysis::RegionDetector::getThresholdValue(), LABEL_ALPHA, LABEL_BETA, LABEL_BLUR_KERNEL_SIZE, LABEL_EPSILON, LABEL_MORPHOLOGICAL_CLOSE_ITERATIONS, LABEL_REGION_AREA_THRESH, LABEL_ROOT_COMMENT, LABEL_THRESHOLD_VALUE, and ROOT_COMMENT.

Here is the call graph for this function:

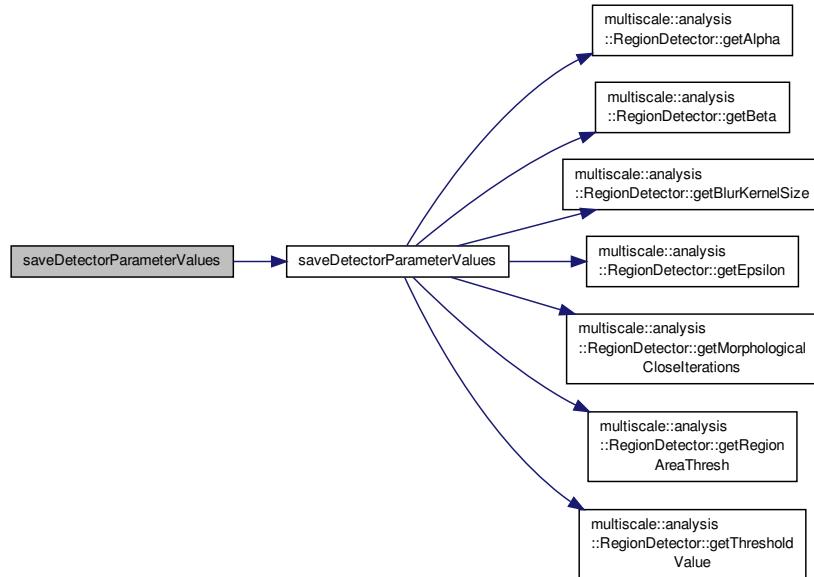


8.61.2.9 void saveDetectorParameterValues (RegionDetector & detector, bool debugMode)

Definition at line 139 of file RectangularDetectRegions.cpp.

References `saveDetectorParameterValues()`.

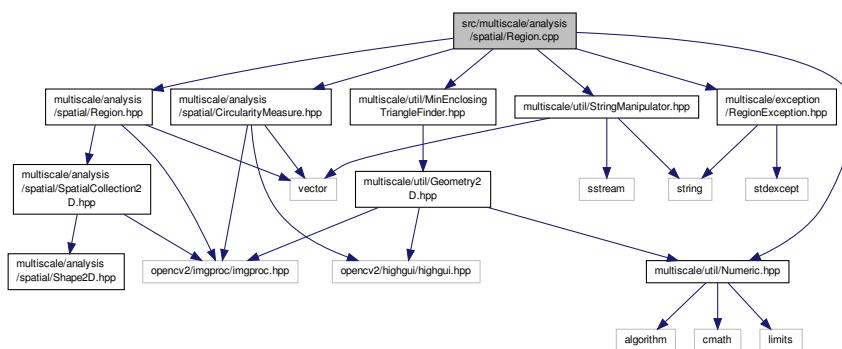
Here is the call graph for this function:



8.62 src/multiscale/analysis/spatial/Region.cpp File Reference

```

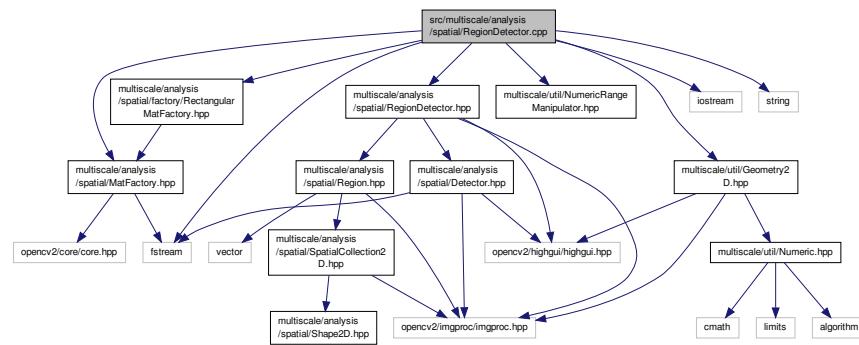
#include "multiscale/analysis/spatial/CircularityMeasure.hpp"
#include "multiscale/analysis/spatial/Region.hpp"
#include "multiscale/exception/RegionException.hpp"
#include "multiscale/util/MinEnclosingTriangleFinder.hpp"
#include "multiscale/util/Numeric.hpp"
#include "multiscale/util/StringManipulator.hpp"
Include dependency graph for Region.cpp:
  
```



8.63 src/multiscale/analysis/spatial/RegionDetector.cpp File Reference

```
#include "multiscale/analysis/spatial/MatFactory.hpp"
```

```
#include "multiscale/analysis/spatial/factory/RectangularMatFactory.hpp"
#include "multiscale/analysis/spatial/RegionDetector.hpp"
#include "multiscale/util/NumericRangeManipulator.hpp"
#include "multiscale/util/Geometry2D.hpp"
#include <iostream>
#include <fstream>
#include <string>
Include dependency graph for RegionDetector.cpp:
```



Functions

- void `convertVertices` (const Point *src, vector< Point > &dst)

8.63.1 Function Documentation

8.63.1.1 void convertVertices (const Point * src, vector< Point > & dst)

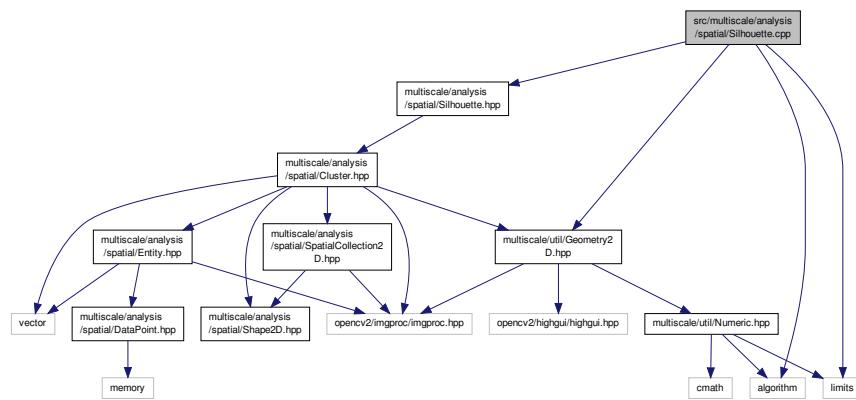
Definition at line 343 of file `RegionDetector.cpp`.

References `ENCLOSING_RECT_VERTICES`.

8.64 src/multiscale/analysis/spatial/Silhouette.cpp File Reference

```
#include "multiscale/analysis/spatial/Silhouette.hpp"
#include "multiscale/util/Geometry2D.hpp"
#include <algorithm>
#include <limits>
```

Include dependency graph for Silhouette.cpp:

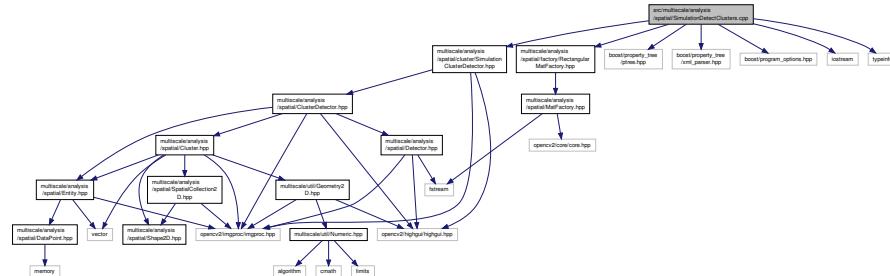


8.65 src/multiscale/analysis/spatial/SimulationDetectClusters.cpp File Reference

```

#include "multiscale/analysis/spatial/cluster/SimulationClusterDetector.hpp"
#include "multiscale/analysis/spatial/factory/RectangularMatFactory.hpp"
#include <boost/property_tree/ptree.hpp>
#include <boost/property_tree/xml_parser.hpp>
#include <boost/program_options.hpp>
#include <iostream>
#include <typeinfo>
  
```

Include dependency graph for SimulationDetectClusters.cpp:



Macros

- #define CONFIG_FILE "config/analysis/spatial/simulation_cluster_detector.xml"
- #define LABEL_ROOT_COMMENT "<xmlcomment>"
- #define LABEL_EPS "detector.eps"
- #define LABEL_MINPOINTS "detector.minPoints"
- #define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."
- #define ERR_CODE 1
- #define ERR_MSG "An error occurred: "

Functions

- `po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, bool &debugFlag, unsigned int &height, unsigned int &width, unsigned int &maxPileup, int argc, char **argv)`
- `void loadDetectorParameterValues (SimulationClusterDetector &detector)`
- `void saveDetectorParameterValues (SimulationClusterDetector &detector)`
- `void loadDetectorParameterValues (SimulationClusterDetector &detector, bool debugMode)`
- `void saveDetectorParameterValues (SimulationClusterDetector &detector, bool debugMode)`
- `int main (int argc, char **argv)`

8.65.1 Macro Definition Documentation

8.65.1.1 `#define CONFIG_FILE "config/analysis/spatial/simulation_cluster_detector.xml"`

Definition at line 31 of file SimulationDetectClusters.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.65.1.2 `#define ERR_CODE 1`

Definition at line 39 of file SimulationDetectClusters.cpp.

Referenced by `main()`.

8.65.1.3 `#define ERR_MSG "An error occurred: "`

Definition at line 40 of file SimulationDetectClusters.cpp.

Referenced by `main()`, and `printWrongParameters()`.

8.65.1.4 `#define LABEL_EPS "detector.eps"`

Definition at line 34 of file SimulationDetectClusters.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.65.1.5 `#define LABEL_MINPOINTS "detector.minPoints"`

Definition at line 35 of file SimulationDetectClusters.cpp.

Referenced by `loadDetectorParameterValues()`, and `saveDetectorParameterValues()`.

8.65.1.6 `#define LABEL_ROOT_COMMENT "<xmlcomment>"`

Definition at line 33 of file SimulationDetectClusters.cpp.

Referenced by `saveDetectorParameterValues()`.

8.65.1.7 `#define ROOT_COMMENT "Warning! This xml file was automatically generated by a C++ program using the Boost PropertyTree library."`

Definition at line 37 of file SimulationDetectClusters.cpp.

Referenced by `saveDetectorParameterValues()`.

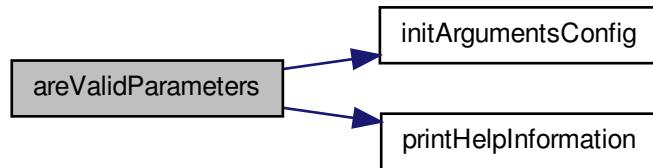
8.65.2 Function Documentation

8.65.2.1 `bool areValidParameters (string & inputFilepath, string & outputFilename, bool & debugFlag, unsigned int & height, unsigned int & width, unsigned int & maxPileup, int argc, char ** argv)`

Definition at line 71 of file SimulationDetectClusters.cpp.

References `initArgumentsConfig()`, and `printHelpInformation()`.

Here is the call graph for this function:



8.65.2.2 `po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)`

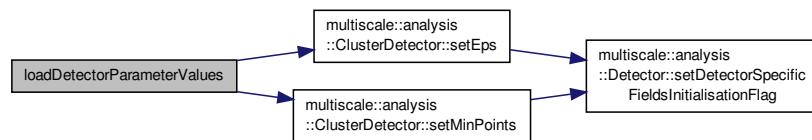
Definition at line 44 of file SimulationDetectClusters.cpp.

8.65.2.3 `void loadDetectorParameterValues (SimulationClusterDetector & detector)`

Definition at line 105 of file SimulationDetectClusters.cpp.

References `CONFIG_FILE`, `LABEL_EPS`, `LABEL_MINPOINTS`, `multiscale::analysis::ClusterDetector::setEps()`, and `multiscale::analysis::ClusterDetector::setMinPoints()`.

Here is the call graph for this function:

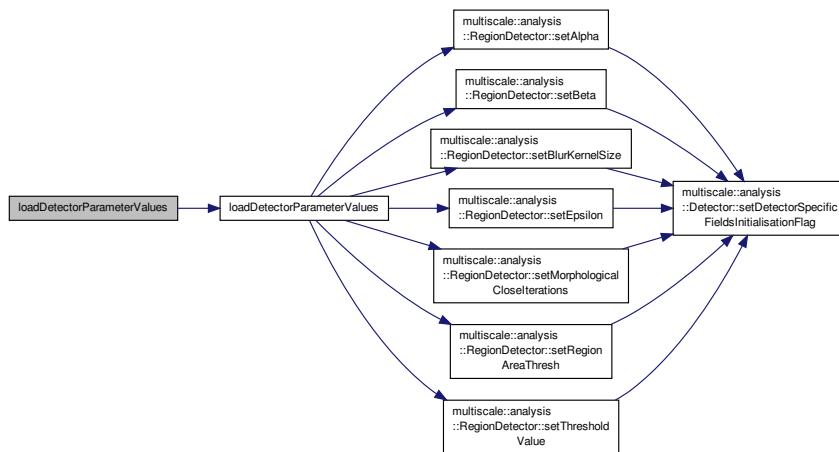


8.65.2.4 `void loadDetectorParameterValues (SimulationClusterDetector & detector, bool debugMode)`

Definition at line 129 of file SimulationDetectClusters.cpp.

References `loadDetectorParameterValues()`.

Here is the call graph for this function:

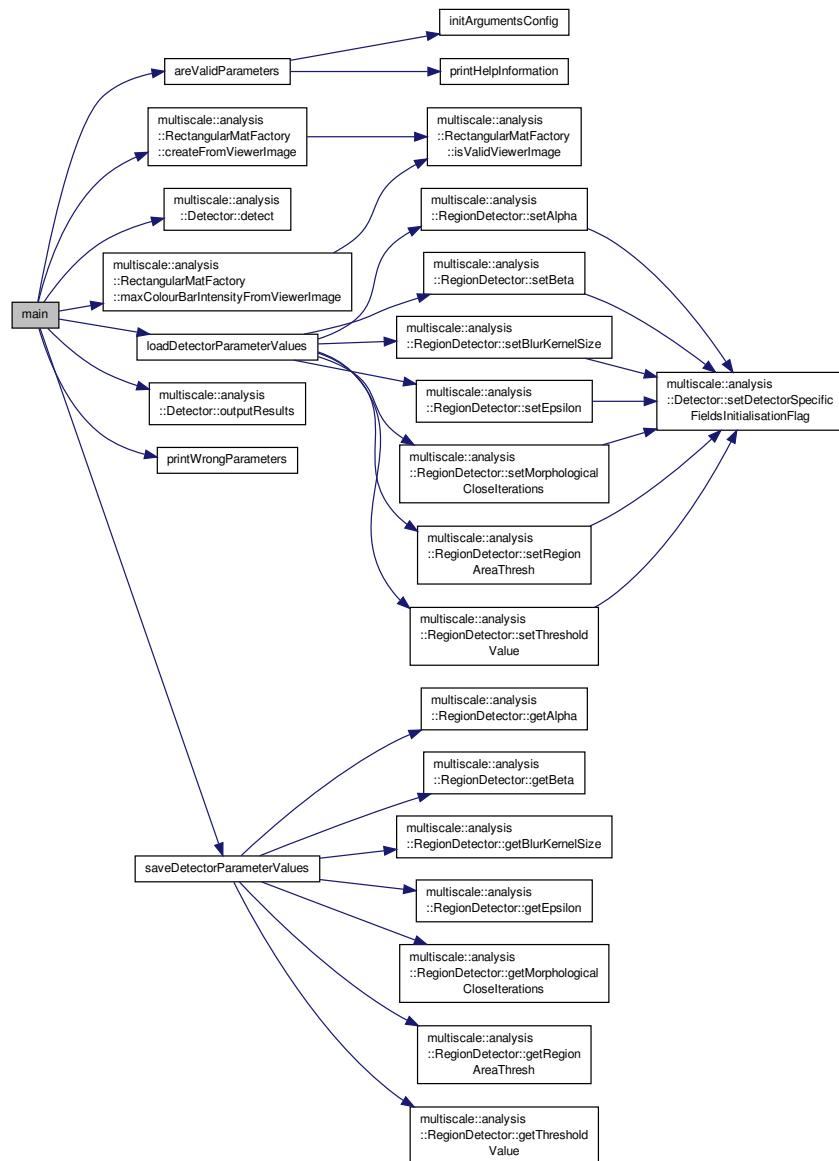


8.65.2.5 int main (int argc, char ** argv)

Definition at line 141 of file SimulationDetectClusters.cpp.

References areValidParameters(), multiscale::analysis::RectangularMatFactory::createFromViewerImage(), multiscale::analysis::Detector::detect(), ERR_CODE, ERR_MSG, loadDetectorParameterValues(), multiscale::analysis::RectangularMatFactory::maxColourBarIntensityFromViewerImage(), multiscale::analysis::Detector::outputResults(), printWrongParameters(), and saveDetectorParameterValues().

Here is the call graph for this function:



8.65.2.6 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 60 of file SimulationDetectClusters.cpp.

8.65.2.7 void printWrongParameters ()

Definition at line 65 of file SimulationDetectClusters.cpp.

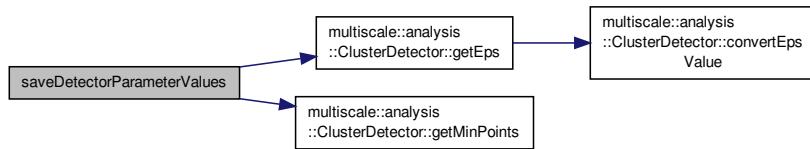
References ERR_MSG.

8.65.2.8 void saveDetectorParameterValues (*SimulationClusterDetector & detector*)

Definition at line 115 of file SimulationDetectClusters.cpp.

References CONFIG_FILE, multiscale::analysis::ClusterDetector::getEps(), multiscale::analysis::ClusterDetector::getMinPoints(), LABEL_EPS, LABEL_MINPOINTS, LABEL_ROOT_COMMENT, and ROOT_COMMENT.

Here is the call graph for this function:

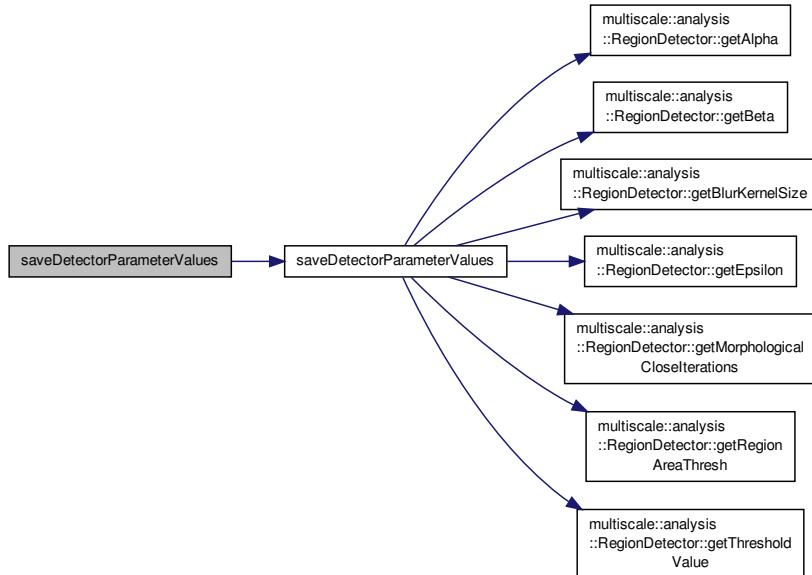


8.65.2.9 void saveDetectorParameterValues (*SimulationClusterDetector & detector, bool debugMode*)

Definition at line 134 of file SimulationDetectClusters.cpp.

References saveDetectorParameterValues().

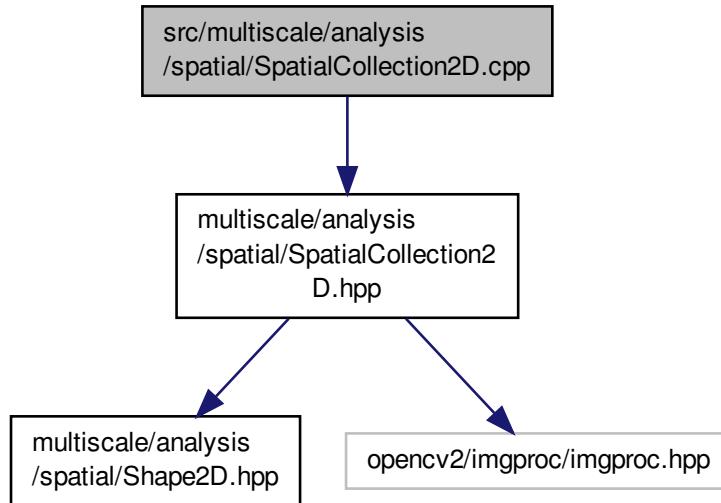
Here is the call graph for this function:



8.66 src/multiscale/analysis/spatial/SpatialCollection2D.cpp File Reference

```
#include "multiscale/analysis/spatial/SpatialCollection2D.hpp"
```

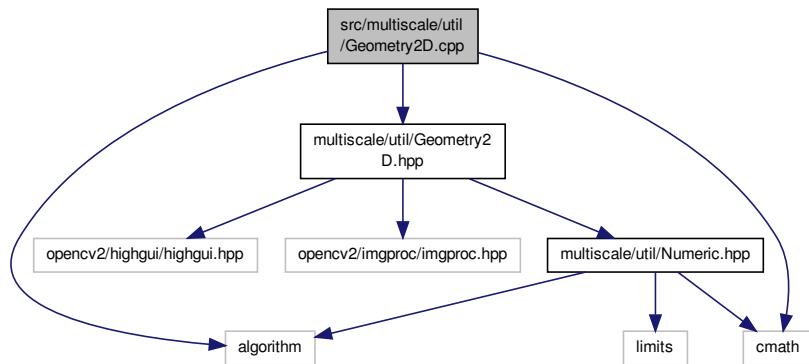
Include dependency graph for SpatialCollection2D.cpp:



8.67 src/multiscale/util/Geometry2D.cpp File Reference

```
#include "multiscale/util/Geometry2D.hpp"
#include <algorithm>
#include <cmath>
```

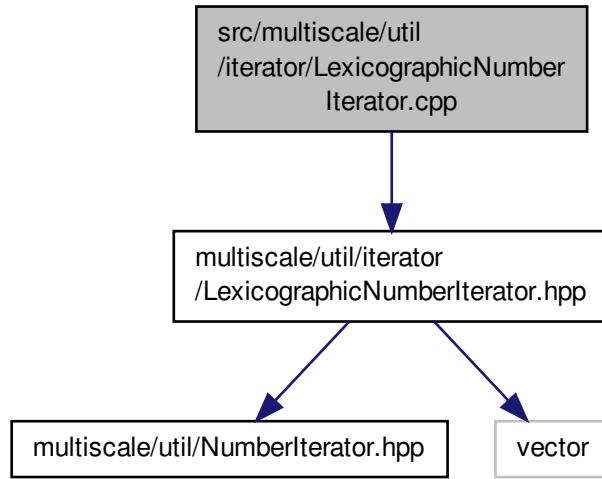
Include dependency graph for Geometry2D.cpp:



8.68 src/multiscale/util/iterator/LexicographicNumberIterator.cpp File Reference

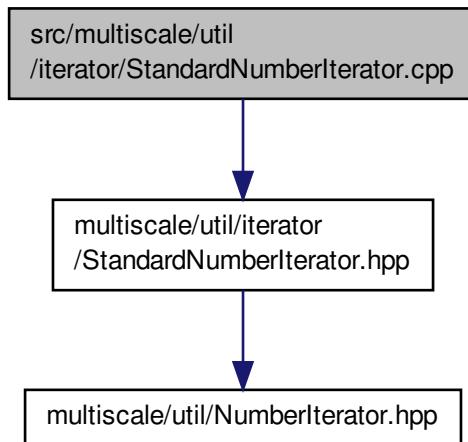
```
#include "multiscale/util/iterator/LexicographicNumberIterator.hpp"
```

Include dependency graph for LexicographicNumberIterator.cpp:



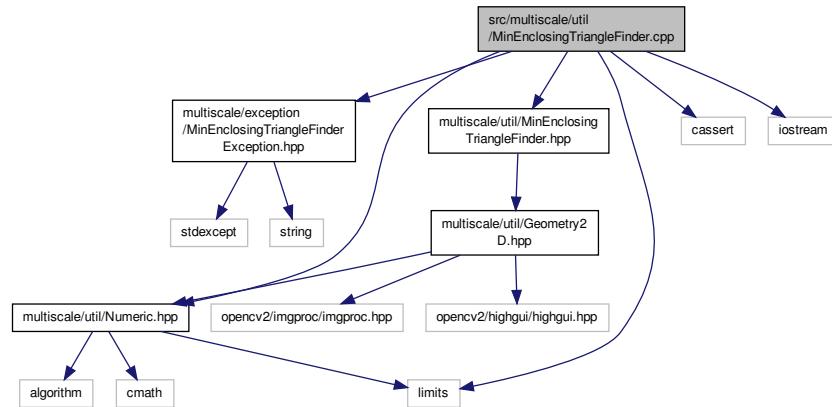
8.69 src/multiscale/util/iterator/StandardNumberIterator.cpp File Reference

```
#include "multiscale/util/iterator/StandardNumberIterator.hpp"  
Include dependency graph for StandardNumberIterator.cpp:
```



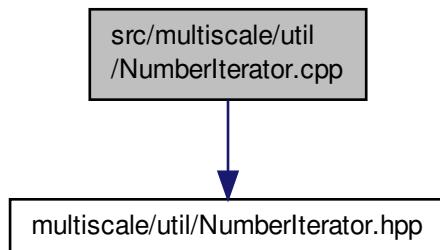
8.70 src/multiscale/util/MinEnclosingTriangleFinder.cpp File Reference

```
#include "multiscale/exception/MinEnclosingTriangleFinderException.hpp"
#include "multiscale/util/MinEnclosingTriangleFinder.hpp"
#include "multiscale/util/Numeric.hpp"
#include <cassert>
#include <iostream>
#include <limits>
Include dependency graph for MinEnclosingTriangleFinder.cpp:
```



8.71 src/multiscale/util/NumberIterator.cpp File Reference

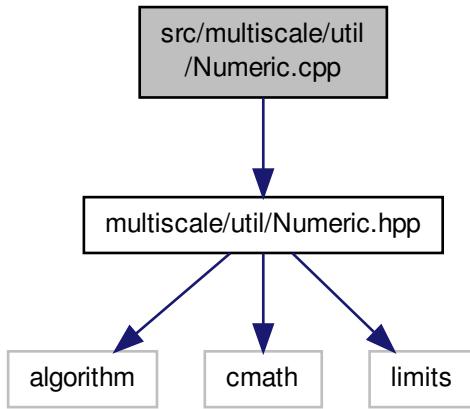
```
#include "multiscale/util/NumberIterator.hpp"
Include dependency graph for NumberIterator.cpp:
```



8.72 src/multiscale/util/Numeric.cpp File Reference

```
#include "multiscale/util/Numeric.hpp"
```

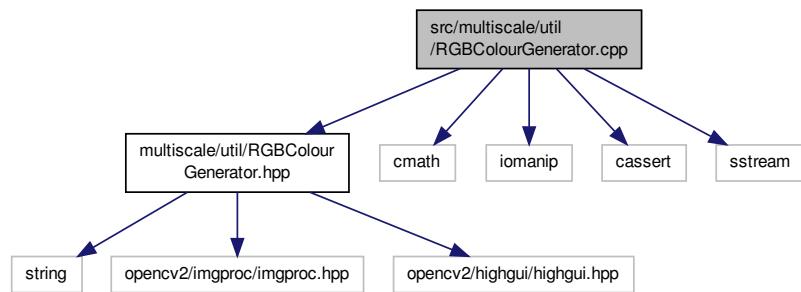
Include dependency graph for Numeric.cpp:



8.73 src/multiscale/util/RGBColourGenerator.cpp File Reference

```
#include "multiscale/util/RGBColourGenerator.hpp"
#include <cmath>
#include <iomanip>
#include <cassert>
#include <sstream>
```

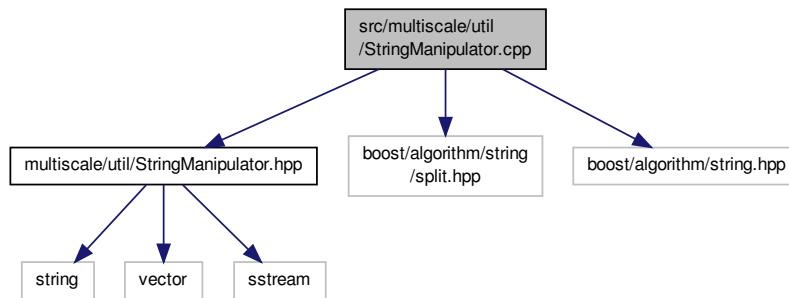
Include dependency graph for RGBColourGenerator.cpp:



8.74 src/multiscale/util/StringManipulator.cpp File Reference

```
#include "multiscale/util/StringManipulator.hpp"
#include <boost/algorithm/string/split.hpp>
#include <boost/algorithm/string.hpp>
```

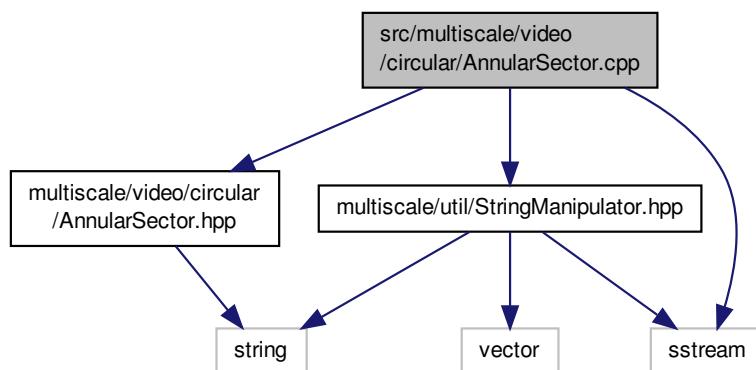
Include dependency graph for StringManipulator.cpp:



8.75 src/multiscale/video/circular/AnnularSector.cpp File Reference

```
#include "multiscale/video/circular/AnnularSector.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <sstream>
```

Include dependency graph for AnnularSector.cpp:



Macros

- #define SEPARATOR " "

8.75.1 Macro Definition Documentation

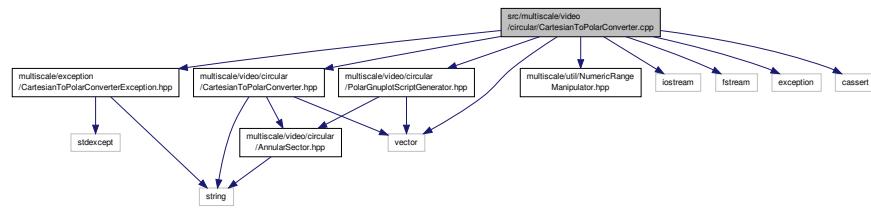
8.75.1.1 #define SEPARATOR " "

Definition at line 6 of file AnnularSector.cpp.

Referenced by multiscale::video::AnnularSector::toString().

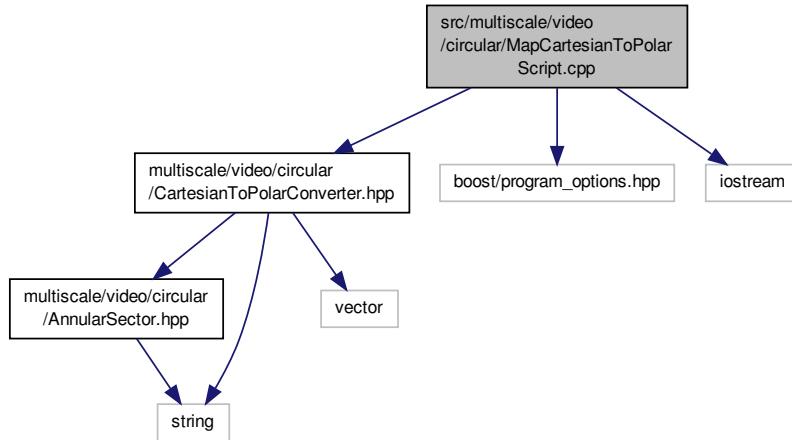
8.76 src/multiscale/video/circular/CartesianToPolarConverter.cpp File Reference

```
#include "multiscale/exception/CartesianToPolarConverterException.hpp"
#include "multiscale/video/circular/CartesianToPolarConverter.hpp"
#include "multiscale/video/circular/PolarGnuplotScriptGenerator.hpp"
#include "multiscale/util/NumericRangeManipulator.hpp"
#include <iostream>
#include <fstream>
#include <exception>
#include <cassert>
#include <vector>
Include dependency graph for CartesianToPolarConverter.cpp:
```



8.77 src/multiscale/video/circular/MapCartesianToPolarScript.cpp File Reference

```
#include "multiscale/video/circular/CartesianToPolarConverter.hpp"
#include <boost/program_options.hpp>
#include <iostream>
Include dependency graph for MapCartesianToPolarScript.cpp:
```



Macros

- `#define ERR_CODE 1`
- `#define ERR_MSG "An error occurred: "`

Functions

- `po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `bool isValidOutputType (const po::variables_map &vm, bool &isScript)`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, bool &isScript, int argc, char **argv)`
- `int main (int argc, char **argv)`

8.77.1 Macro Definition Documentation

8.77.1.1 `#define ERR_CODE 1`

Definition at line 47 of file MapCartesianToPolarScript.cpp.

Referenced by `main()`.

8.77.1.2 `#define ERR_MSG "An error occurred: "`

Definition at line 48 of file MapCartesianToPolarScript.cpp.

Referenced by `isValidOutputType()`, `main()`, and `printWrongParameters()`.

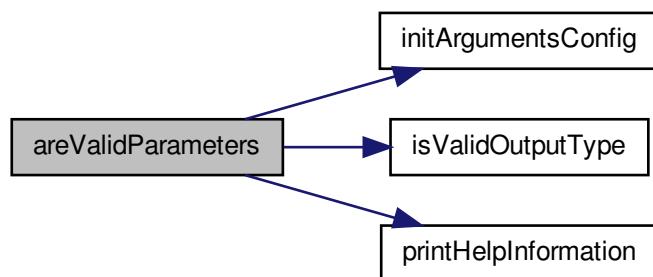
8.77.2 Function Documentation

8.77.2.1 `bool areValidParameters (string & inputFilepath, string & outputFilename, bool & isScript, int argc, char ** argv)`

Definition at line 100 of file MapCartesianToPolarScript.cpp.

References `initArgumentsConfig()`, `isValidOutputType()`, and `printHelpInformation()`.

Here is the call graph for this function:



8.77.2.2 `po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)`

Definition at line 52 of file MapCartesianToPolarScript.cpp.

8.77.2.3 bool isValidOutputType (const po::variables_map & vm, bool & isScript)

Definition at line 79 of file MapCartesianToPolarScript.cpp.

References ERR_MSG.

Referenced by areValidParameters().

Here is the caller graph for this function:

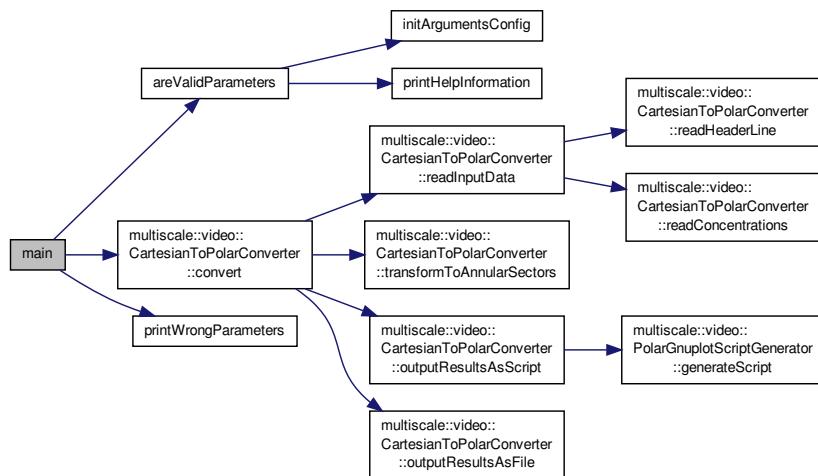


8.77.2.4 int main (int argc, char ** argv)

Definition at line 126 of file MapCartesianToPolarScript.cpp.

References areValidParameters(), multiscale::video::CartesianToPolarConverter::convert(), ERR_CODE, ERR_MSG, and printWrongParameters().

Here is the call graph for this function:



8.77.2.5 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 68 of file MapCartesianToPolarScript.cpp.

8.77.2.6 void printWrongParameters ()

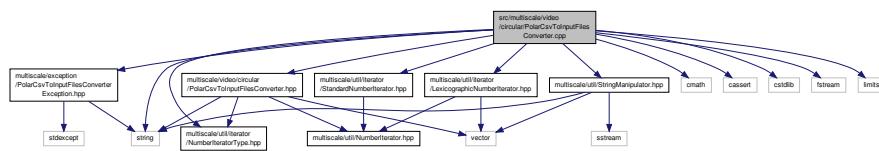
Definition at line 73 of file MapCartesianToPolarScript.cpp.

References ERR_MSG.

8.78 src/multiscale/video/circular/PolarCsvToInputFilesConverter.cpp File Reference

```
#include "multiscale/exception/PolarCsvToInputFilesConverterException.hpp"
#include "multiscale/video/circular/PolarCsvToInputFilesConverter.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include "multiscale/util/iterator/LexicographicNumberIterator.hpp"
#include "multiscale/util/iterator/StandardNumberIterator.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <cmath>
#include <cassert>
#include <cstdlib>
#include <fstream>
#include <limits>
#include <string>
```

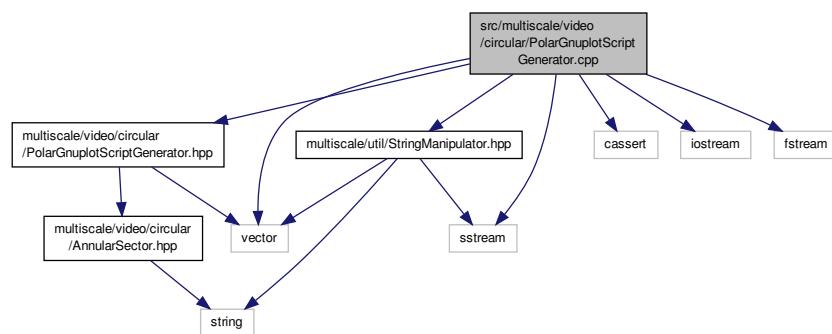
Include dependency graph for PolarCsvToInputFilesConverter.cpp:



8.79 src/multiscale/video/circular/PolarGnuplotScriptGenerator.cpp File Reference

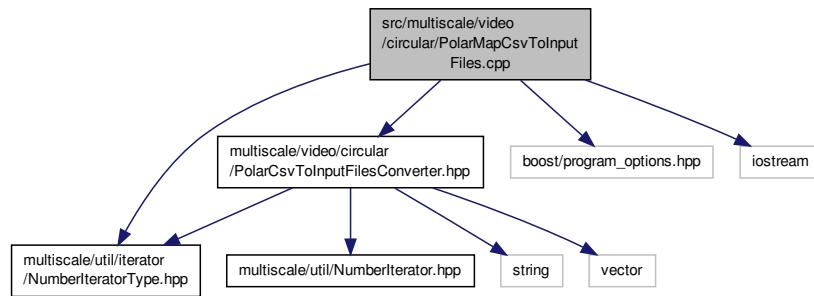
```
#include "multiscale/video/circular/PolarGnuplotScriptGenerator.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <cassert>
#include <iostream>
#include <vector>
#include <sstream>
#include <fstream>
```

Include dependency graph for PolarGnuplotScriptGenerator.cpp:



8.80 src/multiscale/video/circular/PolarMapCsvToInputFiles.cpp File Reference

```
#include "multiscale/video/circular/PolarCsvToInputFilesConverter.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <boost/program_options.hpp>
#include <iostream>
Include dependency graph for PolarMapCsvToInputFiles.cpp:
```



Macros

- `#define ERR_CODE 1`
- `#define ERR_MSG "An error occurred: "`

Functions

- `po::variables_map initArgumentsConfig(po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `void setNumberIteratorType (const po::variables_map &vm, NumberIteratorType &numberIteratorType)`
- `void setSelectedConcentrationIndex (const po::variables_map &vm, unsigned int &selectedConcentrationIndex)`
- `void setLogScaling (const po::variables_map &vm, bool &useLogScaling)`
- `bool isValidNrOfConcentrationsForPosition (const po::variables_map &vm, unsigned int &nrOfConcentrationsForPosition)`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, unsigned int &nrOfConcentricCircles, unsigned int &nrOfSectors, unsigned int &nrOfConcentrationsForPosition, unsigned int &selectedConcentrationIndex, bool &useLogScaling, NumberIteratorType &numberIteratorType, int argc, char **argv)`
- `int main (int argc, char **argv)`

8.80.1 Macro Definition Documentation

8.80.1.1 #define ERR_CODE 1

Definition at line 28 of file `PolarMapCsvToInputFiles.cpp`.

Referenced by `main()`.

8.80.1.2 #define ERR_MSG "An error occurred: "

Definition at line 29 of file PolarMapCsvToInputFiles.cpp.

Referenced by isValidNrOfConcentrationsForPosition(), main(), and printWrongParameters().

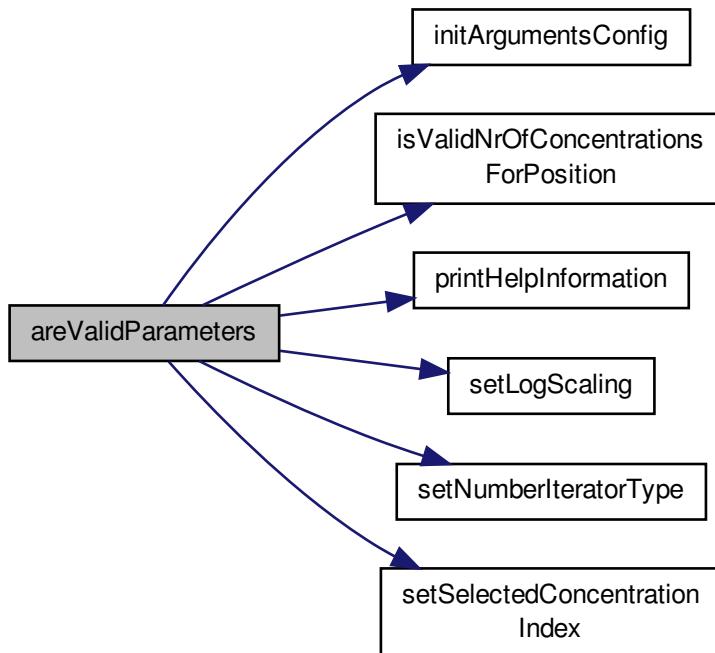
8.80.2 Function Documentation

8.80.2.1 bool areValidParameters (string & *inputFilepath*, string & *outputFilename*, unsigned int & *nrOfConcentricCircles*, unsigned int & *nrOfSectors*, unsigned int & *nrOfConcentrationsForPosition*, unsigned int & *selectedConcentrationIndex*, bool & *useLogScaling*, NumberIteratorType & *numberIteratorType*, int *argc*, char ** *argv*)

Definition at line 92 of file PolarMapCsvToInputFiles.cpp.

References initArgumentsConfig(), isValidNrOfConcentrationsForPosition(), printHelpInformation(), setLogScaling(), setNumberIteratorType(), and setSelectedConcentrationIndex().

Here is the call graph for this function:



8.80.2.2 po::variables_map initArgumentsConfig (po::options_description & *usageDescription*, int *argc*, char ** *argv*)

Definition at line 33 of file PolarMapCsvToInputFiles.cpp.

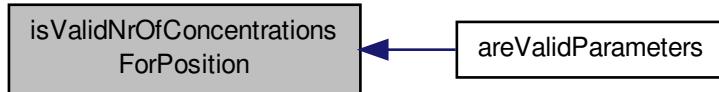
8.80.2.3 bool isValidNrOfConcentrationsForPosition (const po::variables_map & *vm*, unsigned int & *nrOfConcentrationsForPosition*)

Definition at line 77 of file PolarMapCsvToInputFiles.cpp.

References ERR_MSG.

Referenced by areValidParameters().

Here is the caller graph for this function:

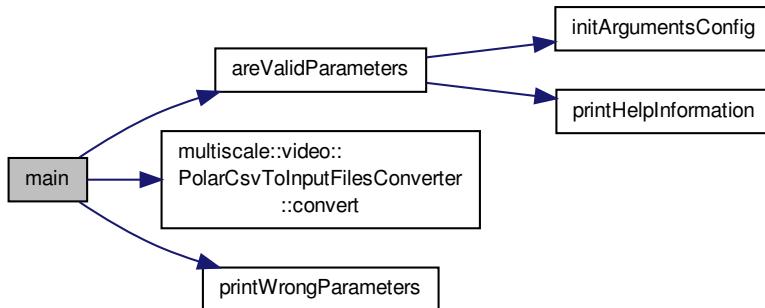


8.80.2.4 int main (int argc, char ** argv)

Definition at line 139 of file PolarMapCsvToInputFiles.cpp.

References areValidParameters(), multiscale::video::PolarCsvToInputFilesConverter::convert(), ERR_CODE, ERR_MSG, printWrongParameters(), and multiscale::STANDARD.

Here is the call graph for this function:



8.80.2.5 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 51 of file PolarMapCsvToInputFiles.cpp.

8.80.2.6 void printWrongParameters ()

Definition at line 56 of file PolarMapCsvToInputFiles.cpp.

References ERR_MSG.

8.80.2.7 void setLogScaling (const po::variables_map & vm, bool & useLogScaling)

Definition at line 72 of file PolarMapCsvToInputFiles.cpp.

Referenced by `isValidParameters()`.

Here is the caller graph for this function:



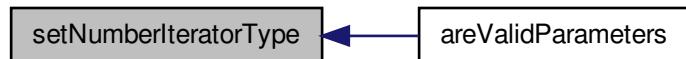
8.80.2.8 void `setNumberIteratorType (const po::variables_map & vm, NumberIteratorType & numberIteratorType)`

Definition at line 62 of file `PolarMapCsvToInputFiles.cpp`.

References `multiscale::LEXICOGRAPHIC`.

Referenced by `isValidParameters()`.

Here is the caller graph for this function:

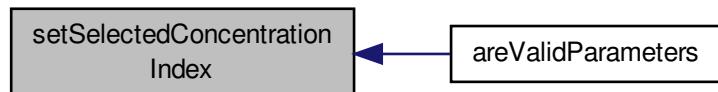


8.80.2.9 void `setSelectedConcentrationIndex (const po::variables_map & vm, unsigned int & selectedConcentrationIndex)`

Definition at line 67 of file `PolarMapCsvToInputFiles.cpp`.

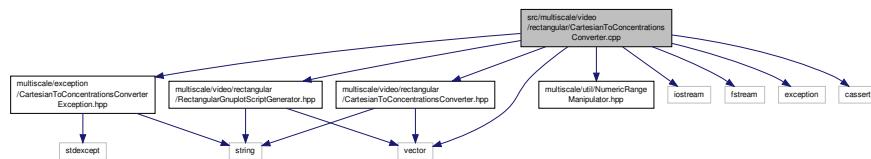
Referenced by `isValidParameters()`.

Here is the caller graph for this function:



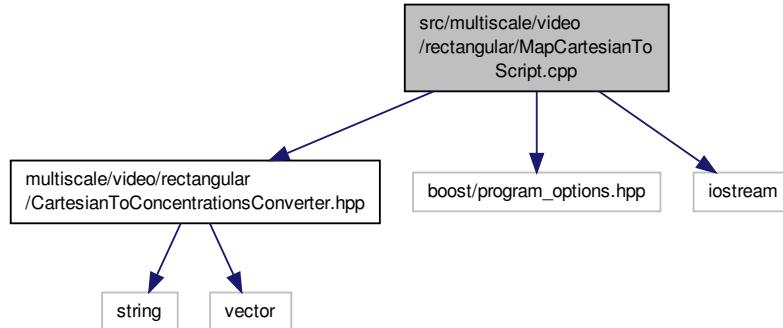
8.81 src/multiscale/video/rectangular/CartesianToConcentrationsConverter.cpp File Reference

```
#include "multiscale/exception/CartesianToConcentrationsConverterException.-  
hpp"  
#include "multiscale/video/rectangular/CartesianToConcentrationsConverter.-  
hpp"  
#include "multiscale/video/rectangular/RectangularGnuplotScriptGenerator.-  
hpp"  
#include "multiscale/util/NumericRangeManipulator.hpp"  
#include <iostream>  
#include <fstream>  
#include <exception>  
#include <cassert>  
#include <vector>  
Include dependency graph for CartesianToConcentrationsConverter.cpp:
```



8.82 src/multiscale/video/rectangular/MapCartesianToScript.cpp File Reference

```
#include "multiscale/video/rectangular/CartesianToConcentrationsConverter.-  
hpp"  
#include <boost/program_options.hpp>  
#include <iostream>  
Include dependency graph for MapCartesianToScript.cpp:
```



Macros

- #define `ERR_CODE` 1
- #define `ERR_MSG` "An error occurred: "

Functions

- `po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, int argc, char **argv)`
- `int main (int argc, char **argv)`

8.82.1 Macro Definition Documentation

8.82.1.1 `#define ERR_CODE 1`

Definition at line 32 of file MapCartesianToScript.cpp.

Referenced by `main()`.

8.82.1.2 `#define ERR_MSG "An error occurred: "`

Definition at line 33 of file MapCartesianToScript.cpp.

Referenced by `main()`, and `printWrongParameters()`.

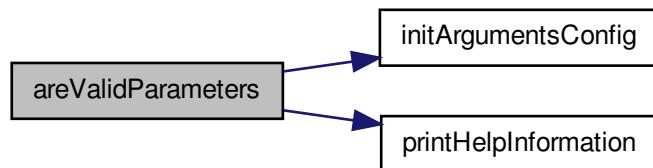
8.82.2 Function Documentation

8.82.2.1 `bool areValidParameters (string & inputfilepath, string & outputFilename, int argc, char ** argv)`

Definition at line 60 of file MapCartesianToScript.cpp.

References `initArgumentsConfig()`, and `printHelpInformation()`.

Here is the call graph for this function:



8.82.2.2 `po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)`

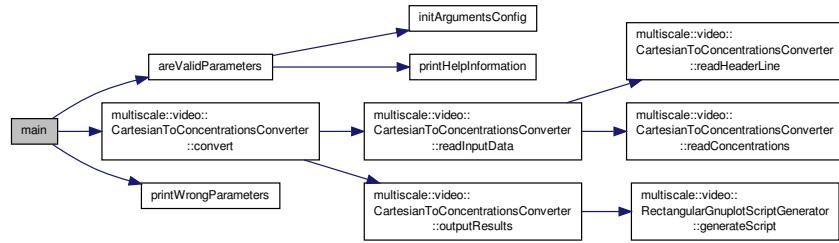
Definition at line 37 of file MapCartesianToScript.cpp.

8.82.2.3 `int main (int argc, char ** argv)`

Definition at line 84 of file MapCartesianToScript.cpp.

References `areValidParameters()`, `multiscale::video::CartesianToConcentrationsConverter::convert()`, `ERR_CODE`, `ERR_MSG`, and `printWrongParameters()`.

Here is the call graph for this function:



8.82.2.4 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 49 of file MapCartesianToScript.cpp.

8.82.2.5 void printWrongParameters ()

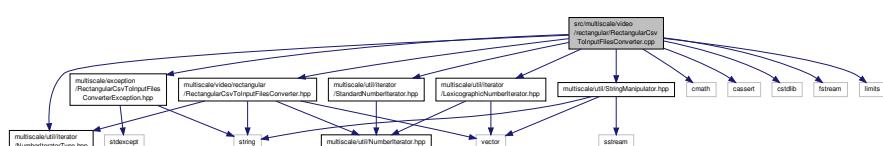
Definition at line 54 of file MapCartesianToScript.cpp.

References ERR_MSG.

8.83 src/multiscale/video/rectangular/RectangularCsvToInputFilesConverter.cpp File Reference

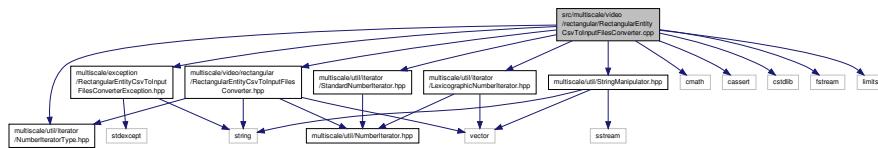
```

#include "multiscale/exception/RectangularCsvToInputFilesConverterException.-  
hpp"
#include "multiscale/video/rectangular/RectangularCsvToInputFilesConverter.-  
hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include "multiscale/util/iterator/LexicographicNumberIterator.hpp"
#include "multiscale/util/iterator/StandardNumberIterator.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <cmath>
#include <cassert>
#include <cstdlib>
#include <fstream>
#include <limits>
Include dependency graph for RectangularCsvToInputFilesConverter.cpp:
  
```



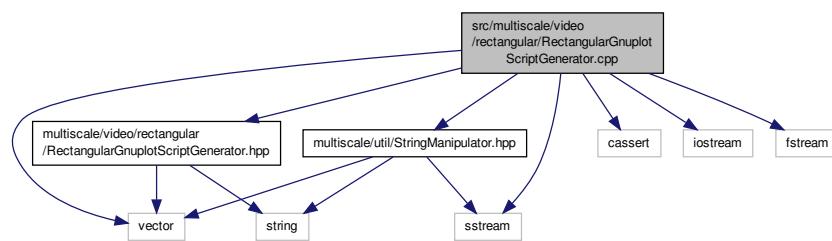
8.84 src/multiscale/video/rectangular/RectangularEntityCsvToInputFilesConverter.cpp File Reference

```
#include "multiscale/exception/RectangularEntityCsvToInputFilesConverter-
Exception.hpp"
#include "multiscale/video/rectangular/RectangularEntityCsvToInputFiles-
Converter.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include "multiscale/util/iterator/LexicographicNumberIterator.hpp"
#include "multiscale/util/iterator/StandardNumberIterator.hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <cmath>
#include <cassert>
#include <cstdlib>
#include <fstream>
#include <limits>
Include dependency graph for RectangularEntityCsvToInputFilesConverter.cpp:
```



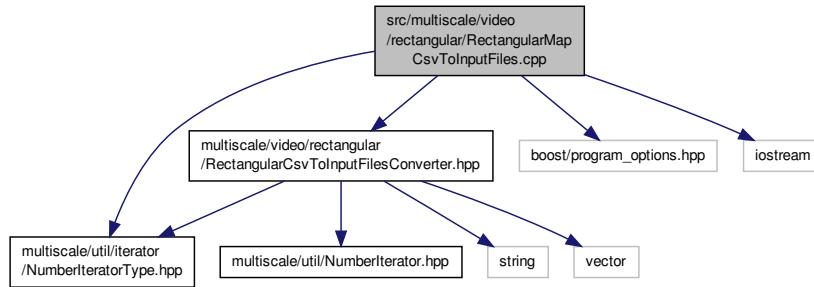
8.85 src/multiscale/video/rectangular/RectangularGnuplotScriptGenerator.cpp File Reference

```
#include "multiscale/video/rectangular/RectangularGnuplotScriptGenerator.-
hpp"
#include "multiscale/util/StringManipulator.hpp"
#include <cassert>
#include <iostream>
#include <vector>
#include <sstream>
#include <fstream>
Include dependency graph for RectangularGnuplotScriptGenerator.cpp:
```



8.86 src/multiscale/video/rectangular/RectangularMapCsvToInputFiles.cpp File Reference

```
#include "multiscale/video/rectangular/RectangularCsvToInputFilesConverter.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <boost/program_options.hpp>
#include <iostream>
Include dependency graph for RectangularMapCsvToInputFiles.cpp:
```



Macros

- `#define ERR_CODE 1`
- `#define ERR_MSG "An error occurred: "`

Functions

- `po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `void setNumberIteratorType (const po::variables_map &vm, NumberIteratorType &numberIteratorType)`
- `void setSelectedConcentrationIndex (const po::variables_map &vm, unsigned int &selectedConcentrationIndex)`
- `void setLogScaling (const po::variables_map &vm, bool &useLogScaling)`
- `bool isValidNrOfConcentrationsForPosition (const po::variables_map &vm, unsigned int &nrOfConcentrationsForPosition)`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, unsigned int &height, unsigned int &width, unsigned int &nrOfConcentrationsForPosition, unsigned int &selectedConcentrationIndex, bool &useLogScaling, NumberIteratorType &numberIteratorType, int argc, char **argv)`
- `int main (int argc, char **argv)`

8.86.1 Macro Definition Documentation

8.86.1.1 #define ERR_CODE 1

Definition at line 28 of file `RectangularMapCsvToInputFiles.cpp`.

Referenced by `main()`.

8.86.1.2 #define ERR_MSG "An error occurred: "

Definition at line 29 of file RectangularMapCsvToInputFiles.cpp.

Referenced by isValidNrOfConcentrationsForPosition(), main(), and printWrongParameters().

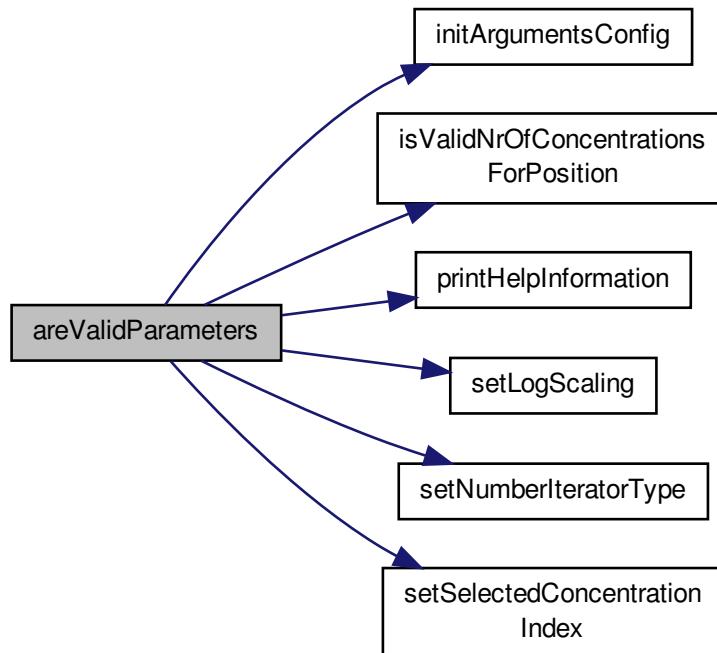
8.86.2 Function Documentation

8.86.2.1 bool areValidParameters (string & *inputFilepath*, string & *outputFilename*, unsigned int & *height*, unsigned int & *width*, unsigned int & *nrOfConcentrationsForPosition*, unsigned int & *selectedConcentrationIndex*, bool & *useLogScaling*, NumberIteratorType & *numberIteratorType*, int *argc*, char ** *argv*)

Definition at line 92 of file RectangularMapCsvToInputFiles.cpp.

References initArgumentsConfig(), isValidNrOfConcentrationsForPosition(), printHelpInformation(), setLogScaling(), setNumberIteratorType(), and setSelectedConcentrationIndex().

Here is the call graph for this function:



8.86.2.2 po::variables_map initArgumentsConfig (po::options_description & *usageDescription*, int *argc*, char ** *argv*)

Definition at line 33 of file RectangularMapCsvToInputFiles.cpp.

8.86.2.3 bool isValidNrOfConcentrationsForPosition (const po::variables_map & *vm*, unsigned int & *nrOfConcentrationsForPosition*)

Definition at line 77 of file RectangularMapCsvToInputFiles.cpp.

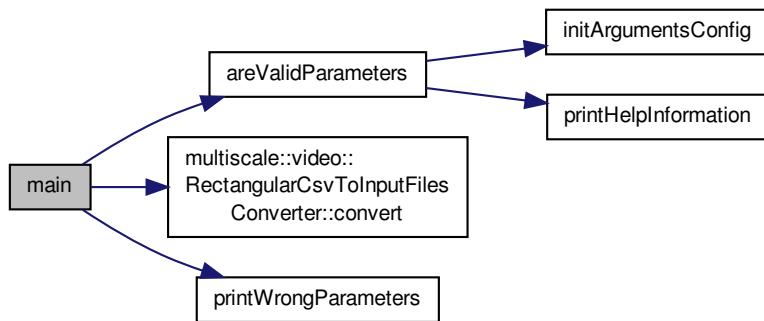
References ERR_MSG.

8.86.2.4 int main (int argc, char ** argv)

Definition at line 139 of file RectangularMapCsvToInputFiles.cpp.

References areValidParameters(), multiscale::video::RectangularCsvToInputFilesConverter::convert(), ERR_CODE, ERR_MSG, printWrongParameters(), and multiscale::STANDARD.

Here is the call graph for this function:



8.86.2.5 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 51 of file RectangularMapCsvToInputFiles.cpp.

8.86.2.6 void printWrongParameters ()

Definition at line 56 of file RectangularMapCsvToInputFiles.cpp.

References ERR_MSG.

8.86.2.7 void setLogScaling (const po::variables_map & vm, bool & useLogScaling)

Definition at line 72 of file RectangularMapCsvToInputFiles.cpp.

8.86.2.8 void setNumberIteratorType (const po::variables_map & vm, NumberIteratorType & numberIteratorType)

Definition at line 62 of file RectangularMapCsvToInputFiles.cpp.

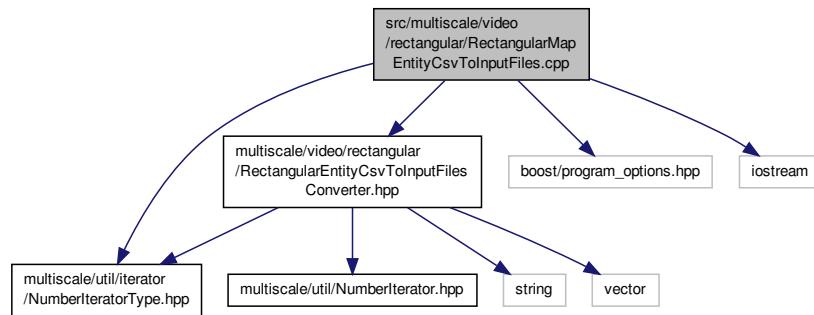
References multiscale::LEXICOGRAPHIC.

8.86.2.9 void setSelectedConcentrationIndex (const po::variables_map & vm, unsigned int & selectedConcentrationIndex)

Definition at line 67 of file RectangularMapCsvToInputFiles.cpp.

8.87 src/multiscale/video/rectangular/RectangularMapEntityCsvToInputFiles.cpp File Reference

```
#include "multiscale/video/rectangular/RectangularEntityCsvToInputFiles-
Converter.hpp"
#include "multiscale/util/iterator/NumberIteratorType.hpp"
#include <boost/program_options.hpp>
#include <iostream>
Include dependency graph for RectangularMapEntityCsvToInputFiles.cpp:
```



Macros

- `#define ERR_CODE 1`
- `#define ERR_MSG "An error occurred: "`

Functions

- `po::variables_map initArgumentsConfig (po::options_description &usageDescription, int argc, char **argv)`
- `void printHelpInformation (const po::variables_map &vm, const po::options_description &usageDescription)`
- `void printWrongParameters ()`
- `void setNumberIteratorType (const po::variables_map &vm, NumberIteratorType &numberIteratorType)`
- `bool areValidParameters (string &inputFilepath, string &outputFilename, unsigned int &height, unsigned int &width, unsigned int &nrOfEntities, unsigned int &maxPileup, NumberIteratorType &numberIteratorType, int argc, char **argv)`
- `int main (int argc, char **argv)`

8.87.1 Macro Definition Documentation

8.87.1.1 #define ERR_CODE 1

Definition at line 28 of file `RectangularMapEntityCsvToInputFiles.cpp`.

Referenced by `main()`.

8.87.1.2 #define ERR_MSG "An error occurred: "

Definition at line 29 of file `RectangularMapEntityCsvToInputFiles.cpp`.

Referenced by `main()`, and `printWrongParameters()`.

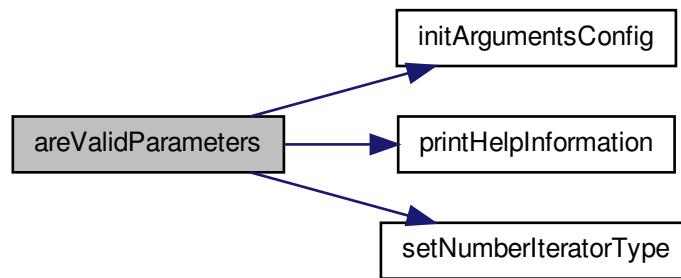
8.87.2 Function Documentation

8.87.2.1 `bool areValidParameters (string & inputfilepath, string & outputfilename, unsigned int & height, unsigned int & width, unsigned int & nrOfEntities, unsigned int & maxPileup, NumberIteratorType & numberIteratorType, int argc, char ** argv)`

Definition at line 66 of file RectangularMapEntityCsvToInputFiles.cpp.

References initArgumentsConfig(), printHelpInformation(), and setNumberIteratorType().

Here is the call graph for this function:



8.87.2.2 `po::variables_map initArgumentsConfig (po::options_description & usageDescription, int argc, char ** argv)`

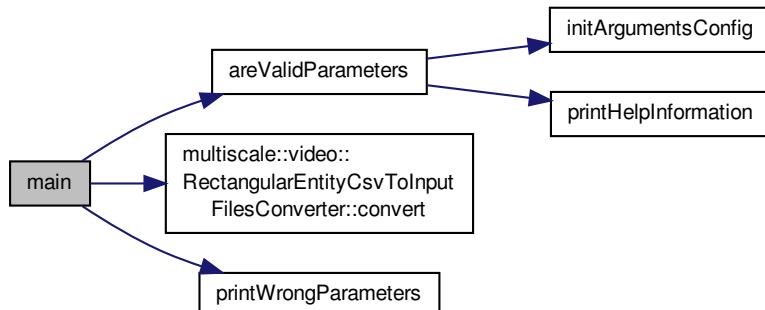
Definition at line 33 of file RectangularMapEntityCsvToInputFiles.cpp.

8.87.2.3 `int main (int argc, char ** argv)`

Definition at line 104 of file RectangularMapEntityCsvToInputFiles.cpp.

References areValidParameters(), multiscale::video::RectangularEntityCsvToInputFilesConverter::convert(), ERR_CODE, ERR_MSG, printWrongParameters(), and multiscale::STANDARD.

Here is the call graph for this function:



8.87.2.4 void printHelpInformation (const po::variables_map & vm, const po::options_description & usageDescription)

Definition at line 50 of file RectangularMapEntityCsvToInputFiles.cpp.

8.87.2.5 void printWrongParameters ()

Definition at line 55 of file RectangularMapEntityCsvToInputFiles.cpp.

References ERR_MSG.

8.87.2.6 void setNumberIteratorType (const po::variables_map & vm, NumberIteratorType & numberIteratorType)

Definition at line 61 of file RectangularMapEntityCsvToInputFiles.cpp.

References multiscale::LEXICOGRAPHIC.