

libopenvgr
0.9.0

作成 : Doxygen 1.6.1

Tue Dec 20 13:47:21 2011

Contents

1	ネームスペース索引	1
1.1	ネームスペース一覧	1
2	データ構造索引	3
2.1	クラス階層	3
3	データ構造索引	5
3.1	データ構造	5
4	ファイル索引	7
4.1	ファイル一覧	7
5	ネームスペース	9
5.1	ネームスペース ovgr	9
5.1.1	型定義	11
5.1.1.1	feature_index_list_t	11
5.1.1.2	feature_indexes_t	11
5.1.1.3	feature_list_t	11
5.1.1.4	feature_map_t	11
5.1.1.5	Segment2D	11
5.1.2	列挙型	11
5.1.2.1	CorrespondingCriteria	11
5.1.3	関数	11
5.1.3.1	calc_crossing_point	11
5.1.3.2	calc_ellipse_rr	12
5.1.3.3	calc_epipolar_line	12
5.1.3.4	calc_epipole	12

5.1.3.5	<code>calc_essential_matrix</code>	12
5.1.3.6	<code>calc_fundamental_matrix</code>	12
5.1.3.7	<code>calc_homography</code>	12
5.1.3.8	<code>calc_infinite_homography</code>	12
5.1.3.9	<code>calc_line_joining_points</code>	12
5.1.3.10	<code>create_new_features_from_old_one</code>	13
5.1.3.11	<code>create_old_features_from_new_one</code>	13
5.1.3.12	<code>distance_from_line</code>	13
5.1.3.13	<code>draw_EllipseFeatures</code>	13
5.1.3.14	<code>draw_VertexFeatures</code>	13
5.1.3.15	<code>extractFeatures</code>	13
5.1.3.16	<code>filter_corresponding_set</code>	13
5.1.3.17	<code>ImageToFeature2D</code>	13
5.1.3.18	<code>make_corresponding_pairs</code>	14
5.1.3.19	<code>solve_quad_eq</code>	14
6	データ構造	15
6.1	構造体 <code>_Ellipse_</code>	15
6.1.1	構造体	15
6.1.1.1	<code>a</code>	16
6.1.1.2	<code>axis</code>	16
6.1.1.3	<code>center</code>	16
6.1.1.4	<code>coef</code>	16
6.1.1.5	<code>cubic</code>	16
6.1.1.6	<code>eval</code>	16
6.1.1.7	<code>M</code>	16
6.1.1.8	<code>maxError</code>	16
6.1.1.9	<code>meanError</code>	16
6.1.1.10	<code>neval</code>	16
6.1.1.11	<code>offset</code>	16
6.1.1.12	<code>P00</code>	17
6.1.1.13	<code>P01</code>	17
6.1.1.14	<code>P10</code>	17
6.1.1.15	<code>P11</code>	17

6.1.1.16	rad	17
6.2	構造体 _ellipse_arc_	18
6.2.1	構造体	18
6.2.1.1	f2Ds	18
6.2.1.2	goal	19
6.2.1.3	ntrack	19
6.2.1.4	start	19
6.3	構造体 _ellipse_arc_list_	20
6.3.1	構造体	20
6.3.1.1	arc	20
6.3.1.2	n	21
6.4	構造体 _offset_prop_	22
6.4.1	構造体	22
6.4.1.1	d	22
6.4.1.2	mode	22
6.5	構造体 _param_ellipse_IW_	23
6.5.1	構造体	23
6.5.1.1	Condition	23
6.5.1.2	MinLength	23
6.5.1.3	MinSD	23
6.5.1.4	MinShortRadPost	23
6.5.1.5	MinShortRadPrev	24
6.5.1.6	OffsetMode	24
6.5.1.7	PostMinLength	24
6.5.1.8	ShortenEllipseMerging	24
6.5.1.9	SwLineEllipse	24
6.5.1.10	SwOldMergeFunc	24
6.5.1.11	ThMaxError	24
6.5.1.12	ThMaxErrorMerging	24
6.5.1.13	ThMeanError	24
6.5.1.14	ThMeanErrorMerging	24
6.6	構造体 _recogImage	25
6.6.1	構造体	25
6.6.1.1	bytePerPixel	25

6.6.1.2	colsize	25
6.6.1.3	pixel	25
6.6.1.4	rowsize	25
6.7	構造体 <code>_SumSet_</code>	26
6.7.1	構造体	26
6.7.1.1	n	26
6.7.1.2	x	26
6.7.1.3	x2	26
6.7.1.4	x2y	26
6.7.1.5	x2y2	27
6.7.1.6	x3	27
6.7.1.7	x3y	27
6.7.1.8	x4	27
6.7.1.9	xy	27
6.7.1.10	xy2	27
6.7.1.11	xy3	27
6.7.1.12	y	27
6.7.1.13	y2	27
6.7.1.14	y3	27
6.7.1.15	y4	28
6.8	構造体 テンプレート <code>ovgr::Array< T, N ></code>	29
6.8.1	型定義	29
6.8.1.1	data_type	29
6.8.2	関数	29
6.8.2.1	operator[]	29
6.8.2.2	operator[]	29
6.8.3	構造体	29
6.8.3.1	elem	30
6.9	構造体 <code>CalibParam</code>	31
6.9.1	説明	32
6.9.2	構造体	32
6.9.2.1	CameraL	32
6.9.2.2	CameraR	32
6.9.2.3	CameraV	32

6.9.2.4	colsize	32
6.9.2.5	numOfCameras	32
6.9.2.6	rowsize	32
6.10	構造体 CameraParam	33
6.10.1	説明	33
6.10.2	構造体	34
6.10.2.1	Distortion	34
6.10.2.2	intrinsicMatrix	34
6.10.2.3	Position	34
6.10.2.4	Rotation	34
6.10.2.5	Translation	34
6.11	構造体 Circle	35
6.11.1	説明	36
6.11.2	構造体	36
6.11.2.1	label	36
6.11.2.2	n	36
6.11.2.3	normal	36
6.11.2.4	numOfTracePoints	36
6.11.2.5	projected	36
6.11.2.6	radius	36
6.11.2.7	side	36
6.11.2.8	tPose	37
6.11.2.9	tracepoints	37
6.11.2.10	transformed	37
6.12	構造体 CircleCandidate	38
6.12.1	説明	38
6.12.2	構造体	38
6.12.2.1	center	38
6.12.2.2	normal	38
6.12.2.3	radius	38
6.12.2.4	valid	39
6.13	構造体 ovgr::ConicFeature2D	40
6.13.1	構造体	40
6.13.1.1	coef	41

6.14 構造体 <code>ovgr::CorrespondenceThresholds</code>	42
6.14.1 コンストラクタとデストラクタ	42
6.14.1.1 <code>CorrespondenceThresholds</code>	42
6.14.2 構造体	42
6.14.2.1 <code>ellipse_tolerance</code>	42
6.14.2.2 <code>vertex_tolerance</code>	42
6.15 構造体 <code>ovgr::CorrespondingPair</code>	43
6.15.1 構造体	43
6.15.1.1 <code>ellipse</code>	43
6.15.1.2 <code>vertex</code>	43
6.16 構造体 <code>ovgr::CorrespondingSet</code>	44
6.16.1 構造体	44
6.16.1.1 <code>ellipse</code>	44
6.16.1.2 <code>vertex</code>	44
6.17 構造体 <code>Data_2D</code>	45
6.17.1 構造体	45
6.17.1.1 <code>col</code>	45
6.17.1.2 <code>row</code>	45
6.18 構造体 <code>DistortionParam</code>	46
6.18.1 説明	46
6.18.2 構造体	46
6.18.2.1 <code>k1</code>	46
6.18.2.2 <code>k2</code>	46
6.18.2.3 <code>k3</code>	46
6.18.2.4 <code>p1</code>	47
6.18.2.5 <code>p2</code>	47
6.19 構造体 <code>ovgr::EllipseFeature</code>	48
6.19.1 構造体	49
6.19.1.1 <code>axis</code>	49
6.19.1.2 <code>center</code>	49
6.19.1.3 <code>theta</code>	49
6.20 構造体 <code>EllipseGroup</code>	50
6.20.1 説明	50
6.20.2 構造体	50

6.20.2.1	groupCenter	50
6.20.2.2	groupNums	50
6.20.2.3	nCurrNum	50
6.21	構造体 テンプレート ovgr::EqualOp< T >	51
6.21.1	関数	51
6.21.1.1	operator()	51
6.22	構造体 tag_Wireframe::Face	52
6.22.1	説明	52
6.22.2	コンストラクタとデストラクタ	52
6.22.2.1	Face	52
6.22.3	構造体	52
6.22.3.1	normal	52
6.22.3.2	segment_id	53
6.23	構造体 ovgr::Feature2D	54
6.23.1	構造体	54
6.23.1.1	error	54
6.23.1.2	segment	54
6.24	構造体 Feature2D_old	55
6.24.1	説明	57
6.24.2	構造体	57
6.24.2.1	all	57
6.24.2.2	arclist	57
6.24.2.3	axis	57
6.24.2.4	center	57
6.24.2.5	coef	57
6.24.2.6	direction	57
6.24.2.7	end	57
6.24.2.8	endPoint	58
6.24.2.9	endSPoint	58
6.24.2.10	error	58
6.24.2.11	ev	58
6.24.2.12	lineAngle	58
6.24.2.13	lineLength	58
6.24.2.14	lineLength1	58

6.24.2.15	lineLength2	58
6.24.2.16	middleSPoint	58
6.24.2.17	nPoints	58
6.24.2.18	nTrack	58
6.24.2.19	start	59
6.24.2.20	startPoint	59
6.24.2.21	startSPoint	59
6.24.2.22	type	59
6.25	構造体 ovgr::Features2D	60
6.25.1	構造体	60
6.25.1.1	ellipse	60
6.25.1.2	vertex	60
6.26	構造体 Features2D_old	61
6.26.1	説明	62
6.26.2	構造体	62
6.26.2.1	feature	62
6.26.2.2	nAlloc	62
6.26.2.3	nFeature	62
6.26.2.4	nTrack	62
6.26.2.5	track	62
6.27	構造体 Features3D	63
6.27.1	説明	64
6.27.2	構造体	64
6.27.2.1	calib	64
6.27.2.2	Circles	64
6.27.2.3	edge	64
6.27.2.4	numOfCircles	64
6.27.2.5	numOfVertices	64
6.27.2.6	pointCounts	64
6.27.2.7	traceCounts	65
6.27.2.8	Vertices	65
6.27.2.9	wireframe	65
6.28	構造体 ovgr::LineFeature2D	66
6.28.1	構造体	67

6.28.1.1	coef	67
6.28.1.2	end	67
6.28.1.3	length	67
6.28.1.4	start	67
6.29	構造体 Match3Dresults	68
6.29.1	説明	68
6.29.2	構造体	68
6.29.2.1	error	68
6.29.2.2	numOfResults	69
6.29.2.3	Results	69
6.30	構造体 MatchResult	70
6.30.1	説明	70
6.30.2	構造体	70
6.30.2.1	cpoint	71
6.30.2.2	mat	71
6.30.2.3	model	71
6.30.2.4	n	71
6.30.2.5	npoint	71
6.30.2.6	scene	71
6.30.2.7	score	71
6.30.2.8	type	71
6.30.2.9	vec	71
6.31	構造体 ModelFileInfo	72
6.31.1	説明	72
6.31.2	構造体	72
6.31.2.1	model	72
6.31.2.2	modelNum	73
6.32	構造体 ModelFileInfoNode	74
6.32.1	説明	74
6.32.2	構造体	74
6.32.2.1	id	74
6.32.2.2	path	74
6.33	構造体 P2D	75
6.33.1	説明	75

6.33.2 構造体	75
6.33.2.1 colrow	75
6.34 構造体 P3D	76
6.34.1 説明	76
6.34.2 構造体	76
6.34.2.1 xyz	76
6.35 構造体 Parameters	77
6.35.1 説明	78
6.35.2 構造体	78
6.35.2.1 colsize	78
6.35.2.2 dbgdisp	78
6.35.2.3 dbgimag	78
6.35.2.4 dbgtex	78
6.35.2.5 feature2D	78
6.35.2.6 imgsize	78
6.35.2.7 match	78
6.35.2.8 outputCandNum	78
6.35.2.9 pairing	78
6.35.2.10 paramEIW	79
6.35.2.11 rowsize	79
6.35.2.12 stereo	79
6.36 構造体 ParametersFeature2D	80
6.36.1 説明	81
6.36.2 構造体	81
6.36.2.1 edgeDetectFunction	82
6.36.2.2 edgeStrength	82
6.36.2.3 max_distance_ellipse_grouping	82
6.36.2.4 max_distance_ellipse_pairing	82
6.36.2.5 max_distance_end_points	82
6.36.2.6 max_distance_similar_line	82
6.36.2.7 max_flatness_ellipse	82
6.36.2.8 max_length_delete_line	82
6.36.2.9 max_length_ellipse_axisL	82
6.36.2.10 maxErrorofConicFit	82

6.36.2.11	maxErrorofLineFit	82
6.36.2.12	min_distance_ellipse_pairing	83
6.36.2.13	min_filling_ellipse	83
6.36.2.14	min_length_ellipse_axis	83
6.36.2.15	min_length_ellipse_axisL	83
6.36.2.16	min_length_ellipse_axisS	83
6.36.2.17	min_length_hyperbola_data	83
6.36.2.18	min_length_hyperbola_vector	83
6.36.2.19	min_length_line	83
6.36.2.20	min_radian_hyperbola	83
6.36.2.21	minFragment	83
6.36.2.22	no_search_features	83
6.36.2.23	overlapRatioCircle	84
6.36.2.24	overlapRatioLine	84
6.37	構造体 ParametersMatch	85
6.37.1	説明	85
6.37.2	構造体	85
6.37.2.1	pdist	85
6.37.2.2	tolerance1	85
6.37.2.3	tolerance2	85
6.38	構造体 ParmetersStereo	86
6.38.1	説明	86
6.38.2	構造体	86
6.38.2.1	amax	86
6.38.2.2	amin	87
6.38.2.3	depf	87
6.38.2.4	depn	87
6.38.2.5	ethr	87
6.38.2.6	ndif	87
6.38.2.7	rdif	87
6.39	構造体 ovgr::Point2D	88
6.39.1	コンストラクタとデストラクタ	88
6.39.1.1	Point2D	88
6.39.1.2	Point2D	88

6.39.1.3	Point2D	88
6.39.2	関数	88
6.39.2.1	operator cv::Point_< T >	89
6.39.3	構造体	89
6.39.3.1	x	89
6.39.3.2	y	89
6.40	クラス ovgr::PointsOnEllipse	90
6.41	クラス ovgr::PointsOnLine	91
6.41.1	コンストラクタとデストラクタ	91
6.41.1.1	PointsOnLine	91
6.41.2	関数	91
6.41.2.1	next	91
6.41.2.2	x	91
6.41.2.3	y	92
6.41.3	構造体	92
6.41.3.1	m_dx	92
6.41.3.2	m_dy	92
6.41.3.3	m_e	92
6.41.3.4	m_ex	92
6.41.3.5	m_ey	92
6.41.3.6	m_x	92
6.41.3.7	m_xinc	92
6.41.3.8	m_y	92
6.41.3.9	m_yinc	92
6.42	構造体 RTVCM_Box	94
6.42.1	説明	94
6.42.2	構造体	94
6.42.2.1	n	94
6.42.2.2	nVertex	95
6.42.2.3	reserved	95
6.42.2.4	Rotate	95
6.42.2.5	Trans	95
6.42.2.6	x	95
6.42.2.7	y	95

6.42.2.8	z	95
6.43	構造体 RTVCM_Circle	96
6.43.1	説明	96
6.43.2	構造体	96
6.43.2.1	center	96
6.43.2.2	n	96
6.43.2.3	ncylinder	97
6.43.2.4	normal	97
6.43.2.5	radius	97
6.43.2.6	reserved	97
6.44	構造体 RTVCM_Cylinder	98
6.44.1	説明	98
6.44.2	構造体	98
6.44.2.1	height	98
6.44.2.2	n	99
6.44.2.3	nCircle	99
6.44.2.4	radius	99
6.44.2.5	reserved	99
6.44.2.6	Rotate	99
6.44.2.7	Trans	99
6.45	構造体 RTVCM_Vertex	100
6.45.1	説明	100
6.45.2	構造体	100
6.45.2.1	angle	100
6.45.2.2	endpoint1	101
6.45.2.3	endpoint2	101
6.45.2.4	n	101
6.45.2.5	nbox	101
6.45.2.6	position	101
6.45.2.7	reserved	101
6.46	構造体 RTVertexCircleModel	102
6.46.1	説明	103
6.46.2	構造体	103
6.46.2.1	box	103

6.46.2.2	circle	103
6.46.2.3	cylinder	103
6.46.2.4	depth	104
6.46.2.5	gravity	104
6.46.2.6	height	104
6.46.2.7	label	104
6.46.2.8	n	104
6.46.2.9	nbox	104
6.46.2.10	ncircle	104
6.46.2.11	ncylinder	104
6.46.2.12	nvertex	104
6.46.2.13	radius	104
6.46.2.14	reserved	105
6.46.2.15	vertex	105
6.46.2.16	width	105
6.47	構造体 tag_Wireframe::Segment	106
6.47.1	説明	106
6.47.2	コンストラクタとデストラクタ	106
6.47.2.1	Segment	106
6.47.2.2	Segment	106
6.47.3	構造体	106
6.47.3.1	vertex_id	106
6.48	構造体 StereoCalib	107
6.48.1	説明	107
6.48.2	構造体	107
6.48.2.1	baselineLR	107
6.48.2.2	baselineLV	107
6.48.2.3	baselineRV	108
6.48.2.4	height	108
6.48.2.5	numOfCameras	108
6.48.2.6	width	108
6.49	構造体 tag_Wireframe	109
6.49.1	説明	109
6.49.2	構造体	109

6.49.2.1	face	109
6.49.2.2	num_vertices	110
6.49.2.3	segment	110
6.49.2.4	vertex	110
6.50	構造体 Trace	111
6.50.1	説明	111
6.50.2	構造体	111
6.50.2.1	colrow	111
6.50.2.2	direction	112
6.50.2.3	edge	112
6.50.2.4	label	112
6.50.2.5	peakcr	112
6.50.2.6	search	112
6.50.2.7	weight	112
6.50.2.8	xyz	112
6.51	構造体 Track	113
6.51.1	説明	113
6.51.2	構造体	113
6.51.2.1	nPoint	113
6.51.2.2	offset	113
6.51.2.3	Point	113
6.52	クラス テンプレート ovgr::VariableWatcher< T, Equal >	114
6.52.1	コンストラクタとデストラクタ	114
6.52.1.1	VariableWatcher	114
6.52.2	関数	114
6.52.2.1	is_changed	114
6.53	構造体 Vertex	115
6.53.1	説明	116
6.53.2	構造体	116
6.53.2.1	angle	116
6.53.2.2	direction1	116
6.53.2.3	direction2	116
6.53.2.4	endpoint1	116
6.53.2.5	endpoint2	116

6.53.2.6	label	117
6.53.2.7	n	117
6.53.2.8	numOfTracePoints	117
6.53.2.9	projected	117
6.53.2.10	side	117
6.53.2.11	tPose	117
6.53.2.12	tracepoints	117
6.53.2.13	transformed	117
6.54	構造体 VertexCandidate	118
6.54.1	説明	119
6.54.2	構造体	119
6.54.2.1	angle	119
6.54.2.2	endpoint1	119
6.54.2.3	endpoint2	119
6.54.2.4	len1	119
6.54.2.5	len2	119
6.54.2.6	n1	119
6.54.2.7	n2	119
6.54.2.8	n3	119
6.54.2.9	position	119
6.54.2.10	valid	120
6.54.2.11	vector1	120
6.54.2.12	vector2	120
6.55	構造体 ovgr::VertexFeature	121
6.55.1	構造体	122
6.55.1.1	end	122
6.55.1.2	length	122
6.55.1.3	line_coef	122
6.55.1.4	mid	122
6.55.1.5	start	122
7	ファイル	123
7.1	calib.cpp	123
7.1.1	説明	124

7.1.2	関数	124
7.1.2.1	backprojectPoint	124
7.1.2.2	distortPosition	124
7.1.2.3	projectPoint	124
7.1.2.4	undistortPosition	124
7.2	calib.h	125
7.2.1	説明	126
7.2.2	関数	126
7.2.2.1	backprojectPoint	126
7.2.2.2	distortPosition	126
7.2.2.3	projectPoint	126
7.2.2.4	undistortPosition	126
7.3	calibUtil.cpp	127
7.3.1	関数	127
7.3.1.1	setCalibFromCameraImage	127
7.4	calibUtil.h	128
7.4.1	説明	128
7.4.2	関数	128
7.4.2.1	setCalibFromCameraImage	129
7.5	circle.cpp	130
7.5.1	説明	131
7.5.2	関数	131
7.5.2.1	calc_3d_axes_of_circle	131
7.5.2.2	reconstruct_ellipse2D_to_circle3D	131
7.6	circle.h	132
7.6.1	説明	133
7.6.2	関数	133
7.6.2.1	calc_3d_axes_of_circle	133
7.6.2.2	reconstruct_ellipse2D_to_circle3D	133
7.7	common.h	134
7.7.1	説明	135
7.7.2	マクロ定義	135
7.7.2.1	INVISIBLE	135
7.7.2.2	NO_SEARCH_ELLIPSE	135

7.7.2.3	NO_SEARCH_VERTEX	135
7.7.2.4	VISIBLE	135
7.7.2.5	VISION_EPS	135
7.7.3	列挙型	135
7.7.3.1	StereoPairing	135
7.8	conic.cpp	136
7.8.1	説明	136
7.8.2	関数	137
7.8.2.1	addConicSum	137
7.8.2.2	clearConicSum	137
7.8.2.3	distanceConic	137
7.8.2.4	fitConic	137
7.8.2.5	fitConicAny	137
7.8.2.6	getConicProperty	137
7.8.2.7	getConicType	137
7.8.2.8	subConicSum	137
7.9	conic.h	138
7.9.1	説明	139
7.9.2	列挙型	139
7.9.2.1	ConicType	139
7.9.3	関数	139
7.9.3.1	addConicSum	139
7.9.3.2	clearConicSum	139
7.9.3.3	distanceConic	140
7.9.3.4	fitConic	140
7.9.3.5	fitConicAny	140
7.9.3.6	getConicProperty	140
7.9.3.7	getConicType	140
7.9.3.8	subConicSum	140
7.10	constants.hpp	141
7.11	correspondence.cpp	142
7.12	correspondence.hpp	143
7.13	debugutil.cpp	145
7.13.1	説明	146

7.13.2 関数	146
7.13.2.1 drawCircleCandidate	146
7.13.2.2 drawDetectedEllipses	146
7.13.2.3 drawDetectedLines	146
7.13.2.4 drawDetectedVertices	146
7.13.2.5 drawEdgeImage	147
7.13.2.6 drawInputImage	147
7.13.2.7 drawTrackPoints	147
7.13.2.8 printVertex	147
7.14 debugutil.h	148
7.14.1 説明	149
7.14.2 関数	149
7.14.2.1 drawCircleCandidate	149
7.14.2.2 drawDetectedEllipses	149
7.14.2.3 drawDetectedLines	149
7.14.2.4 drawDetectedVertices	150
7.14.2.5 drawEdgeImage	150
7.14.2.6 drawInputImage	150
7.14.2.7 printVertex	150
7.15 drawing.cpp	151
7.16 drawing.hpp	152
7.17 ellipse_to_ylimit.cpp	153
7.17.1 関数	153
7.17.1.1 ellipse_to_ylimit	153
7.18 ellipseIW.h	154
7.18.1 マクロ定義	155
7.18.1.1 CHECK_ELLIPSE_NG	155
7.18.1.2 CHECK_ELLIPSE_OK	155
7.18.1.3 MERGE_ELLIPSE_NG	155
7.18.1.4 MERGE_ELLIPSE_OK	156
7.18.1.5 NAXIS	156
7.18.1.6 NCOEFCUBIC	156
7.18.1.7 NDIM2	156
7.18.1.8 NDIM3	156

7.18.1.9	NDIM_CONIC_FULL	156
7.18.1.10	NDIM_CONIC_HALF	156
7.18.1.11	SEARCH_FEATURES2_NG	156
7.18.1.12	SEARCH_FEATURES2_OK	156
7.18.2	型定義	156
7.18.2.1	Ellipse	156
7.18.2.2	OffsetProp	157
7.18.2.3	SumSet	157
7.18.3	関数	157
7.18.3.1	addArcSum	157
7.18.3.2	avec_to_ellipse	157
7.18.3.3	check_ellipse_cond	157
7.18.3.4	distanceAConic	157
7.18.3.5	merge_ellipse	157
7.18.3.6	mod_nPoint	157
7.18.3.7	P_to_avec_and_fix	157
7.18.3.8	searchEllipseIW	158
7.18.3.9	sum_to_P_dynamic	158
7.19	extractEdge.cpp	159
7.19.1	説明	159
7.19.2	マクロ定義	160
7.19.2.1	Edge	160
7.19.2.2	Gray	160
7.19.3	関数	160
7.19.3.1	extractEdge	160
7.19.3.2	extractEdge_new	160
7.20	extractEdge.h	161
7.20.1	説明	162
7.20.2	マクロ定義	162
7.20.2.1	EE6	162
7.20.2.2	EE7	162
7.20.2.3	EEcandidate	162
7.20.2.4	EEerasedThin	162
7.20.2.5	EEextended	162

7.20.2.6	EEnotEdge	162
7.20.2.7	EEnotSearched	162
7.20.2.8	EEsearchedLarge	162
7.20.2.9	EEsearchedSmall	163
7.20.3	関数	163
7.20.3.1	extractEdge	163
7.20.3.2	extractEdge_new	163
7.21	extractFeature.cpp	164
7.21.1	説明	165
7.21.2	型定義	165
7.21.2.1	Points	165
7.21.2.2	PointSet	165
7.21.2.3	support_index_t	165
7.22	extractFeature.hpp	166
7.23	extractFeature_old.cpp	168
7.23.1	マクロ定義	169
7.23.1.1	CONIC_MATCH_NG	169
7.23.1.2	CONIC_MATCH_OK	169
7.23.1.3	Work	169
7.23.2	型定義	169
7.23.2.1	Feature_List	169
7.23.3	関数	169
7.23.3.1	destructFeatures	169
7.23.3.2	expandFeatures	170
7.23.3.3	extractFeatures_old	170
7.23.3.4	ImageToFeature2D_old	170
7.23.3.5	mark_similar_lines	170
7.24	extractFeature_old.h	171
7.24.1	説明	172
7.24.2	型定義	172
7.24.2.1	EllipseArc	172
7.24.2.2	EllipseArcList	173
7.24.3	関数	173
7.24.3.1	destructFeatures	173

7.24.3.2	expandFeatures	173
7.24.3.3	ImageToFeature2D_old	173
7.25	geometry.cpp	174
7.26	geometry.hpp	175
7.27	imageUtil.cpp	177
7.27.1	説明	177
7.27.2	関数	178
7.27.2.1	convertTimedMultiCameraImageToRecogImage	178
7.27.2.2	freeConvertedRecogImage	178
7.28	imageUtil.h	179
7.28.1	説明	179
7.28.2	関数	180
7.28.2.1	convertTimedMultiCameraImageToRecogImage	180
7.28.2.2	freeConvertedRecogImage	180
7.29	local.h	181
7.30	match3Dfeature.cpp	182
7.30.1	説明	183
7.30.2	関数	183
7.30.2.1	freeFeatures3D	183
7.30.2.2	freeMatch3Dresults	183
7.30.2.3	matchFeatures3D	183
7.30.2.4	shrinkMatch3Dresults	183
7.31	match3Dfeature.h	184
7.31.1	説明	186
7.31.2	型定義	186
7.31.2.1	Wireframe	186
7.31.3	列挙型	186
7.31.3.1	m3df_feature_label	186
7.31.3.2	m3df_side	187
7.31.4	関数	187
7.31.4.1	freeFeatures3D	187
7.31.4.2	freeMatch3Dresults	187
7.31.4.3	matchFeatures3D	187
7.31.4.4	matchPairedCircles	187

7.31.4.5	<code>shrinkMatch3Dresults</code>	187
7.32	<code>mathmisc.cpp</code>	188
7.33	<code>mathmisc.hpp</code>	189
7.34	<code>merge_ellipse.cpp</code>	190
7.34.1	マクロ定義	191
7.34.1.1	<code>ADD_NEW_MULTI_ELLIPSE_NG</code>	191
7.34.1.2	<code>ADD_NEW_MULTI_ELLIPSE_OK</code>	191
7.34.1.3	<code>ALL_REF_NG</code>	191
7.34.1.4	<code>ALL_REF_OK</code>	191
7.34.1.5	<code>ANOTHER_EXIST_ERROR</code>	191
7.34.1.6	<code>ANOTHER_EXIST_NG</code>	192
7.34.1.7	<code>ANOTHER_EXIST_OK</code>	192
7.34.1.8	<code>ELLIPSE_TERMINAL_PART</code>	192
7.34.1.9	<code>ELLIPSE_TERMINAL_WHOLE</code>	192
7.34.1.10	<code>INITIAL_ARRAYS_NG</code>	192
7.34.1.11	<code>INITIAL_ARRAYS_OK</code>	192
7.34.1.12	<code>NOELLIPSE</code>	192
7.34.1.13	<code>NTERMINAL</code>	192
7.34.1.14	<code>REF_NG</code>	192
7.34.1.15	<code>REF_OK</code>	192
7.34.1.16	<code>REF_SELF</code>	192
7.34.1.17	<code>SEARCH_ANOTHER_ARC_NG</code>	193
7.34.1.18	<code>SEARCH_ANOTHER_ARC_OK</code>	193
7.34.1.19	<code>SIGN_DEC</code>	193
7.34.1.20	<code>SIGN_INC</code>	193
7.34.1.21	<code>SIGN_UNDEF</code>	193
7.34.1.22	<code>SIGN_ZERO</code>	193
7.34.1.23	<code>TRY_ELLIPSE_CONTINUE</code>	193
7.34.1.24	<code>TRY_ELLIPSE_REGISTER</code>	193
7.34.1.25	<code>TRY_ELLIPSE_TERMINATE</code>	193
7.34.2	型定義	193
7.34.2.1	<code>EllipseTerminal</code>	193
7.34.2.2	<code>MergeEllipseArrays</code>	194
7.34.3	関数	194

7.34.3.1	merge_ellipse	194
7.35	modelFileio.cpp	195
7.35.1	関数	195
7.35.1.1	loadModelFile	195
7.36	modelFileio.h	196
7.36.1	説明	196
7.36.2	関数	196
7.36.2.1	loadModelFile	197
7.37	modelListFileIO.cpp	198
7.37.1	関数	198
7.37.1.1	clearModelFileInfo	198
7.37.1.2	loadModelListFile	198
7.38	modelListFileIO.h	199
7.38.1	説明	199
7.38.2	マクロ定義	199
7.38.2.1	MAX_PATH	200
7.38.3	関数	200
7.38.3.1	clearModelFileInfo	200
7.38.3.2	loadModelListFile	200
7.39	modelpoints.cpp	201
7.39.1	説明	202
7.39.2	関数	202
7.39.2.1	calcEvaluationValue2DMultiCameras	202
7.39.2.2	drawModelPoints	202
7.39.2.3	getPropertyVector	202
7.39.2.4	makeModelPoints	202
7.40	modelpoints.h	203
7.40.1	説明	204
7.40.2	型定義	204
7.40.2.1	plot_t	204
7.40.3	関数	204
7.40.3.1	calcEvaluationValue2DMultiCameras	204
7.40.3.2	drawModelPoints	204
7.40.3.3	getPropertyVector	204

7.40.3.4	<code>isValidPixelPosition</code>	204
7.40.3.5	<code>makeModelPoints</code>	205
7.41	<code>pairedcircle.cpp</code>	206
7.41.1	説明	206
7.41.2	関数	207
7.41.2.1	<code>matchPairedCircles</code>	207
7.42	<code>paramEllipseIW.h</code>	208
7.42.1	マクロ定義	209
7.42.1.1	<code>DEF_PARAME_CONDITION</code>	209
7.42.1.2	<code>DEF_PARAME_MIN_LENGTH</code>	209
7.42.1.3	<code>DEF_PARAME_MIN_SD</code>	209
7.42.1.4	<code>DEF_PARAME_MIN_SHORT_RAD_POST</code>	209
7.42.1.5	<code>DEF_PARAME_MIN_SHORT_RAD_PREV</code>	209
7.42.1.6	<code>DEF_PARAME_OFFSET_MODE</code>	209
7.42.1.7	<code>DEF_PARAME_POST_MIN_LENGTH</code>	210
7.42.1.8	<code>DEF_PARAME_SW_LINE_ELLIPSE</code>	210
7.42.1.9	<code>DEF_PARAME_SW_OLD_MERGE_FUNC</code>	210
7.42.1.10	<code>DEF_PARAME_TH_MAX_ERROR</code>	210
7.42.1.11	<code>DEF_PARAME_TH_MAX_ERROR_MERGING</code>	210
7.42.1.12	<code>DEF_PARAME_TH_MEAN_ERROR</code>	210
7.42.1.13	<code>DEF_PARAME_TH_MEAN_ERROR_MERGING</code>	210
7.42.1.14	<code>DEF_SHORTEN_ELLIPSE_MERGING</code>	210
7.42.1.15	<code>MINIMUM_MIN_LENGTH</code>	210
7.42.1.16	<code>MINIMUM_MIN_SHORT_RAD</code>	211
7.42.1.17	<code>MINIMUM_TH_MAX_ERROR</code>	211
7.42.1.18	<code>MINIMUM_TH_MEAN_ERROR</code>	211
7.42.2	型定義	211
7.42.2.1	<code>ParamEllipseIW</code>	211
7.42.3	列挙型	211
7.42.3.1	<code>paramEllipseIW_Ellipse_with_line_key</code>	211
7.42.3.2	<code>paramEllipseIW_ErrCond_key</code>	211
7.42.3.3	<code>paramEllipseIW_OffsetMode_key</code>	211
7.42.3.4	<code>paramEllipseIW_Old_Merge_func_key</code>	212
7.43	<code>parameters.h</code>	213

7.43.1	説明	214
7.43.2	型定義	214
7.43.2.1	ParametersStereo	214
7.44	quaternion.c	215
7.44.1	関数	215
7.44.1.1	quat_conj	215
7.44.1.2	quat_copy	216
7.44.1.3	quat_fprintf	216
7.44.1.4	quat_irot	216
7.44.1.5	quat_make_from_rvec	216
7.44.1.6	quat_mult	216
7.44.1.7	quat_norm2	216
7.44.1.8	quat_normalize	216
7.44.1.9	quat_q_from_R	216
7.44.1.10	quat_R_from_q	216
7.44.1.11	quat_rot	216
7.45	quaternion.h	217
7.45.1	マクロ定義	218
7.45.1.1	QUAT_EPS	218
7.45.1.2	quat_fprint	218
7.45.1.3	quat_im	218
7.45.1.4	QUAT_INIT_ONE	218
7.45.1.5	QUAT_INIT_ZERO	218
7.45.1.6	quat_print	218
7.45.1.7	quat_printf	218
7.45.1.8	quat_re	219
7.45.2	型定義	219
7.45.2.1	quaternion_t	219
7.45.3	関数	219
7.45.3.1	quat_conj	219
7.45.3.2	quat_copy	219
7.45.3.3	quat_fprintf	219
7.45.3.4	quat_irot	219
7.45.3.5	quat_make_from_rvec	219

7.45.3.6	quat_mult	219
7.45.3.7	quat_norm2	219
7.45.3.8	quat_normalize	220
7.45.3.9	quat_q_from_R	220
7.45.3.10	quat_R_from_q	220
7.45.3.11	quat_rot	220
7.46	recogImage.cpp	221
7.46.1	説明	222
7.46.2	関数	222
7.46.2.1	constructImage	222
7.46.2.2	convertImage	222
7.46.2.3	destructImage	222
7.46.2.4	rgb2grayImage	222
7.46.2.5	undistortImage	222
7.46.2.6	writeRecogImage	222
7.47	recogImage.h	223
7.47.1	説明	224
7.47.2	型定義	224
7.47.2.1	RecogImage	224
7.47.3	関数	224
7.47.3.1	constructImage	224
7.47.3.2	convertImage	224
7.47.3.3	destructImage	225
7.47.3.4	rgb2grayImage	225
7.47.3.5	undistortImage	225
7.47.3.6	writeRecogImage	225
7.48	RecognitionKernel.cpp	226
7.48.1	関数	227
7.48.1.1	RecognitionKernel	227
7.49	RecognitionKernel.h	228
7.49.1	説明	228
7.49.2	関数	228
7.49.2.1	RecognitionKernel	228
7.50	recogParameter.cpp	229

7.50.1	列挙型	230
7.50.1.1	paramKey	230
7.50.2	関数	231
7.50.2.1	loadDebugParameter	231
7.50.2.2	loadRecogParameter	231
7.50.2.3	setDefaultRecogParameter	231
7.51	recogParameter.h	232
7.51.1	説明	232
7.51.2	関数	232
7.51.2.1	loadDebugParameter	232
7.51.2.2	loadRecogParameter	233
7.51.2.3	setDefaultRecogParameter	233
7.52	recogResult.h	234
7.52.1	説明	234
7.52.2	マクロ定義	234
7.52.2.1	RecogResultElementNum	234
7.52.3	列挙型	234
7.52.3.1	RecogResultElement	234
7.53	rtvcm.cpp	236
7.53.1	説明	237
7.53.2	関数	237
7.53.2.1	convertRTVCMtoFeatures3D	237
7.53.2.2	create_wireframe_model	237
7.53.2.3	freeRTVCM	237
7.53.2.4	readRTVCMModel	237
7.53.2.5	reverseCircle	237
7.53.2.6	reverseVertex	238
7.54	rtvcm.h	239
7.54.1	説明	240
7.54.2	型定義	240
7.54.2.1	RTVCM	240
7.54.2.2	RTVCM_Label	240
7.54.3	関数	240
7.54.3.1	convertRTVCMtoFeatures3D	240

7.54.3.2	freeRTVCM	240
7.54.3.3	readRTVCModel	240
7.54.3.4	reverseCircle	241
7.54.3.5	reverseVertex	241
7.55	score2d.cpp	242
7.55.1	説明	242
7.55.2	関数	243
7.55.2.1	compareResultScore	243
7.55.2.2	getResultScore	243
7.56	score2d.h	244
7.56.1	説明	244
7.56.2	関数	244
7.56.2.1	compareResultScore	244
7.56.2.2	getResultScore	244
7.57	searchEllipse_IW.cpp	245
7.57.1	マクロ定義	246
7.57.1.1	ADD_NEW_ELLIPSE_NG	246
7.57.1.2	ADD_NEW_ELLIPSE_OK	246
7.57.1.3	CHECK_NEXT_FAIL	247
7.57.1.4	CHECK_NEXT_SUCCESS	247
7.57.1.5	CHECK_SA_NG	247
7.57.1.6	CHECK_SA_OK	247
7.57.1.7	CHECK_TRACK_FAIL	247
7.57.1.8	CHECK_TRACK_SUCCESS	247
7.57.1.9	DG_SHIFT	247
7.57.1.10	DG_SKIP	247
7.57.1.11	DIR_GOAL_DEC	247
7.57.1.12	DIR_GOAL_INC	247
7.57.1.13	DIR_START_DEC	247
7.57.1.14	DIR_START_INC	248
7.57.1.15	DS_SHIFT	248
7.57.1.16	DS_SKIP	248
7.57.1.17	LOOP_CONTINUE	248
7.57.1.18	LOOP_EXIT_NORMAL	248

7.57.1.19	LOOP_EXIT_WHOLE	248
7.57.1.20	MAX_CURVE_LEN_UNDEFINED	248
7.57.1.21	NDIR4	248
7.57.1.22	POINT_TO_ELLIPSE_FAIL	248
7.57.1.23	POINT_TO_ELLIPSE_SUCCESS	248
7.57.1.24	SKIP_LOOP_CONTINUE	248
7.57.1.25	SKIP_LOOP_FULL	249
7.57.1.26	SKIP_LOOP_STOP	249
7.57.1.27	TRACKING_OFF	249
7.57.1.28	TRACKING_ON	249
7.57.1.29	TURN_BACKWARD	249
7.57.1.30	TURN_FORWARD	249
7.57.1.31	TURN_LEFT	249
7.57.1.32	TURN_RIGHT	249
7.57.2	関数	249
7.57.2.1	addArcSum	249
7.57.2.2	avec_to_ellipse	249
7.57.2.3	check_ellipse_cond	250
7.57.2.4	distanceAConic	250
7.57.2.5	mod_nPoint	250
7.57.2.6	P_to_avec_and_fix	250
7.57.2.7	searchEllipseIW	250
7.57.2.8	sum_to_P_dynamic	250
7.58	stereo.cpp	251
7.58.1	説明	252
7.58.2	関数	252
7.58.2.1	calculateLR2XYZ	252
7.58.2.2	calculatePlane3D	252
7.58.2.3	projectXYZ2LR	252
7.58.2.4	set_circle_to_OldFeature3D	252
7.58.2.5	set_vertex_to_OldFeature3D	253
7.59	stereo.h	254
7.59.1	説明	255
7.59.2	関数	255

7.59.2.1	calculateLR2XYZ	255
7.59.2.2	calculatePlane3D	256
7.59.2.3	projectXYZ2LR	256
7.59.2.4	set_circle_to_OldFeature3D	256
7.59.2.5	set_vertex_to_OldFeature3D	256
7.60	vectorutil.cpp	257
7.60.1	説明	258
7.60.2	関数	258
7.60.2.1	addV3	258
7.60.2.2	copyV2	258
7.60.2.3	copyV3	258
7.60.2.4	eigenM22	258
7.60.2.5	getAngle2D	258
7.60.2.6	getCrossProductV3	259
7.60.2.7	getDirectionVector	259
7.60.2.8	getDistanceV2	259
7.60.2.9	getDistanceV3	259
7.60.2.10	getInnerProductV3	259
7.60.2.11	getNormV2	259
7.60.2.12	getNormV3	259
7.60.2.13	getOrthogonalDir	259
7.60.2.14	getOrthogonalDir	259
7.60.2.15	inverseM33	259
7.60.2.16	isZero	260
7.60.2.17	mulM33	260
7.60.2.18	mulM33V3	260
7.60.2.19	mulV2S	260
7.60.2.20	mulV3S	260
7.60.2.21	normalizeV2	260
7.60.2.22	normalizeV3	260
7.60.2.23	quaternion_rotation	260
7.60.2.24	subM33	260
7.60.2.25	subV3	260
7.60.2.26	transposeM33	261

7.60.2.27	zeroV3	261
7.61	vectorutil.h	262
7.61.1	説明	263
7.61.2	型定義	263
7.61.2.1	M33	263
7.61.2.2	V2	263
7.61.2.3	V3	264
7.61.3	関数	264
7.61.3.1	addV3	264
7.61.3.2	copyV2	264
7.61.3.3	copyV3	264
7.61.3.4	eigenM22	264
7.61.3.5	getAngle2D	264
7.61.3.6	getCrossProductV3	264
7.61.3.7	getDirectionVector	264
7.61.3.8	getDistanceV2	264
7.61.3.9	getDistanceV3	265
7.61.3.10	getInnerProductV3	265
7.61.3.11	getNormV2	265
7.61.3.12	getNormV3	265
7.61.3.13	getOrthogonalDir	265
7.61.3.14	getOrthogonalDir	265
7.61.3.15	inverseM33	265
7.61.3.16	isZero	265
7.61.3.17	mulM33	265
7.61.3.18	mulM33V3	265
7.61.3.19	mulV2S	266
7.61.3.20	mulV3S	266
7.61.3.21	normalizeV2	266
7.61.3.22	normalizeV3	266
7.61.3.23	quaternion_rotation	266
7.61.3.24	subM33	266
7.61.3.25	subV3	266
7.61.3.26	transposeM33	266

7.62	vertex.cpp	267
7.62.1	説明	268
7.62.2	関数	268
7.62.2.1	isValidPixelPosition	268
7.62.2.2	reconstruct_hyperbola_to_vertex3D	268
7.63	vertex.h	269
7.63.1	説明	270
7.63.2	関数	270
7.63.2.1	reconstruct_hyperbola_to_vertex3D	270
7.64	visionErrorCode.h	271
7.64.1	説明	271
7.64.2	列挙型	271
7.64.2.1	VisionErrorCode	271

Chapter 1

ネームスペース索引

1.1 ネームスペース一覧

ネームスペースの一覧です。

[ovgr](#) 9

Chapter 2

データ構造索引

2.1 クラス階層

この継承一覧はおおまかにはソートされていますが、完全にアルファベット順でソートされてはいません。

Ellipse	15
_ellipse_arc_	18
_ellipse_arc_list_	20
_offset_prop_	22
_param_ellipse_IW_	23
_recogImage	25
SumSet	26
ovgr::Array< T, N >	29
CalibParam	31
CameraParam	33
Circle	35
CircleCandidate	38
ovgr::CorrespondenceThresholds	42
ovgr::CorrespondingPair	43
ovgr::CorrespondingSet	44
Data_2D	45
DistortionParam	46
EllipseGroup	50
ovgr::EqualOp< T >	51
tag_Wireframe::Face	52
ovgr::Feature2D	54
ovgr::ConicFeature2D	40
ovgr::EllipseFeature	48
ovgr::VertexFeature	121
ovgr::LineFeature2D	66
Feature2D_old	55
ovgr::Features2D	60
Features2D_old	61

Features3D	63
Match3Dresults	68
MatchResult	70
ModelFileInfo	72
ModelFileInfoNode	74
P2D	75
P3D	76
Parameters	77
ParametersFeature2D	80
ParametersMatch	85
ParametersStereo	86
ovgr::Point2D	88
ovgr::PointsOnEllipse	90
ovgr::PointsOnLine	91
RTVCM_Box	94
RTVCM_Circle	96
RTVCM_Cylinder	98
RTVCM_Vertex	100
RTVertexCircleModel	102
tag_Wireframe::Segment	106
StereoCalib	107
tag_Wireframe	109
Trace	111
Track	113
ovgr::VariableWatcher< T, Equal >	114
Vertex	115
VertexCandidate	118

Chapter 3

データ構造索引

3.1 データ構造

データ構造の説明です。

Ellipse	15
_ellipse_arc_	18
_ellipse_arc_list_	20
_offset_prop_	22
_param_ellipse_IW_	23
_recogImage	25
SumSet	26
ovgr::Array< T, N >	29
CalibParam (キャリブレーションパラメータ)	31
CameraParam (カメラパラメータ)	33
Circle (3次元円情報)	35
CircleCandidate (3次元円特徴候補データ)	38
ovgr::ConicFeature2D	40
ovgr::CorrespondenceThresholds	42
ovgr::CorrespondingPair	43
ovgr::CorrespondingSet	44
Data_2D	45
DistortionParam (歪みパラメータ)	46
ovgr::EllipseFeature	48
EllipseGroup (楕円重複除去用グループ情報)	50
ovgr::EqualOp< T >	51
tag_Wireframe::Face (< 面)	52
ovgr::Feature2D	54
Feature2D_old (各 2次元特徴)	55
ovgr::Features2D	60
Features2D_old (2次元特徴情報)	61
Features3D (3次元特徴情報)	63
ovgr::LineFeature2D	66
Match3Dresults (全認識結果)	68

MatchResult (各認識結果情報)	70
ModelFileInfo (モデルファイルリスト)	72
ModelFileInfoNode (モデルリストのノード)	74
P2D (2次元位置情報)	75
P3D (3次元位置情報)	76
Parameters (全パラメータ)	77
ParametersFeature2D (2次元特徴抽出用パラメータ)	80
ParametersMatch (認識用パラメータ)	85
ParametersStereo (ステレオ対応処理用パラメータ)	86
ovgr::Point2D	88
ovgr::PointsOnEllipse	90
ovgr::PointsOnLine	91
RTVCM_Box (モデル内の立方体データ)	94
RTVCM_Circle (モデル内の円データ)	96
RTVCM_Cylinder (モデル内の円筒データ)	98
RTVCM_Vertex (モデル内の頂点データ)	100
RTVertexCircleModel (モデルデータ構造体)	102
tag_Wireframe::Segment (< 線分)	106
StereoCalib (ステレオカメラキャリブレーションデータ)	107
tag_Wireframe (ワイヤフレームモデル)	109
Trace (認識結果評価用サンプリング点列情報)	111
Track (輪郭情報)	113
ovgr::VariableWatcher< T, Equal >	114
Vertex (3次元頂点情報)	115
VertexCandidate (三次元頂点特徴候補データ)	118
ovgr::VertexFeature	121

Chapter 4

ファイル索引

4.1 ファイル一覧

これはファイル一覧です。

calib.cpp (キャリブレーション関連の関数)	123
calib.h (キャリブレーション関連の関数)	125
calibUtil.cpp	127
calibUtil.h (キャリブレーションデータの変換関連)	128
circle.cpp (3次元円特徴生成関連関数)	130
circle.h (3次元円特徴生成関連関数)	132
common.h (各種の共通定義)	134
conic.cpp (二次曲線特徴抽出関連関数)	136
conic.h (二次曲線特徴抽出関連関数)	138
constants.hpp	141
correspondence.cpp	142
correspondence.hpp	143
debugutil.cpp (デバッグ用関数)	145
debugutil.h (デバッグ用関数)	148
drawing.cpp	151
drawing.hpp	152
ellipse_to_ylimit.cpp	153
ellipseIW.h	154
extractEdge.cpp (エッジ抽出関連関数)	159
extractEdge.h (エッジ抽出関連関数)	161
extractFeature.cpp (2次元特徴抽出関連関数)	164
extractFeature.hpp	166
extractFeature_old.cpp	168
extractFeature_old.h (2次元特徴抽出関連関数)	171
geometry.cpp	174
geometry.hpp	175
imageUtil.cpp (画像入出力関数)	177
imageUtil.h (画像入出力関連)	179
local.h	181

match3Dfeature.cpp (3次元特徴による認識関連関数)	182
match3Dfeature.h (3次元特徴による認識関連関数)	184
mathmisc.cpp	188
mathmisc.hpp	189
merge_ellipse.cpp	190
modelFileio.cpp	195
modelFileio.h (モデルをファイルから読み込む。)	196
modelListFileIO.cpp	198
modelListFileIO.h (モデルファイルの入出力関連)	199
modelpoints.cpp (モデル評価点生成関連関数)	201
modelpoints.h (モデル評価点生成関連関数)	203
pairedcircle.cpp (2円照合関連関数)	206
paramEllipseIW.h	208
parameters.h (処理パラメータ設定関連関数)	213
quaternion.c	215
quaternion.h	217
recogImage.cpp (画像入出力関数)	221
recogImage.h (画像入出力関数)	223
RecognitionKernel.cpp	226
RecognitionKernel.h (3次元物体認識の中核処理)	228
recogParameter.cpp	229
recogParameter.h (認識パラメータ設定関連)	232
recogResult.h (認識結果の定義)	234
rtvcm.cpp (モデル入出力関連関数)	236
rtvcm.h (モデル入出力関連関数)	239
score2d.cpp (2次元評価関連関数)	242
score2d.h (2次元評価関連関数)	244
searchEllipse_IW.cpp	245
stereo.cpp (ステレオ処理関連関数)	251
stereo.h (ステレオ処理関連関数)	254
vectorutil.cpp (ベクトル処理、行列処理 ユーティリティ関数)	257
vectorutil.h (ベクトル処理、行列処理 ユーティリティ関数)	262
vertex.cpp (3次元頂点特徴生成関連関数)	267
vertex.h (3次元頂点特徴生成関連関数)	269
visionErrorCode.h (返り値の定義)	271

Chapter 5

ネームスペース

5.1 ネームスペース ovgr

データ構造

- struct [CorrespondingPair](#)
- struct [CorrespondingSet](#)
- struct [CorrespondenceThresholds](#)
- struct [EqualOp](#)
- class [VariableWatcher](#)
- class [PointsOnLine](#)
- class [PointsOnEllipse](#)
- struct [Feature2D](#)
- struct [LineFeature2D](#)
- struct [ConicFeature2D](#)
- struct [VertexFeature](#)
- struct [EllipseFeature](#)
- struct [Features2D](#)
- struct [Array](#)
- struct [Point2D](#)

型定義

- typedef std::list< int > [feature_index_list_t](#)
- typedef std::vector< [feature_index_list_t](#) > [feature_map_t](#)
- typedef std::vector< int > [feature_indexes_t](#)
- typedef std::list< [feature_indexes_t](#) > [feature_list_t](#)
- typedef std::vector< [Point2D](#) > [Segment2D](#)

列挙型

- enum [CorrespondingCriteria](#) { [CorresOr](#), [CorresAnd](#) }

関数

- [CorrespondingPair](#) [make_corresponding_pairs](#) (const [Features2D](#) &feature1, const [CameraParam](#) ¶m1, const [Features2D](#) &feature2, const [CameraParam](#) ¶m2, const [CorrespondenceThresholds](#) &thres=[CorrespondenceThresholds](#)())
- [CorrespondingSet](#) [filter_corresponding_set](#) (const std::vector< const [CorrespondingPair](#) * > &corres_pair, const [CorrespondingCriteria](#) criteria=[CorresOr](#))
- [Features2D](#) [extractFeatures](#) (const unsigned char *edge, const [Parameters](#) ¶meters, const [Features3D](#) &model)
- [Features2D](#) [ImageToFeature2D](#) (unsigned char *src, unsigned char *edge, const [Parameters](#) ¶meters, const [Features3D](#) &model)
- [Features2D_old](#) * [create_old_features_from_new_one](#) (const [Features2D](#) &features)
- [Features2D](#) [create_new_features_from_old_one](#) (const [Features2D_old](#) *old_features, unsigned char *img=NULL, const [Parameters](#) *parameters=NULL)
- cv::RotatedRect [calc_ellipse_rr](#) (const double coeff[6])
- template<class VF >
void [draw_VertexFeatures](#) (cv::Mat &img, const VF &vf)
- template<class EF >
void [draw_EllipseFeatures](#) (cv::Mat &img, const EF &ef)
- void [calc_crossing_point](#) (double p[3], const double l1[3], const double l2[3])
- cv::Mat [calc_homography](#) (const std::vector< [Array](#)< double, 3 > > &point1, const std::vector< [Array](#)< double, 3 > > &point2, const std::vector< [Array](#)< double, 3 > > &line1=std::vector< [Array](#)< double, 3 > >(), const std::vector< [Array](#)< double, 3 > > &line2=std::vector< [Array](#)< double, 3 > >())
- cv::Mat [calc_essential_matrix](#) (const [CameraParam](#) &cp1, const [CameraParam](#) &cp2)
- cv::Mat [calc_fundamental_matrix](#) (const [CameraParam](#) &cp1, const [CameraParam](#) &cp2)
- void [calc_epipolar_line](#) (double line_coef[3], const cv::Mat &E, const [Point2D](#) &point, const bool inv=false)
- void [calc_epipole](#) (double e[3], const [CameraParam](#) &cp1, const [CameraParam](#) &cp2)
- cv::Mat [calc_infinite_homography](#) (const [CameraParam](#) &cp1, const [CameraParam](#) &cp2)
- double [distance_from_line](#) (const double line_coef[3], const [Point2D](#) &point)
- void [calc_line_joining_points](#) (double line_coef[3], const double p1[3], const double p2[3])
- int [solve_quad_eq](#) (double x[2], const double a, const double b, const double c)

2 次方程式 $ax^2 + bx + c = 0$ を解く

5.1.1 型定義

5.1.1.1 `typedef std::list<int> ovgr::feature_index_list_t`

5.1.1.2 `typedef std::vector<int> ovgr::feature_indexes_t`

5.1.1.3 `typedef std::list<feature_indexes_t> ovgr::feature_list_t`

5.1.1.4 `typedef std::vector<feature_index_list_t> ovgr::feature_map_t`

5.1.1.5 `typedef std::vector<Point2D> ovgr::Segment2D`

5.1.2 列挙型

5.1.2.1 `enum ovgr::CorrespondingCriteria`

列挙型の値:

CorresOr

CorresAnd

5.1.3 関数

5.1.3.1 `void ovgr::calc_crossing_point (double p[3], const double l1[3], const double l2[3])`

- 5.1.3.2 `cv::RotatedRect ovgr::calc_ellipse_rr (const double coeff[6])`
- 5.1.3.3 `void ovgr::calc_epipolar_line (double line_coef[3], const cv::Mat & E, const Point2D & point, const bool inv = false)`
- 5.1.3.4 `void ovgr::calc_epipole (double e[3], const CameraParam & cp1, const CameraParam & cp2)`
- 5.1.3.5 `cv::Mat ovgr::calc_essential_matrix (const CameraParam & cp1, const CameraParam & cp2)`
- 5.1.3.6 `cv::Mat ovgr::calc_fundamental_matrix (const CameraParam & cp1, const CameraParam & cp2)`
- 5.1.3.7 `cv::Mat ovgr::calc_homography (const std::vector< Array< double, 3 > > & point1, const std::vector< Array< double, 3 > > & point2, const std::vector< Array< double, 3 > > & line1 = std::vector<Array<double, 3> >(), const std::vector< Array< double, 3 > > & line2 = std::vector<Array<double, 3> >())`
- 5.1.3.8 `cv::Mat ovgr::calc_infinite_homography (const CameraParam & cp1, const CameraParam & cp2)`
- 5.1.3.9 `void ovgr::calc_line_joining_points (double line_coef[3], const double p1[3], const double p2[3])`

- 5.1.3.10 **Features2D** ovgr::create_new_features_from_old_one (const Features2D_old * *old_features*, unsigned char * *img* = NULL, const Parameters * *parameters* = NULL)
- 5.1.3.11 **Features2D_old** * ovgr::create_old_features_from_new_one (const Features2D & *features*)
- 5.1.3.12 **double** ovgr::distance_from_line (const double *line_coef*[3], const Point2D & *point*)
- 5.1.3.13 **template<class EF > void** ovgr::draw_EllipseFeatures (cv::Mat & *img*, const EF & *ef*) [**inline**]
- 5.1.3.14 **template<class VF > void** ovgr::draw_VertexFeatures (cv::Mat & *img*, const VF & *vf*) [**inline**]
- 5.1.3.15 **Features2D** ovgr::extractFeatures (const unsigned char * *edge*, const Parameters & *parameters*, const Features3D & *model*)
- 5.1.3.16 **CorrespondingSet** ovgr::filter_corresponding_set (const std::vector< const CorrespondingPair * > & *corres_pair*, const CorrespondingCriteria *criteria* = **CorresOr**)
- 5.1.3.17 **Features2D** ovgr::ImageToFeature2D (unsigned char * *src*, unsigned char * *edge*, const Parameters & *parameters*, const Features3D & *model*)

5.1.3.18 `CorrespondingPair ovgr::make_corresponding_pairs (const Features2D & feature1, const CameraParam & param1, const Features2D & feature2, const CameraParam & param2, const CorrespondenceThresholds & thres = CorrespondenceThresholds ())`

5.1.3.19 `int ovgr::solve_quad_eq (double x[2], const double a, const double b, const double c)`

2 次方程式 $ax^2 + bx + c = 0$ を解く

Chapter 6

データ構造

6.1 構造体 `_Ellipse_`

```
#include <ellipseIW.h>
```

変数

- double `center` [NDIM2]
- double `rad` [NDIM2]
- double `axis` [NAXIS][NDIM2]
- double `coef` [NDIM_CONIC_FULL]
- double `P00` [NDIM_CONIC_HALF][NDIM_CONIC_HALF]
- double `P01` [NDIM_CONIC_HALF][NDIM_CONIC_HALF]
- double `P10` [NDIM_CONIC_HALF][NDIM_CONIC_HALF]
- double `P11` [NDIM_CONIC_HALF][NDIM_CONIC_HALF]
- double `a` [NDIM_CONIC_HALF][NDIM_CONIC_FULL]
- double `M` [NDIM_CONIC_HALF][NDIM_CONIC_HALF]
- double `cubic` [NCOEFCUBIC]
- int `neval`
- double `eval` [NDIM_CONIC_HALF]
- double `offset` [NDIM2]
- double `meanError`
- double `maxError`

6.1.1 構造体

6.1.1.1 `double _Ellipse_::a[NDIM_CONIC_HALF][NDIM_CONIC_FULL]`

6.1.1.2 `double _Ellipse_::axis[NAXIS][NDIM2]`

6.1.1.3 `double _Ellipse_::center[NDIM2]`

6.1.1.4 `double _Ellipse_::coef[NDIM_CONIC_FULL]`

6.1.1.5 `double _Ellipse_::cubic[NCOEFCUBIC]`

6.1.1.6 `double _Ellipse_::eval[NDIM_CONIC_HALF]`

6.1.1.7 `double _Ellipse_::M[NDIM_CONIC_HALF][NDIM_CONIC_HALF]`

6.1.1.8 `double _Ellipse_::maxError`

6.1.1.9 `double _Ellipse_::meanError`

6.1.1.10 `int _Ellipse_::neval`

6.1.1.11 `double _Ellipse_::offset[NDIM2]`

6.1.1.12 `double _Ellipse_::P00[NDIM_CONIC_HALF][NDIM_CONIC_HALF]`

6.1.1.13 `double _Ellipse_::P01[NDIM_CONIC_HALF][NDIM_CONIC_HALF]`

6.1.1.14 `double _Ellipse_::P10[NDIM_CONIC_HALF][NDIM_CONIC_HALF]`

6.1.1.15 `double _Ellipse_::P11[NDIM_CONIC_HALF][NDIM_CONIC_HALF]`

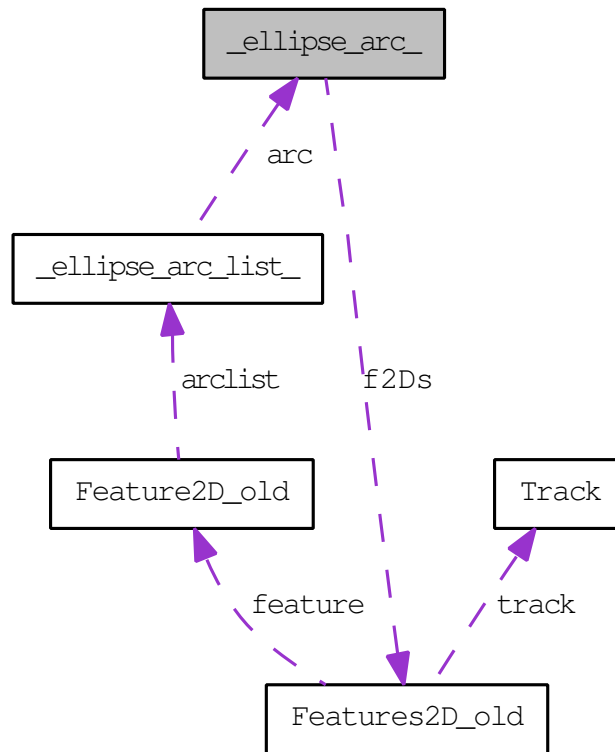
6.1.1.16 `double _Ellipse_::rad[NDIM2]`

この構造体の説明は次のファイルから生成されました:

- [ellipseIW.h](#)

6.2 構造体 `_ellipse_arc_`

`#include <extractFeature_old.h>` `_ellipse_arc_` のコラボレーション図



変数

- struct `Features2D_old` * `f2Ds`
- int `ntrack`
- int `start`
- int `goal`

6.2.1 構造体

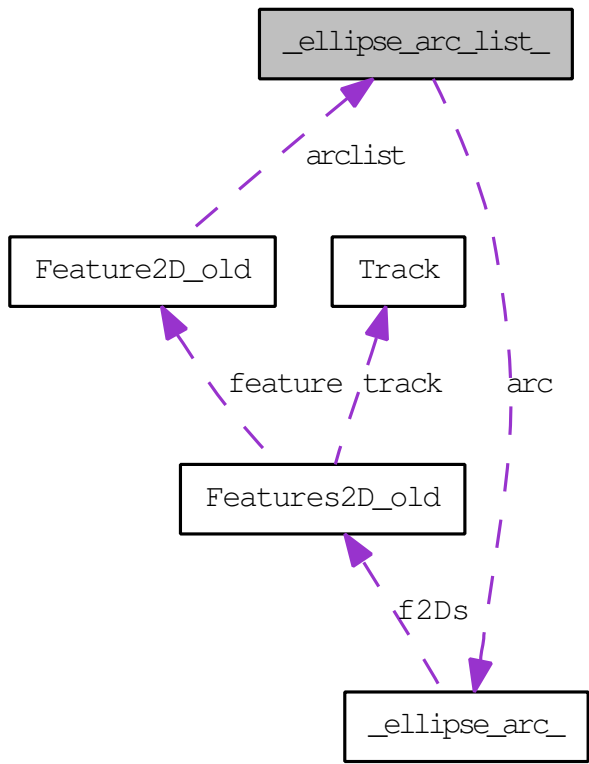
6.2.1.1 `struct Features2D_old* _ellipse_arc_::f2Ds` [read]

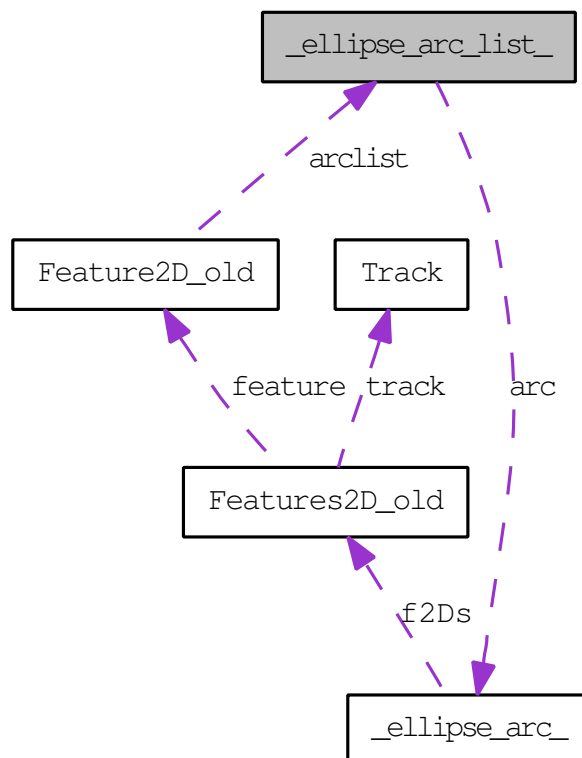
6.2.1.2 `int _ellipse_arc_::goal`**6.2.1.3** `int _ellipse_arc_::ntrack`**6.2.1.4** `int _ellipse_arc_::start`

この構造体の説明は次のファイルから生成されました:

- [extractFeature_old.h](#)

6.3 構造体 `_ellipse_arc_list_`

`#include <extractFeature_old.h>_ellipse_arc_list_のコラボレーション`




変数

- `int n`
- `EllipseArc * arc`

6.3.1 構造体

6.3.1.1 `EllipseArc* _ellipse_arc_list_::arc`

6.3.1.2 `int _ellipse_arc_list_::n`

この構造体の説明は次のファイルから生成されました:

- [extractFeature_old.h](#)

6.4 構造体 `_offset_prop_`

```
#include <ellipseIW.h>
```

変数

- `int mode`
- `double d [2]`

6.4.1 構造体

6.4.1.1 `double _offset_prop_::d[2]`

6.4.1.2 `int _offset_prop_::mode`

この構造体の説明は次のファイルから生成されました:

- `ellipseIW.h`

6.5 構造体 `_param_ellipse_IW_`

```
#include <paramEllipseIW.h>
```

変数

- int `Condition`
- int `MinLength`
- int `PostMinLength`
- double `MinShortRadPrev`
- double `MinShortRadPost`
- double `ThMeanError`
- double `ThMaxError`
- double `ThMeanErrorMerging`
- double `ThMaxErrorMerging`
- double `MinSD`
- int `OffsetMode`
- int `SwLineEllipse`
- int `SwOldMergeFunc`
- int `ShortenEllipseMerging`

6.5.1 構造体

6.5.1.1 int `_param_ellipse_IW_::Condition`

6.5.1.2 int `_param_ellipse_IW_::MinLength`

6.5.1.3 double `_param_ellipse_IW_::MinSD`

6.5.1.4 double `_param_ellipse_IW_::MinShortRadPost`

6.5.1.5 double _param_ellipse_IW_::MinShortRadPrev

6.5.1.6 int _param_ellipse_IW_::OffsetMode

6.5.1.7 int _param_ellipse_IW_::PostMinLength

6.5.1.8 int _param_ellipse_IW_::ShortenEllipseMerging

6.5.1.9 int _param_ellipse_IW_::SwLineEllipse

6.5.1.10 int _param_ellipse_IW_::SwOldMergeFunc

6.5.1.11 double _param_ellipse_IW_::ThMaxError

6.5.1.12 double _param_ellipse_IW_::ThMaxErrorMerging

6.5.1.13 double _param_ellipse_IW_::ThMeanError

6.5.1.14 double _param_ellipse_IW_::ThMeanErrorMerging

この構造体の説明は次のファイルから生成されました:

- [paramEllipseIW.h](#)

6.6 構造体 _recogImage

```
#include <recogImage.h>
```

変数

- int [colsize](#)
- int [rowsize](#)
- int [bytePerPixel](#)
- unsigned char * [pixel](#)

6.6.1 構造体

6.6.1.1 int _recogImage::bytePerPixel

6.6.1.2 int _recogImage::colsize

6.6.1.3 unsigned char* _recogImage::pixel

6.6.1.4 int _recogImage::rowsize

この構造体の説明は次のファイルから生成されました:

- [recogImage.h](#)

6.7 構造体 `_SumSet_`

```
#include <ellipseIW.h>
```

変数

- double `x4`
- double `x3y`
- double `x2y2`
- double `xy3`
- double `y4`
- double `x3`
- double `x2y`
- double `xy2`
- double `y3`
- double `x2`
- double `xy`
- double `y2`
- double `x`
- double `y`
- double `n`

6.7.1 構造体

6.7.1.1 `double _SumSet::n`

6.7.1.2 `double _SumSet::x`

6.7.1.3 `double _SumSet::x2`

6.7.1.4 `double _SumSet::x2y`

6.7.1.5 double _SumSet_::x2y2

6.7.1.6 double _SumSet_::x3

6.7.1.7 double _SumSet_::x3y

6.7.1.8 double _SumSet_::x4

6.7.1.9 double _SumSet_::xy

6.7.1.10 double _SumSet_::xy2

6.7.1.11 double _SumSet_::xy3

6.7.1.12 double _SumSet_::y

6.7.1.13 double _SumSet_::y2

6.7.1.14 double _SumSet_::y3

6.7.1.15 double _SumSet_::y4

この構造体の説明は次のファイルから生成されました:

- [ellipseIW.h](#)

6.8 構造体 テンプレート ovgr::Array< T, N >

```
#include <geometry.hpp>
```

Public 型

- typedef T [data_type](#)

Public メソッド

- T [operator\[\]](#) (size_t i) const
- T & [operator\[\]](#) (size_t i)

変数

- T [elem](#) [N]

```
template<typename T, int N> struct ovgr::Array< T, N >
```

6.8.1 型定義

```
6.8.1.1 template<typename T , int N> typedef T ovgr::Array< T, N  
::data_type
```

6.8.2 関数

```
6.8.2.1 template<typename T , int N> T& ovgr::Array< T, N >::operator[]  
(size_t i) [inline]
```

```
6.8.2.2 template<typename T , int N> T ovgr::Array< T, N >::operator[]  
(size_t i) const [inline]
```

6.8.3 構造体

6.8.3.1 `template<typename T ,int N> T ovgr::Array< T, N >::elem[N]`

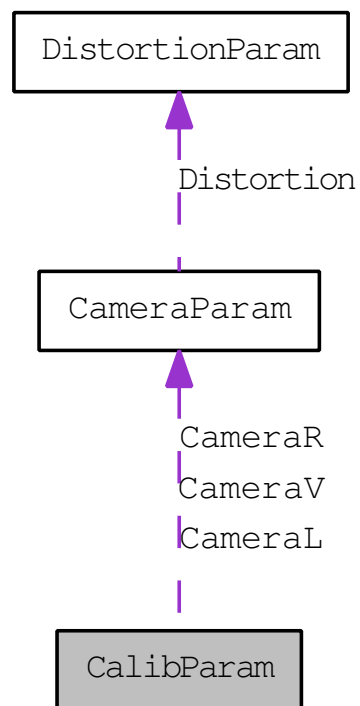
この構造体の説明は次のファイルから生成されました:

- [geometry.hpp](#)

6.9 構造体 CalibParam

キャリブレーションパラメータ

#include <calib.h> CalibParam のコラボレーション図



変数

- int numOfCameras
カメラ数
- int colsize
画像幅
- int rowsize
画像高さ
- CameraParam CameraL
左カメラの実画像とワールド座標の関係の全パラメータ

- [CameraParam CameraR](#)
右カメラの実画像とワールド座標の関係の全パラメータ
- [CameraParam CameraV](#)
右カメラの実画像とワールド座標の関係の全パラメータ

6.9.1 説明

キャリブレーションパラメータ

6.9.2 構造体

6.9.2.1 CameraParam CalibParam::CameraL

左カメラの実画像とワールド座標の関係の全パラメータ

6.9.2.2 CameraParam CalibParam::CameraR

右カメラの実画像とワールド座標の関係の全パラメータ

6.9.2.3 CameraParam CalibParam::CameraV

右カメラの実画像とワールド座標の関係の全パラメータ

6.9.2.4 int CalibParam::colsize

画像幅

6.9.2.5 int CalibParam::numOfCameras

カメラ数

6.9.2.6 int CalibParam::rowsize

画像高さ

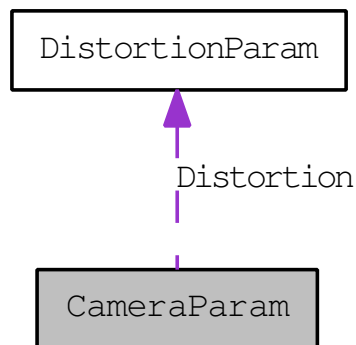
この構造体の説明は次のファイルから生成されました:

- [calib.h](#)

6.10 構造体 CameraParam

カメラパラメータ

#include <calib.h>CameraParam のコラボレーション図



変数

- double [Rotation](#) [3][3]
回転行列
- double [Translation](#) [3]
移動ベクトル
- double [Position](#) [3]
カメラ位置 ($-rR \cdot T$)
- [DistortionParam Distortion](#)
歪みパラメータ
- double [intrinsicMatrix](#) [3][3]
内部パラメータ行列

6.10.1 説明

カメラパラメータ

6.10.2 構造体

6.10.2.1 DistortionParam CameraParam::Distortion

歪みパラメータ

6.10.2.2 double CameraParam::intrinsicMatrix[3][3]

内部パラメータ行列

6.10.2.3 double CameraParam::Position[3]

カメラ位置 ($-rR \cdot T$)

6.10.2.4 double CameraParam::Rotation[3][3]

回転行列

6.10.2.5 double CameraParam::Translation[3]

移動ベクトル

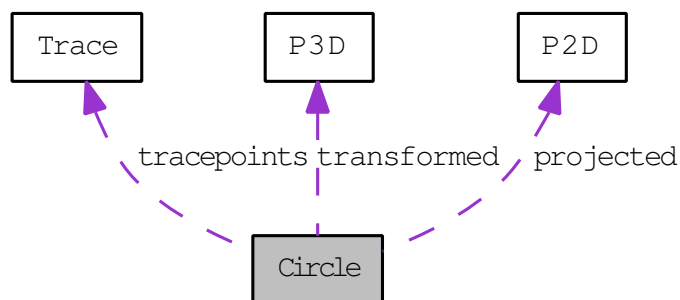
この構造体の説明は次のファイルから生成されました:

- [calib.h](#)

6.11 構造体 Circle

3次元円情報

#include <match3Dfeature.h> Circle のコラボレーション図



変数

- int **label**
ラベル
- int **n**
通し番号
- int **side**
表裏情報
- double **radius**
半径
- double **normal** [3]
法線
- double **tPose** [4][4]
認識用姿勢行列
- int **numOfTracePoints**
認識評価用のサンプリング点列数
- **Trace** * **tracepoints**
認識評価用のサンプリング点列情報

- **P3D * transformed**
認識時の位置・姿勢変換後の 3 次元点列
- **P2D * projected**
2 次元評価時の画像投影 2 次元点列

6.11.1 説明

3 次元円情報

6.11.2 構造体

6.11.2.1 int Circle::label

ラベル

6.11.2.2 int Circle::n

通し番号

6.11.2.3 double Circle::normal[3]

法線

6.11.2.4 int Circle::numOfTracePoints

認識評価用のサンプリング点列数

6.11.2.5 P2D* Circle::projected

2 次元評価時の画像投影 2 次元点列

6.11.2.6 double Circle::radius

半径

6.11.2.7 int Circle::side

表裏情報

6.11.2.8 double Circle::tPose[4][4]

認識用姿勢行列

6.11.2.9 Trace* Circle::tracepoints

認識評価用のサンプリング点列情報

6.11.2.10 P3D* Circle::transformed

認識時の位置・姿勢変換後の3次元点列

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.12 構造体 CircleCandidate

三次元円特徴候補データ

```
#include <stereo.h>
```

変数

- int **valid**
有効・無効フラグ
- double **center** [3]
中心座標
- double **normal** [3]
法線ベクトル (単位化済)
- double **radius**
半径

6.12.1 説明

三次元円特徴候補データ

6.12.2 構造体

6.12.2.1 double CircleCandidate::center[3]

中心座標

6.12.2.2 double CircleCandidate::normal[3]

法線ベクトル (単位化済)

6.12.2.3 double CircleCandidate::radius

半径

6.12.2.4 int CircleCandidate::valid

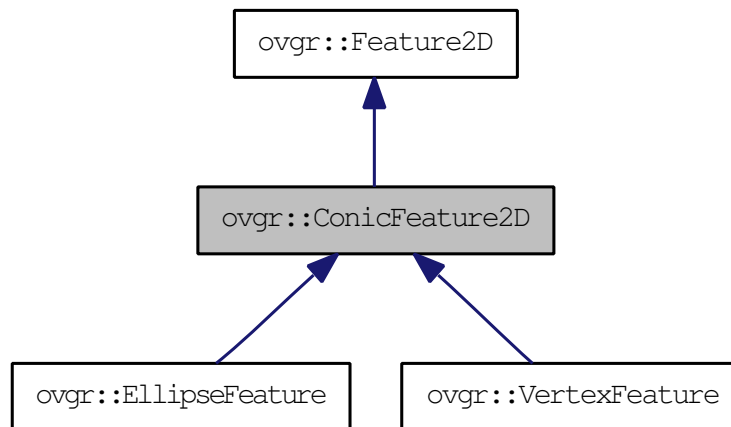
有効・無効フラグ

この構造体の説明は次のファイルから生成されました:

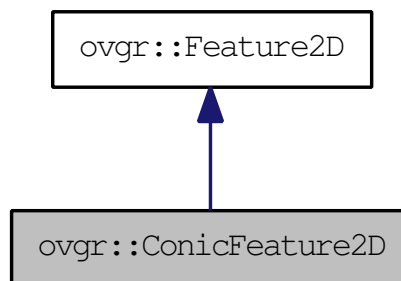
- [stereo.h](#)

6.13 構造体 `ovgr::ConicFeature2D`

`#include <extractFeature.hpp>` `ovgr::ConicFeature2D` に対する継承グラフ



`ovgr::ConicFeature2D` のコラボレーション図



変数

- `double coef[6]`

$$coef[0]*x^2 + 2*coef[1]*x*y + coef[2]*y^2 + 2*coef[3]*x + 2*coef[4]*y + coef[5] = 0$$

6.13.1 構造体

6.13.1.1 double ovgr::ConicFeature2D::coef[6]
$$\text{coef}[0]*x^2 + 2*\text{coef}[1]*x*y + \text{coef}[2]*y^2 + 2*\text{coef}[3]*x + 2*\text{coef}[4]*y + \text{coef}[5] = 0$$

この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

6.14 構造体 `ovgr::CorrespondenceThresholds`

```
#include <correspondence.hpp>
```

Public メソッド

- [CorrespondenceThresholds](#) (const double vt=1.0, const double et=5.0)

変数

- double [vertex_tolerance](#)
- double [ellipse_tolerance](#)

6.14.1 コンストラクタとデストラクタ

6.14.1.1 `ovgr::CorrespondenceThresholds::CorrespondenceThresholds (const double vt = 1.0, const double et = 5.0) [inline]`

6.14.2 構造体

6.14.2.1 `double ovgr::CorrespondenceThresholds::ellipse_tolerance`

6.14.2.2 `double ovgr::CorrespondenceThresholds::vertex_tolerance`

この構造体の説明は次のファイルから生成されました:

- [correspondence.hpp](#)

6.15 構造体 `ovgr::CorrespondingPair`

```
#include <correspondence.hpp>
```

変数

- `feature_map_t vertex`
- `feature_map_t ellipse`

6.15.1 構造体

6.15.1.1 `feature_map_t ovgr::CorrespondingPair::ellipse`

6.15.1.2 `feature_map_t ovgr::CorrespondingPair::vertex`

この構造体の説明は次のファイルから生成されました:

- `correspondence.hpp`

6.16 構造体 `ovgr::CorrespondingSet`

```
#include <correspondence.hpp>
```

変数

- [feature_list_t vertex](#)
- [feature_list_t ellipse](#)

6.16.1 構造体

6.16.1.1 `feature_list_t ovgr::CorrespondingSet::ellipse`

6.16.1.2 `feature_list_t ovgr::CorrespondingSet::vertex`

この構造体の説明は次のファイルから生成されました:

- [correspondence.hpp](#)

6.17 構造体 Data_2D

```
#include <common.h>
```

変数

- double [col](#)
- double [row](#)

6.17.1 構造体

6.17.1.1 double Data_2D::col

6.17.1.2 double Data_2D::row

この構造体の説明は次のファイルから生成されました:

- [common.h](#)

6.18 構造体 DistortionParam

歪みパラメータ

```
#include <calib.h>
```

変数

- double [k1](#)
- double [k2](#)
半径方向の歪み係数
- double [p1](#)
- double [p2](#)
円周方向の歪み係数
- double [k3](#)
半径方向の歪み係数

6.18.1 説明

歪みパラメータ

6.18.2 構造体

6.18.2.1 double DistortionParam::k1

6.18.2.2 double DistortionParam::k2

半径方向の歪み係数

6.18.2.3 double DistortionParam::k3

半径方向の歪み係数

6.18.2.4 `double DistortionParam::p1`**6.18.2.5** `double DistortionParam::p2`

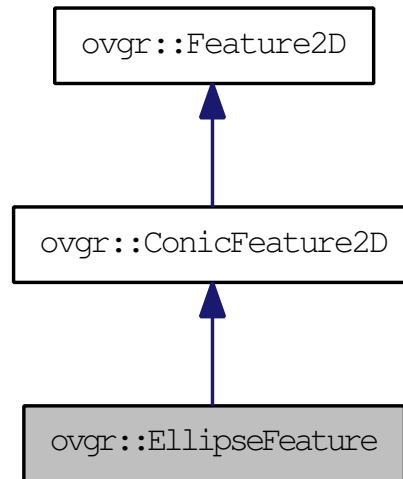
円周方向の歪み係数

この構造体の説明は次のファイルから生成されました:

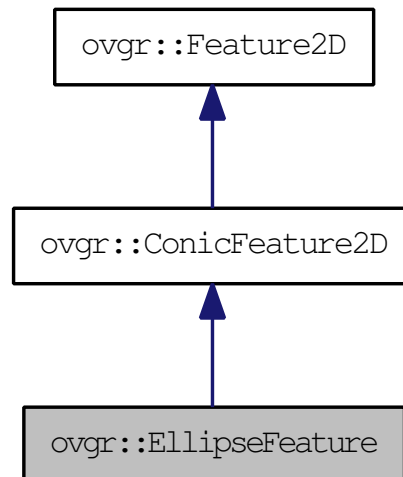
- [calib.h](#)

6.19 構造体 `ovgr::EllipseFeature`

`#include <extractFeature.hpp>` `ovgr::EllipseFeature` に対する継承グラフ



`ovgr::EllipseFeature` のコラボレーション図



変数

- double `center` [2]

中心

- double [axis](#) [2]
長軸、短軸の長さ
- double [theta](#)
 x 軸に対する回転角

6.19.1 構造体

6.19.1.1 double `ovgr::EllipseFeature::axis[2]`

長軸、短軸の長さ

6.19.1.2 double `ovgr::EllipseFeature::center[2]`

中心

6.19.1.3 double `ovgr::EllipseFeature::theta`

x 軸に対する回転角

この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

6.20 構造体 EllipseGroup

楕円重複除去用グループ情報

```
#include <extractFeature_old.h>
```

変数

- `int * groupNums`
グループ要素番号
- `double groupCenter [2]`
(work) グループの中心座標
- `int nCurrNum`
(work) グループ要素数

6.20.1 説明

楕円重複除去用グループ情報

6.20.2 構造体

6.20.2.1 `double EllipseGroup::groupCenter[2]`

(work) グループの中心座標

6.20.2.2 `int* EllipseGroup::groupNums`

グループ要素番号

6.20.2.3 `int EllipseGroup::nCurrNum`

(work) グループ要素数

この構造体の説明は次のファイルから生成されました:

- `extractFeature_old.h`

6.21 構造体 テンプレート ovgr::EqualOp< T >

```
#include <debugutil.h>
```

Public メソッド

- bool [operator\(\)](#) (const T &a, const T &b)

```
template<class T> struct ovgr::EqualOp< T >
```

6.21.1 関数

```
6.21.1.1 template<class T > bool ovgr::EqualOp< T >::operator() (const T &  
a, const T & b) [inline]
```

この構造体の説明は次のファイルから生成されました:

- [debugutil.h](#)

6.22 構造体 `tag_Wireframe::Face`

< 面

```
#include <match3Dfeature.h>
```

Public メソッド

- `Face()`

変数

- `std::vector< int > segment_id`
面を構成する線分
- `double normal [3]`
面の法線ベクトル

6.22.1 説明

< 面

6.22.2 コンストラクタとデストラクタ

6.22.2.1 `tag_Wireframe::Face::Face()` [`inline`]

6.22.3 構造体

6.22.3.1 `double tag_Wireframe::Face::normal[3]`

面の法線ベクトル

6.22.3.2 `std::vector<int> tag_Wireframe::Face::segment_id`

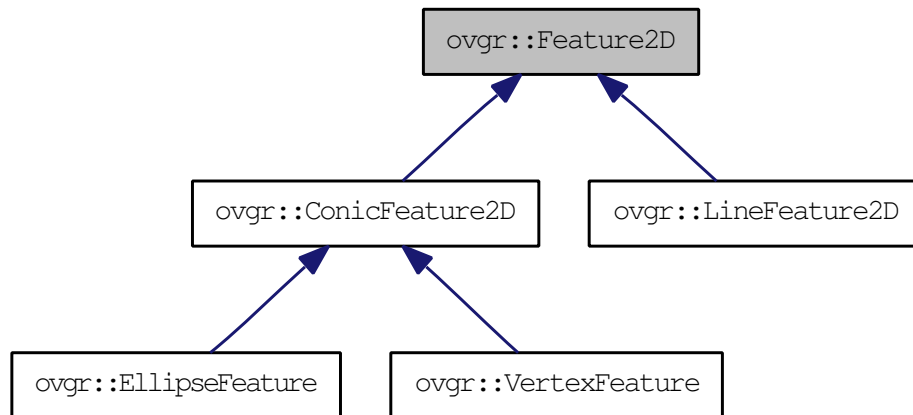
面を構成する線分

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.23 構造体 `ovgr::Feature2D`

`#include <extractFeature.hpp>` `ovgr::Feature2D` に対する継承グラフ



変数

- `std::vector< Segment2D > segment`
元になった点列
- `double error`
当てはめ平均誤差 [*pixel*]

6.23.1 構造体

6.23.1.1 `double ovgr::Feature2D::error`

当てはめ平均誤差 [*pixel*]

6.23.1.2 `std::vector<Segment2D> ovgr::Feature2D::segment`

元になった点列

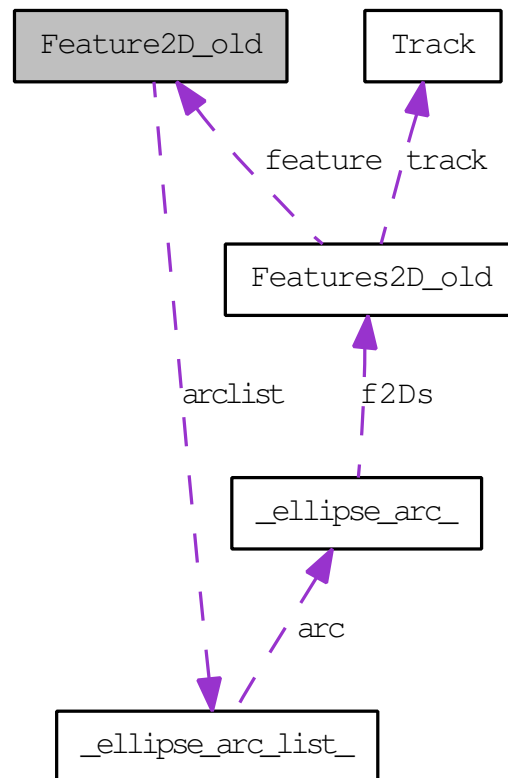
この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

6.24 構造体 Feature2D_old

各 2 次元特徴

#include <extractFeature_old.h> Feature2D_old のコラボレーション図



変数

- [ConicType type](#)
二次曲線の分類
- double [coef](#) [6]
- double [center](#) [2]
楕円中心または、双曲線の漸近線交点
- double [startPoint](#) [2]
曲線上の始点

- double `endPoint` [2]
曲線上の終点
- int `start`
始点番号
- int `end`
終点番号
- int `all`
輪郭全体の点数 (*nPoint*)
- double `startSPoint` [2]
点列の始点の位置
- double `middleSPoint` [2]
点列の中間の位置
- double `endSPoint` [2]
点列の終点の位置
- double `ev` [2][2]
楕円の回転行列
- double `axis` [2]
楕円の長半径、短半径
- double `direction` [2]
直線の方角ベクトル
- int `nPoints`
特徴抽出に使われた点数
- int `nTrack`
輪郭番号
- double `error`
当てはめ誤差
- double `lineLength`
直線の長さ
- double `lineLength1`
双曲線の線分 *l* の長さ
- double `lineLength2`

双曲線の線分 2 の長さ

- double [lineAngle](#)
双曲線の 2 線分のなす角度
- [EllipseArcList arclist](#)

6.24.1 説明

各 2 次元特徴

6.24.2 構造体

6.24.2.1 int Feature2D_old::all

輪郭全体の点数 (nPoint)

6.24.2.2 EllipseArcList Feature2D_old::arclist

6.24.2.3 double Feature2D_old::axis[2]

楕円の長半径、短半径

6.24.2.4 double Feature2D_old::center[2]

楕円中心または、双曲線の漸近線交点

6.24.2.5 double Feature2D_old::coef[6]

二次曲線の係数。 $ax^2 + bxy + cy^2 + dx + ey + f = 0$, $a = \text{coef}[0]$, ... , $f = \text{coef}[5]$ に対応

6.24.2.6 double Feature2D_old::direction[2]

直線の方法ベクトル

6.24.2.7 int Feature2D_old::end

終点番号

6.24.2.8 double Feature2D_old::endPoint[2]

曲線上の終点

6.24.2.9 double Feature2D_old::endSPoint[2]

点列の終点の位置

6.24.2.10 double Feature2D_old::error

当てはめ誤差

6.24.2.11 double Feature2D_old::ev[2][2]

楕円の回転行列

6.24.2.12 double Feature2D_old::lineAngle

双曲線の 2 線分のなす角度

6.24.2.13 double Feature2D_old::lineLength

直線の長さ

6.24.2.14 double Feature2D_old::lineLength1

双曲線の線分 1 の長さ

6.24.2.15 double Feature2D_old::lineLength2

双曲線の線分 2 の長さ

6.24.2.16 double Feature2D_old::middleSPoint[2]

点列の中間の位置

6.24.2.17 int Feature2D_old::nPoints

特徴抽出に使われた点数

6.24.2.18 int Feature2D_old::nTrack

輪郭番号

6.24.2.19 int Feature2D_old::start

始点番号

6.24.2.20 double Feature2D_old::startPoint[2]

曲線上の始点

6.24.2.21 double Feature2D_old::startSPoint[2]

点列の始点の位置

6.24.2.22 ConicType Feature2D_old::type

二次曲線の分類

この構造体の説明は次のファイルから生成されました:

- [extractFeature_old.h](#)

6.25 構造体 `ovgr::Features2D`

```
#include <extractFeature.hpp>
```

変数

- `std::vector< VertexFeature > vertex`
- `std::vector< EllipseFeature > ellipse`

6.25.1 構造体

6.25.1.1 `std::vector<EllipseFeature> ovgr::Features2D::ellipse`

6.25.1.2 `std::vector<VertexFeature> ovgr::Features2D::vertex`

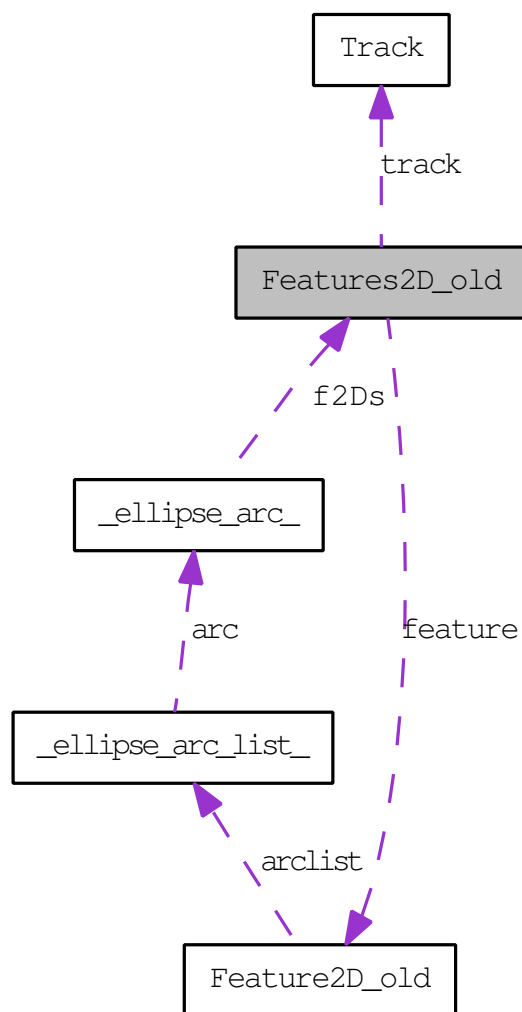
この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

6.26 構造体 Features2D_old

2次元特徴情報

#include <extractFeature_old.h>Features2D_old のコラボレーション図



変数

- int `nAlloc`
メモリ確保量

- `int nFeature`
2次元特徴数
- `Feature2D_old * feature`
2次元特徴情報
- `int nTrack`
輪郭数
- `Track * track`
輪郭情報

6.26.1 説明

2次元特徴情報

6.26.2 構造体

6.26.2.1 `Feature2D_old* Features2D_old::feature`

2次元特徴情報

6.26.2.2 `int Features2D_old::nAlloc`

メモリ確保量

6.26.2.3 `int Features2D_old::nFeature`

2次元特徴数

6.26.2.4 `int Features2D_old::nTrack`

輪郭数

6.26.2.5 `Track* Features2D_old::track`

輪郭情報

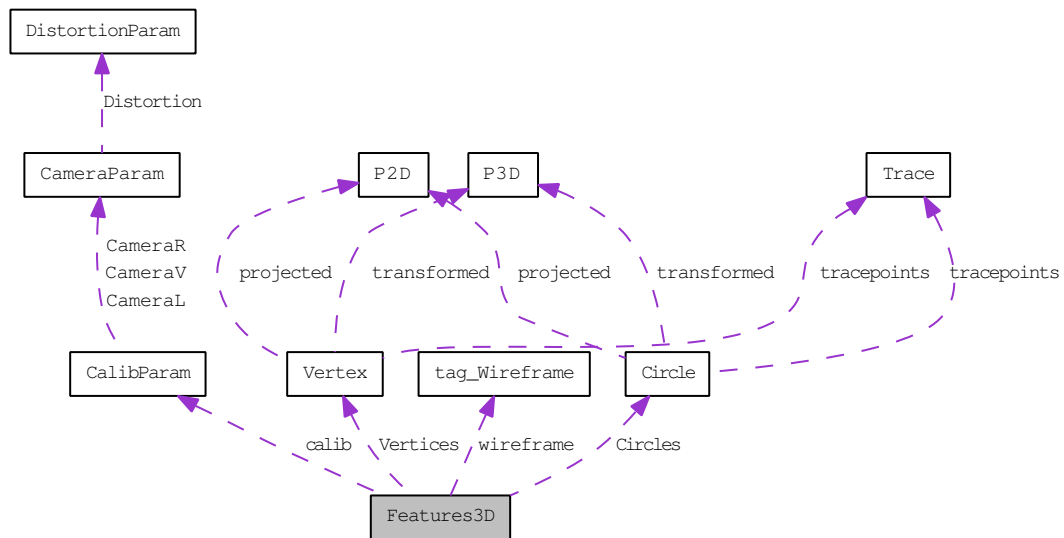
この構造体の説明は次のファイルから生成されました:

- `extractFeature_old.h`

6.27 構造体 Features3D

3次元特徴情報

#include <match3Dfeature.h>Features3D のコラボレーション図



変数

- **CalibParam * calib**
キャリブレーションデータポインタ
- **int numOfVertices**
3次元頂点特徴数
- **Vertex * Vertices**
3次元頂点特徴
- **Wireframe wireframe**
ワイヤフレームデータ
- **int numOfCircles**
3次元円特徴数
- **Circle * Circles**
3次元円特徴

- `uchar * edge [3]`
エッジ画像ポインタ
- `int pointCounts`
2次元評価のための全評価点数
- `double traceCounts`
2次元評価に用いた評価点数

6.27.1 説明

3次元特徴情報

6.27.2 構造体

6.27.2.1 `CalibParam* Features3D::calib`

キャリブレーションデータポインタ

6.27.2.2 `Circle* Features3D::Circles`

3次元円特徴

6.27.2.3 `uchar* Features3D::edge[3]`

エッジ画像ポインタ

6.27.2.4 `int Features3D::numOfCircles`

3次元円特徴数

6.27.2.5 `int Features3D::numOfVertices`

3次元頂点特徴数

6.27.2.6 `int Features3D::pointCounts`

2次元評価のための全評価点数

6.27.2.7 double Features3D::traceCounts

2次元評価に用いた評価点数

6.27.2.8 Vertex* Features3D::Vertices

3次元頂点特徴

6.27.2.9 Wireframe Features3D::wireframe

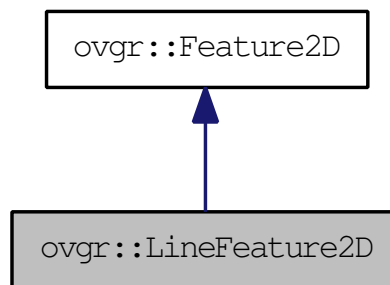
ワイヤフレームデータ

この構造体の説明は次のファイルから生成されました:

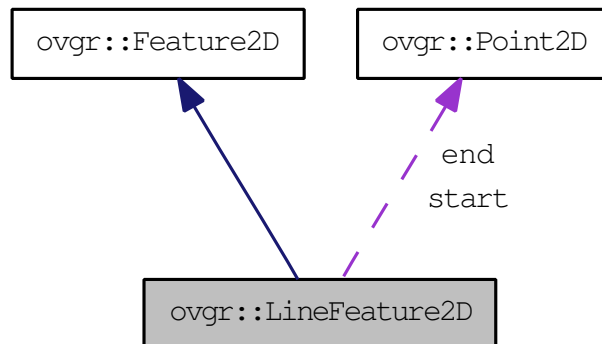
- [match3Dfeature.h](#)

6.28 構造体 `ovgr::LineFeature2D`

`#include <extractFeature.hpp>` `ovgr::LineFeature2D` に対する継承グラフ



`ovgr::LineFeature2D` のコラボレーション図



変数

- double `coef` [3]
 $coef[0]*x + coef[1]*y + coef[2] = 0$
- `Point2D` `start`
- `Point2D` `end`
 始点・終点
- double `length`
 線分の長さ

6.28.1 構造体

6.28.1.1 double ovgr::LineFeature2D::coef[3]

$\text{coef}[0]*x + \text{coef}[1]*y + \text{coef}[2] = 0$

6.28.1.2 Point2D ovgr::LineFeature2D::end

始点・終点

6.28.1.3 double ovgr::LineFeature2D::length

線分の長さ

6.28.1.4 Point2D ovgr::LineFeature2D::start

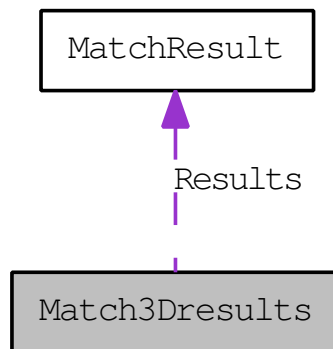
この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

6.29 構造体 Match3Dresults

全認識結果

#include <match3Dfeature.h> Match3Dresults のコラボレーション図



変数

- `int error`
認識エラーフラグ
- `int numOfResults`
認識結果数
- `MatchResult * Results`
各認識結果

6.29.1 説明

全認識結果

6.29.2 構造体

6.29.2.1 `int Match3Dresults::error`

認識エラーフラグ

6.29.2.2 int Match3Dresults::numOfResults

認識結果数

6.29.2.3 MatchResult* Match3Dresults::Results

各認識結果

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.30 構造体 MatchResult

各認識結果情報

```
#include <match3Dfeature.h>
```

変数

- int `n`
通し番号
- int `type`
特徴タイプ 0:頂点、1:単円、2: 2 円
- int `scene` [2]
シーン特徴番号
- int `model` [2]
モデル特徴番号
- double `score`
2 次元評価値
- double `mat` [4][4]
変換行列
- double `vec` [7]
変換行列の 7 次元のベクトル (位置 + 回転) 表現
- int `npoint`
総投影点数
- int `cpoint`
総対応点数

6.30.1 説明

各認識結果情報

6.30.2 構造体

6.30.2.1 `int MatchResult::cpoint`

総対応点数

6.30.2.2 `double MatchResult::mat[4][4]`

変換行列

6.30.2.3 `int MatchResult::model[2]`

モデル特徴番号

6.30.2.4 `int MatchResult::n`

通し番号

6.30.2.5 `int MatchResult::npoint`

総投影点数

6.30.2.6 `int MatchResult::scene[2]`

シーン特徴番号

6.30.2.7 `double MatchResult::score`

2次元評価値

6.30.2.8 `int MatchResult::type`

特徴タイプ 0:頂点、1:単円、2:2 円

6.30.2.9 `double MatchResult::vec[7]`

変換行列の7次元のベクトル（位置 + 回転）表現

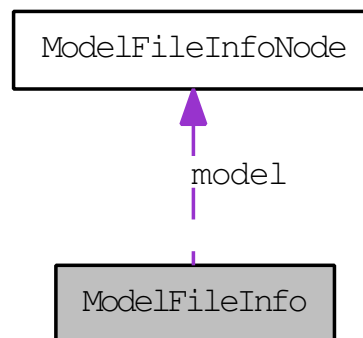
この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.31 構造体 ModelFileInfo

モデルファイルリスト

#include <modelListFileIO.h>ModelFileInfo のコラボレーション図



変数

- `ModelFileInfoNode * model`
モデルリスト
- `int modelNum`
モデル数

6.31.1 説明

モデルファイルリスト

6.31.2 構造体

6.31.2.1 ModelFileInfoNode* ModelFileInfo::model

モデルリスト

6.31.2.2 `int ModelFileInfo::modelNum`

モデル数

この構造体の説明は次のファイルから生成されました:

- [modelListFileIO.h](#)

6.32 構造体 `ModelFileInfoNode`

モデルリストのノード

```
#include <modelListFileIO.h>
```

変数

- `int id`
モデル *ID*
- `char path [MAX_PATH]`
モデルファイル名

6.32.1 説明

モデルリストのノード

6.32.2 構造体

6.32.2.1 `int ModelFileInfoNode::id`

モデル ID

6.32.2.2 `char ModelFileInfoNode::path[MAX_PATH]`

モデルファイル名

この構造体の説明は次のファイルから生成されました:

- [modelListFileIO.h](#)

6.33 構造体 P2D

2次元位置情報

```
#include <match3Dfeature.h>
```

変数

- double [colrow](#) [2]
2次元座標

6.33.1 説明

2次元位置情報

6.33.2 構造体

6.33.2.1 double P2D::colrow[2]

2次元座標

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.34 構造体 P3D

3次元位置情報

```
#include <match3Dfeature.h>
```

変数

- `double xyz[3]`
3次元座標

6.34.1 説明

3次元位置情報

6.34.2 構造体

6.34.2.1 `double P3D::xyz[3]`

3次元座標

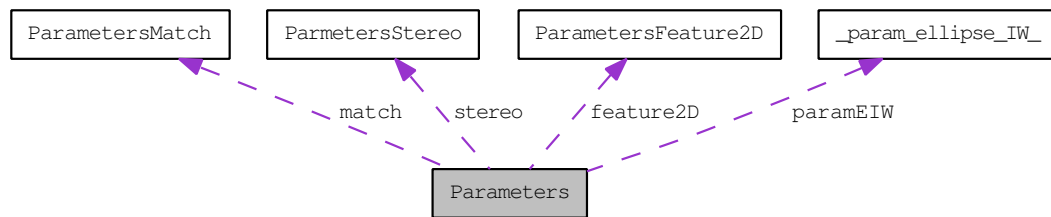
この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.35 構造体 Parameters

全パラメータ

#include <parameters.h>Parameters のコラボレーション図



変数

- `ParametersFeature2D feature2D`
- `ParametersStereo stereo`
- `ParametersMatch match`
- `StereoPairing pairing`
- `ParamEllipseIW paramEIW`
- `int outputCandNum`
出力候補数
- `int colsize`
画像サイズ横（画素）
- `int rowsize`
画像サイズ縦（画素）
- `int imgsize`
画像サイズ総画素数
- `int dbgtext`
デバッグテキスト生成
- `int dbgimag`
デバッグ画像生成
- `int dbgdisp`
デバッグ画像表示

6.35.1 説明

全パラメータ

6.35.2 構造体

6.35.2.1 `int Parameters::colsize`

画像サイズ横（画素）

6.35.2.2 `int Parameters::dbgdisp`

デバッグ画像表示

6.35.2.3 `int Parameters::dbgimag`

デバッグ画像生成

6.35.2.4 `int Parameters::dbgtext`

デバッグテキスト生成

6.35.2.5 `ParametersFeature2D Parameters::feature2D`

6.35.2.6 `int Parameters::imgsize`

画像サイズ総画素数

6.35.2.7 `ParametersMatch Parameters::match`

6.35.2.8 `int Parameters::outputCandNum`

出力候補数

6.35.2.9 `StereoPairing Parameters::pairing`

6.35.2.10 ParamEllipseIW Parameters::paramEIW**6.35.2.11 int Parameters::rowsize**

画像サイズ縦（画素）

6.35.2.12 ParametersStereo Parameters::stereo

この構造体の説明は次のファイルから生成されました:

- [parameters.h](#)

6.36 構造体 ParametersFeature2D

2次元特徴抽出用パラメータ

```
#include <parameters.h>
```

変数

- int [edgeDetectFunction](#)
エッジ検出アルゴリズム (0: Sobel3x3 1: Sobel 5x5)
- double [edgeStrength](#)
検出するエッジの最低微分強度
- int [minFragment](#)
検出するエッジの最低外周長 (画素)
- double [maxErrorofLineFit](#)
直線をあてはめる時の最大誤差 (画素)
- double [max_distance_similar_line](#)
同一の線分と見なす最大端点距離 (画素)
- double [maxErrorofConicFit](#)
二次曲線をあてはめる時の最大誤差 (画素)
- double [overlapRatioLine](#)
直線、双曲線の特徴点を抽出する区間の重複可能な最大比率 (0.0 から 1.0)
- double [overlapRatioCircle](#)
楕円の特徴点を抽出する区間の重複可能な最大比率 (0.0 から 1.0)
- double [max_length_delete_line](#)
削除する直線の最大の長さ (画素)
- double [min_radian_hyperbola](#)
双曲線のなす角度閾値 (0,180 度に近いものを除去する) (ラジアン)
- double [min_length_hyperbola_data](#)
双曲線での中心からデータまでの距離の閾値 (画素)
- double [min_length_hyperbola_vector](#)
双曲線での中心から端点までの距離の閾値 (画素)

- double `min_length_ellipse_axis`
楕円の軸長の閾値 (画素)
- double `min_filling_ellipse`
楕円の充填率の閾値 (0.0 から 1.0)
- double `max_flatness_ellipse`
楕円の偏平率 (長軸/短軸)
- double `max_distance_end_points`
端点間距離の閾値 (画素)
- double `min_length_line`
直線の長さの閾値 (画素)
- double `max_distance_ellipse_grouping`
中心距離判定閾値 (画素)
- double `min_distance_ellipse_pairing`
中心距離判定閾値 (画素)
- double `max_distance_ellipse_pairing`
中心距離判定閾値 (画素)
- double `min_length_ellipse_axisS`
楕円の軸長の閾値 (画素)
- double `min_length_ellipse_axisL`
楕円の軸長の閾値 (画素)
- double `max_length_ellipse_axisL`
楕円の軸長の閾値 (画素)
- int `no_search_features`
検出しない特徴のフラグ

6.36.1 説明

2次元特徴抽出用パラメータ

6.36.2 構造体

6.36.2.1 int ParametersFeature2D::edgeDetectFunction

エッジ検出アルゴリズム (0: Sobel3x3 1: Sobel 5x5)

6.36.2.2 double ParametersFeature2D::edgeStrength

検出するエッジの最低微分強度

6.36.2.3 double ParametersFeature2D::max_distance_ellipse_grouping

中心距離判定閾値 (画素)

6.36.2.4 double ParametersFeature2D::max_distance_ellipse_pairing

中心距離判定閾値 (画素)

6.36.2.5 double ParametersFeature2D::max_distance_end_points

端点間距離の閾値 (画素)

6.36.2.6 double ParametersFeature2D::max_distance_similar_line

同一の線分と見なす最大端点距離 (画素)

6.36.2.7 double ParametersFeature2D::max_flatness_ellipse

楕円の偏平率 (長軸/短軸)

6.36.2.8 double ParametersFeature2D::max_length_delete_line

削除する直線の最大の長さ (画素)

6.36.2.9 double ParametersFeature2D::max_length_ellipse_axisL

楕円の軸長の閾値 (画素)

6.36.2.10 double ParametersFeature2D::maxErrorofConicFit

二次曲線をあてはめる時の最大誤差 (画素)

6.36.2.11 double ParametersFeature2D::maxErrorofLineFit

直線をあてはめる時の最大誤差 (画素)

6.36.2.12 double ParametersFeature2D::min_distance_ellipse_pairing

中心距離判定閾値 (画素)

6.36.2.13 double ParametersFeature2D::min_filling_ellipse

楕円の充填率の閾値 (0.0 から 1.0)

6.36.2.14 double ParametersFeature2D::min_length_ellipse_axis

楕円の軸長の閾値 (画素)

6.36.2.15 double ParametersFeature2D::min_length_ellipse_axisL

楕円の軸長の閾値 (画素)

6.36.2.16 double ParametersFeature2D::min_length_ellipse_axisS

楕円の軸長の閾値 (画素)

6.36.2.17 double ParametersFeature2D::min_length_hyperbola_data

双曲線での中心からデータまでの距離の閾値 (画素)

6.36.2.18 double ParametersFeature2D::min_length_hyperbola_vector

双曲線での中心から端点までの距離の閾値 (画素)

6.36.2.19 double ParametersFeature2D::min_length_line

直線の長さの閾値 (画素)

6.36.2.20 double ParametersFeature2D::min_radian_hyperbola

双曲線のなす角度閾値 (0,180 度に近いものを除去する) (ラジアン)

6.36.2.21 int ParametersFeature2D::minFragment

検出するエッジの最低外周長 (画素)

6.36.2.22 int ParametersFeature2D::no_search_features

検出しない特徴のフラグ

6.36.2.23 double ParametersFeature2D::overlapRatioCircle

楕円の特徴点を抽出する区間の重複可能な最大比率 (0.0 から 1.0)

6.36.2.24 double ParametersFeature2D::overlapRatioLine

直線、双曲線の特徴点を抽出する区間の重複可能な最大比率 (0.0 から 1.0)

この構造体の説明は次のファイルから生成されました:

- [parameters.h](#)

6.37 構造体 ParametersMatch

認識用パラメータ

```
#include <parameters.h>
```

変数

- double [tolerance1](#)
頂点の角度差の許容割合 (%)
- double [tolerance2](#)
円の半径の差の許容割合 (%)
- double [pdist](#)
モデルサンプル点間隔 (mm)

6.37.1 説明

認識用パラメータ

6.37.2 構造体

6.37.2.1 double ParametersMatch::pdist

モデルサンプル点間隔 (mm)

6.37.2.2 double ParametersMatch::tolerance1

頂点の角度差の許容割合 (%)

6.37.2.3 double ParametersMatch::tolerance2

円の半径の差の許容割合 (%)

この構造体の説明は次のファイルから生成されました:

- [parameters.h](#)

6.38 構造体 ParmetersStereo

ステレオ対応処理用パラメータ

```
#include <parameters.h>
```

変数

- double [ethr](#)
対応誤差閾値 (*mm*)
- double [rdif](#)
半径許容差 (*mm*)
- double [ndif](#)
左右法線角度許容差 (度)
- double [depn](#)
円中心・頂点位置奥行開始 (*mm*)
- double [depf](#)
円中心・頂点位置奥行終了 (*mm*)
- double [amin](#)
頂点 角度最小値 (度)
- double [amax](#)
頂点 角度最大値 (度)

6.38.1 説明

ステレオ対応処理用パラメータ

6.38.2 構造体

6.38.2.1 double ParmetersStereo::amax

頂点 角度最大値 (度)

6.38.2.2 double `ParmetersStereo::amin`

頂点 角度最小値 (度)

6.38.2.3 double `ParmetersStereo::depf`

円中心・頂点位置奥行終了 (mm)

6.38.2.4 double `ParmetersStereo::depn`

円中心・頂点位置奥行開始 (mm)

6.38.2.5 double `ParmetersStereo::ethr`

対応誤差閾値 (mm)

6.38.2.6 double `ParmetersStereo::ndif`

左右法線角度許容差 (度)

6.38.2.7 double `ParmetersStereo::rdif`

半径許容差 (mm)

この構造体の説明は次のファイルから生成されました:

- [parameters.h](#)

6.39 構造体 `ovgr::Point2D`

```
#include <geometry.hpp>
```

Public メソッド

- `Point2D ()`
- `Point2D (const double _x, const double _y)`
- `template<class T >`
`Point2D (const cv::Point_< T > point)`
- `template<class T >`
`operator cv::Point_< T > () const`

変数

- `double x`
- `double y`

6.39.1 コンストラクタとデストラクタ

6.39.1.1 `ovgr::Point2D::Point2D ()` [`inline`]

6.39.1.2 `ovgr::Point2D::Point2D (const double _x, const double _y)`
[`inline`]

6.39.1.3 `template<class T > ovgr::Point2D::Point2D (const cv::Point_< T >`
`point)` [`inline`]

6.39.2 関数

6.39.2.1 `template<class T > ovgr::Point2D::operator cv::Point_< T > () const`
 `[inline]`

6.39.3 構造体

6.39.3.1 `double ovgr::Point2D::x`

6.39.3.2 `double ovgr::Point2D::y`

この構造体の説明は次のファイルから生成されました:

- [geometry.hpp](#)

6.40 クラス `ovgr::PointsOnEllipse`

```
#include <drawing.hpp>
```

このクラスの説明は次のファイルから生成されました:

- [drawing.hpp](#)

6.41 クラス ovgr::PointsOnLine

```
#include <drawing.hpp>
```

Public メソッド

- [PointsOnLine](#) (const int x1, const int y1, const int x2, const int y2)
- int [x](#) () const
- int [y](#) () const
- int [next](#) ()

Protected 変数

- int [m_x](#)
- int [m_y](#)
- int [m_ex](#)
- int [m_ey](#)
- int [m_xinc](#)
- int [m_yinc](#)
- int [m_e](#)
- int [m_dx](#)
- int [m_dy](#)

6.41.1 コンストラクタとデストラクタ

6.41.1.1 ovgr::PointsOnLine::PointsOnLine (const int x1, const int y1, const int x2, const int y2)

6.41.2 関数

6.41.2.1 int ovgr::PointsOnLine::next ()

6.41.2.2 int ovgr::PointsOnLine::x () const [inline]

6.41.2.3 `int ovgr::PointsOnLine::y () const [inline]`

6.41.3 構造体

6.41.3.1 `int ovgr::PointsOnLine::m_dx [protected]`

6.41.3.2 `int ovgr::PointsOnLine::m_dy [protected]`

6.41.3.3 `int ovgr::PointsOnLine::m_e [protected]`

6.41.3.4 `int ovgr::PointsOnLine::m_ex [protected]`

6.41.3.5 `int ovgr::PointsOnLine::m_ey [protected]`

6.41.3.6 `int ovgr::PointsOnLine::m_x [protected]`

6.41.3.7 `int ovgr::PointsOnLine::m_xinc [protected]`

6.41.3.8 `int ovgr::PointsOnLine::m_y [protected]`

6.41.3.9 `int ovgr::PointsOnLine::m_yinc [protected]`

このクラスの説明は次のファイルから生成されました:

- [drawing.hpp](#)

- [drawing.cpp](#)

6.42 構造体 RTVCM_Box

モデル内の立方体データ

```
#include <rtvcm.h>
```

変数

- int **n**
通し番号
- double **x**
• double **y**
• double **z**
幅、奥行き、高さ
- double **Rotate** [3][3]
基準位置からの回転
- double **Trans** [3]
基準位置からの移動
- int **nVertex** [24]
頂点通し番号列
- void * **reserved**
拡張

6.42.1 説明

モデル内の立方体データ

6.42.2 構造体

6.42.2.1 int RTVCM_Box::n

通し番号

6.42.2.2 int RTVCM_Box::nVertex[24]

頂点通し番号列

6.42.2.3 void* RTVCM_Box::reserved

拡張

6.42.2.4 double RTVCM_Box::Rotate[3][3]

基準位置からの回転

6.42.2.5 double RTVCM_Box::Trans[3]

基準位置からの移動

6.42.2.6 double RTVCM_Box::x**6.42.2.7 double RTVCM_Box::y****6.42.2.8 double RTVCM_Box::z**

幅、奥行き、高さ

この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

6.43 構造体 RTVCM_Circle

モデル内の円データ

```
#include <rtvcm.h>
```

変数

- int **n**
通し番号
- double **radius**
半径
- double **center** [3]
中心の 3 次元位置
- double **normal** [3]
3 次元法線方向
- int **ncyliner**
属する円筒の通し番号
- void * **reserved**
拡張

6.43.1 説明

モデル内の円データ

6.43.2 構造体

6.43.2.1 double RTVCM_Circle::center[3]

中心の 3 次元位置

6.43.2.2 int RTVCM_Circle::n

通し番号

6.43.2.3 int RTVCM_Circle::ncyliner

属する円筒の通し番号

6.43.2.4 double RTVCM_Circle::normal[3]

3次元法線方向

6.43.2.5 double RTVCM_Circle::radius

半径

6.43.2.6 void* RTVCM_Circle::reserved

拡張

この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

6.44 構造体 RTVCM_Cylinder

モデル内の円筒データ

```
#include <rtvcm.h>
```

変数

- int **n**
通し番号
- double **radius**
半径
- double **height**
高さ
- double **Rotate** [3][3]
基準位置からの回転
- double **Trans** [3]
基準位置からの移動
- int * **nCircle** [2]
構成円の通し番号配列
- void * **reserved**
拡張

6.44.1 説明

モデル内の円筒データ

6.44.2 構造体

6.44.2.1 double RTVCM_Cylinder::height

高さ

6.44.2.2 int RTVCM_Cylinder::n

通し番号

6.44.2.3 int* RTVCM_Cylinder::nCircle[2]

構成円の通し番号配列

6.44.2.4 double RTVCM_Cylinder::radius

半径

6.44.2.5 void* RTVCM_Cylinder::reserved

拡張

6.44.2.6 double RTVCM_Cylinder::Rotate[3][3]

基準位置からの回転

6.44.2.7 double RTVCM_Cylinder::Trans[3]

基準位置からの移動

この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

6.45 構造体 RTVCM_Vertex

モデル内の頂点データ

```
#include <rtvcm.h>
```

変数

- int `n`
通し番号
- double `position` [3]
3次元位置
- double `endpoint1` [3]
3次元端点 1
- double `endpoint2` [3]
3次元端点 2
- double `angle`
2直線のなす角度
- int `nbox`
属する立方体の通し番号
- void * `reserved`
拡張

6.45.1 説明

モデル内の頂点データ

6.45.2 構造体

6.45.2.1 double RTVCM_Vertex::angle

2直線のなす角度

6.45.2.2 double RTVCM_Vertex::endpoint1[3]

3 次元端点 1

6.45.2.3 double RTVCM_Vertex::endpoint2[3]

3 次元端点 2

6.45.2.4 int RTVCM_Vertex::n

通し番号

6.45.2.5 int RTVCM_Vertex::nbox

属する立方体の通し番号

6.45.2.6 double RTVCM_Vertex::position[3]

3 次元位置

6.45.2.7 void* RTVCM_Vertex::reserved

拡張

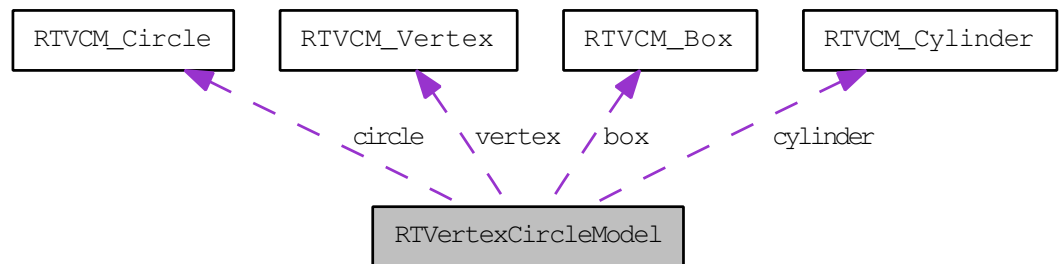
この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

6.46 構造体 RTVertexCircleModel

モデルデータ構造体

#include <rtvcm.h>RTVertexCircleModel のコラボレーション図



変数

- `RTVCM_Label label`
属性
- `int n`
通し番号
- `double gravity [3]`
重心
- `double width`
属性が立方体の場合は幅 (x), (円柱の場合は外接矩形の幅 (x) に使っても良い)
- `double height`
属性が立方体の場合は奥行き (y), 円柱の場合は高さ (y)
- `double depth`
属性が立方体の場合は高さ (z), (円柱の場合は外接矩形の奥行き (z) に使っても良い)
- `double radius`
属性が円柱の場合の半径
- `int nvertex`
頂点数

- `RTVCM_Vertex * vertex`
頂点列
- `int ncircle`
円数
- `RTVCM_Circle * circle`
円列
- `int nbox`
直方体数
- `RTVCM_Box * box`
直方体列 (表示用)
- `int ncylinder`
円筒数
- `RTVCM_Cylinder * cylinder`
円筒列 (表示用)
- `void * reserved`
拡張

6.46.1 説明

モデルデータ構造体

6.46.2 構造体

6.46.2.1 `RTVCM_Box* RTVertexCircleModel::box`

直方体列 (表示用)

6.46.2.2 `RTVCM_Circle* RTVertexCircleModel::circle`

円列

6.46.2.3 `RTVCM_Cylinder* RTVertexCircleModel::cylinder`

円筒列 (表示用)

6.46.2.4 double RTVertexCircleModel::depth

属性が立方体の場合は高さ (z), (円柱の場合は外接矩形の奥行き (z) に使っても良い)

6.46.2.5 double RTVertexCircleModel::gravity[3]

重心

6.46.2.6 double RTVertexCircleModel::height

属性が立方体の場合は奥行き (y), 円柱の場合は高さ (y)

6.46.2.7 RTVCM_Label RTVertexCircleModel::label

属性

6.46.2.8 int RTVertexCircleModel::n

通し番号

6.46.2.9 int RTVertexCircleModel::nbox

直方体数

6.46.2.10 int RTVertexCircleModel::ncircle

円数

6.46.2.11 int RTVertexCircleModel::ncylinder

円筒数

6.46.2.12 int RTVertexCircleModel::nvertex

頂点数

6.46.2.13 double RTVertexCircleModel::radius

属性が円柱の場合の半径

6.46.2.14 void* RTVertexCircleModel::reserved

拡張

6.46.2.15 RTVCM_Vertex* RTVertexCircleModel::vertex

頂点列

6.46.2.16 double RTVertexCircleModel::width

属性が立方体の場合は幅 (x), (円柱の場合は外接矩形の幅 (x) に使っても良い)

この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

6.47 構造体 `tag_Wireframe::Segment`

< 線分

```
#include <match3Dfeature.h>
```

Public メソッド

- [Segment](#) ()
- [Segment](#) (const int id1, const int id2)

変数

- int [vertex_id](#) [2]

6.47.1 説明

< 線分

6.47.2 コンストラクタとデストラクタ

6.47.2.1 `tag_Wireframe::Segment::Segment () [inline]`

6.47.2.2 `tag_Wireframe::Segment::Segment (const int id1, const int id2) [inline]`

6.47.3 構造体

6.47.3.1 `int tag_Wireframe::Segment::vertex_id[2]`

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.48 構造体 StereoCalib

ステレオカメラキャリブレーションデータ

```
#include <stereo.h>
```

変数

- int `numOfCameras`
ステレオセットのカメラ数
- int `width`
画像幅
- int `height`
画像高さ
- double `baselineLR`
LR 間ベースライン長.
- double `baselineLV`
LV 間ベースライン長.
- double `baselineRV`
RV 間ベースライン長.

6.48.1 説明

ステレオカメラキャリブレーションデータ

6.48.2 構造体

6.48.2.1 double StereoCalib::baselineLR

LR 間ベースライン長.

6.48.2.2 double StereoCalib::baselineLV

LV 間ベースライン長.

6.48.2.3 double StereoCalib::baselineRV

RV 間ベースライン長.

6.48.2.4 int StereoCalib::height

画像高さ

6.48.2.5 int StereoCalib::numOfCameras

ステレオセットのカメラ数

6.48.2.6 int StereoCalib::width

画像幅

この構造体の説明は次のファイルから生成されました:

- [stereo.h](#)

6.49 構造体 tag_Wireframe

ワイヤフレームモデル

```
#include <match3Dfeature.h>
```

データ構造

- struct [Face](#)
 < 面
- struct [Segment](#)
 < 線分

変数

- int [num_vertices](#)
 頂点数
- double(* [vertex](#))[3]
 頂点座標の配列
- std::vector< [Segment](#) > [segment](#)
 線分情報の配列
- std::vector< [Face](#) > [face](#)
 面情報の配列

6.49.1 説明

ワイヤフレームモデル

6.49.2 構造体

6.49.2.1 std::vector<Face> tag_Wireframe::face

面情報の配列

6.49.2.2 int tag_Wireframe::num_vertices

頂点数

6.49.2.3 std::vector<Segment> tag_Wireframe::segment

線分情報の配列

6.49.2.4 double(* tag_Wireframe::vertex)[3]

頂点座標の配列

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.50 構造体 Trace

認識結果評価用サンプリング点列情報

```
#include <match3Dfeature.h>
```

変数

- int `label`
ラベル：可視情報 (*VISIBLE/INVISIBLE*)
- double `weight`
- double `xyz` [4]
3次元位置情報 (*mm*)
- double `colrow` [2]
画像投影位置情報（画素）
- int `direction`
対応エッジ探索方向
- int `search`
対応エッジの距離（画素）
- int `edge`
対応エッジの強度
- double `peakcr` [2]
対応エッジ点の位置（画素）

6.50.1 説明

認識結果評価用サンプリング点列情報

6.50.2 構造体

6.50.2.1 double Trace::colrow[2]

画像投影位置情報（画素）

6.50.2.2 int Trace::direction

対応エッジ探索方向

6.50.2.3 int Trace::edge

対応エッジの強度

6.50.2.4 int Trace::label

ラベル：可視情報 (VISIBLE/INVISIBLE)

6.50.2.5 double Trace::peakcr[2]

対応エッジ点の位置（画素）

6.50.2.6 int Trace::search

対応エッジの距離（画素）

6.50.2.7 double Trace::weight**6.50.2.8 double Trace::xyz[4]**

3次元位置情報 (mm)

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.51 構造体 Track

輪郭情報

```
#include <extractFeature_old.h>
```

変数

- int [nPoint](#)
点数
- int * [Point](#)
点列
- double [offset](#) [2]
楕円係数計算時のオフセット

6.51.1 説明

輪郭情報

6.51.2 構造体

6.51.2.1 int Track::nPoint

点数

6.51.2.2 double Track::offset[2]

楕円係数計算時のオフセット

6.51.2.3 int* Track::Point

点列

この構造体の説明は次のファイルから生成されました:

- [extractFeature_old.h](#)

6.52 クラス テンプレート `ovgr::VariableWatcher< T, Equal >`

```
#include <debugutil.h>
```

Public メソッド

- `VariableWatcher` (T &val)
- `bool is_changed ()`

```
template<class T, class Equal = EqualOp<T>> class ovgr::VariableWatcher<  
T, Equal >
```

6.52.1 コンストラクタとデストラクタ

```
6.52.1.1 template<class T , class Equal = EqualOp<T>>  
ovgr::VariableWatcher< T, Equal >::VariableWatcher (T & val)  
[inline]
```

6.52.2 関数

```
6.52.2.1 template<class T , class Equal = EqualOp<T>> bool  
ovgr::VariableWatcher< T, Equal >::is_changed () [inline]
```

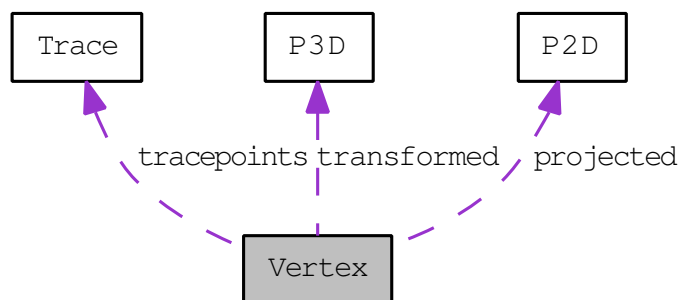
このクラスの説明は次のファイルから生成されました:

- `debugutil.h`

6.53 構造体 Vertex

3次元頂点情報

#include <match3Dfeature.h>Vertex のコラボレーション図



変数

- int `label`
ラベル
- int `n`
通し番号
- int `side`
表裏情報
- double `endpoint1` [3]
辺の端点 (*mm*)
- double `endpoint2` [3]
辺の端点 (*mm*)
- double `direction1` [3]
辺の方向
- double `direction2` [3]
辺の方向
- double `tPose` [4][4]
認識用姿勢行列

- double `angle`
頂点角度 (ラジアン)
- int `numOfTracePoints`
認識評価用のサンプリング点列数
- `Trace * tracepoints`
認識評価用のサンプリング点列情報
- `P3D * transformed`
認識時の位置・姿勢変換後の 3 次元点列 (*mm*)
- `P2D * projected`
2 次元評価時の画像投影 2 次元点列 (画素)

6.53.1 説明

3 次元頂点情報

6.53.2 構造体

6.53.2.1 double `Vertex::angle`

頂点角度 (ラジアン)

6.53.2.2 double `Vertex::direction1[3]`

辺の方向

6.53.2.3 double `Vertex::direction2[3]`

辺の方向

6.53.2.4 double `Vertex::endpoint1[3]`

辺の端点 (*mm*)

6.53.2.5 double `Vertex::endpoint2[3]`

辺の端点 (*mm*)

6.53.2.6 int Vertex::label

ラベル

6.53.2.7 int Vertex::n

通し番号

6.53.2.8 int Vertex::numOfTracePoints

認識評価用のサンプリング点列数

6.53.2.9 P2D* Vertex::projected

2次元評価時の画像投影 2次元点列 (画素)

6.53.2.10 int Vertex::side

表裏情報

6.53.2.11 double Vertex::tPose[4][4]

認識用姿勢行列

6.53.2.12 Trace* Vertex::tracepoints

認識評価用のサンプリング点列情報

6.53.2.13 P3D* Vertex::transformed

認識時の位置・姿勢変換後の3次元点列 (mm)

この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

6.54 構造体 VertexCandidate

三次元頂点特徴候補データ

```
#include <stereo.h>
```

変数

- int **valid**
有効・無効フラグ
- int **n1**
for debug
- int **n2**
for debug
- int **n3**
for debug
- double **angle**
頂点を成す線分の角度
- double **position** [3]
頂点座標
- double **endpoint1** [3]
端点 1 座標
- double **endpoint2** [3]
端点 2 座標
- double **vector1** [3]
端点 1 向き単位ベクトル
- double **vector2** [3]
端点 2 向き単位ベクトル
- double **len1**
線分 1 の長さ
- double **len2**
線分 2 の長さ

6.54.1 説明

三次元頂点特徴候補データ

6.54.2 構造体

6.54.2.1 double VertexCandidate::angle

頂点を成す線分の角度

6.54.2.2 double VertexCandidate::endpoint1[3]

端点 1 座標

6.54.2.3 double VertexCandidate::endpoint2[3]

端点 2 座標

6.54.2.4 double VertexCandidate::len1

線分 1 の長さ

6.54.2.5 double VertexCandidate::len2

線分 2 の長さ

6.54.2.6 int VertexCandidate::n1

for debug

6.54.2.7 int VertexCandidate::n2

for debug

6.54.2.8 int VertexCandidate::n3

for debug

6.54.2.9 double VertexCandidate::position[3]

頂点座標

6.54.2.10 int VertexCandidate::valid

有効・無効フラグ

6.54.2.11 double VertexCandidate::vector1[3]

端点 1 向き単位ベクトル

6.54.2.12 double VertexCandidate::vector2[3]

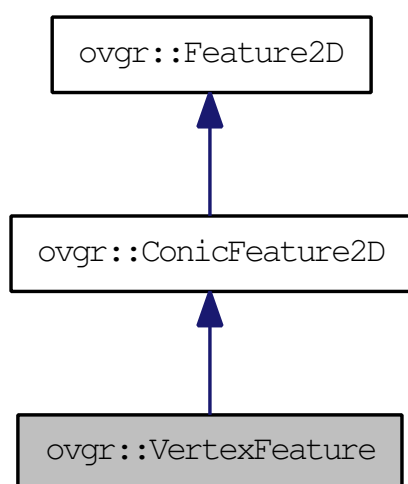
端点 2 向き単位ベクトル

この構造体の説明は次のファイルから生成されました:

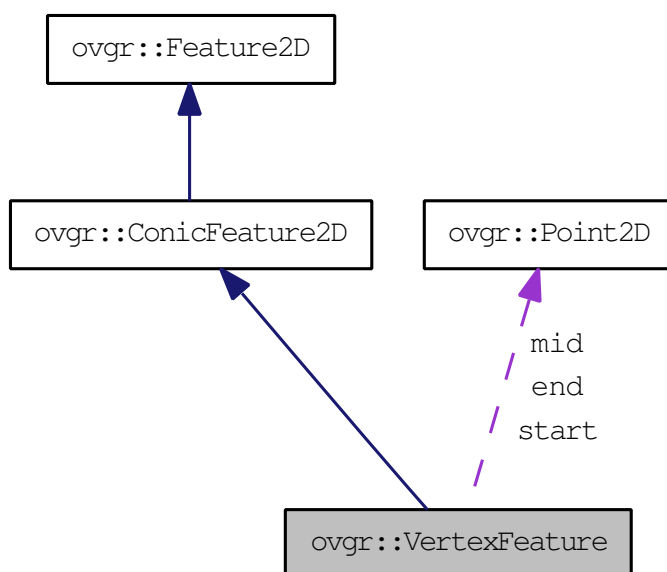
- [stereo.h](#)

6.55 構造体 ovgr::VertexFeature

#include <extractFeature.hpp> ovgr::VertexFeature に対する継承グラフ



ovgr::VertexFeature のコラボレーション図



変数

- double [line_coef](#) [2][3]
各線分の係数
- [Point2D start](#)
- [Point2D mid](#)
- [Point2D end](#)
始点・中点・終点
- double [length](#) [2]
線分の長さ

6.55.1 構造体

6.55.1.1 [Point2D ovgr::VertexFeature::end](#)

始点・中点・終点

6.55.1.2 [double ovgr::VertexFeature::length\[2\]](#)

線分の長さ

6.55.1.3 [double ovgr::VertexFeature::line_coef\[2\]\[3\]](#)

各線分の係数

6.55.1.4 [Point2D ovgr::VertexFeature::mid](#)

6.55.1.5 [Point2D ovgr::VertexFeature::start](#)

この構造体の説明は次のファイルから生成されました:

- [extractFeature.hpp](#)

Chapter 7

ファイル

7.1 calib.cpp

キャリブレーション関連の関数

```
#include <stdio.h>
```

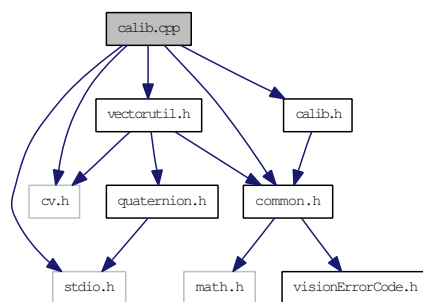
```
#include <cv.h>
```

```
#include "common.h"
```

```
#include "vectorutil.h"
```

```
#include "calib.h"
```

calib.cpp のインクルード依存関係図



関数

- void `undistortPosition` (`Data_2D *icPos`, `Data_2D iPos`, `CameraParam *cameraParam`)
- void `distortPosition` (`Data_2D *iPos2D`, `Data_2D icPos2D`, `CameraParam *cameraParam`)

- void `backprojectPoint` (`Data_2D *icPos`, `Data_2D iPos`, `CameraParam *cameraParam`)
- void `projectPoint` (`Data_2D *iPos2D`, `Data_2D icPos2D`, `CameraParam *cameraParam`)

7.1.1 説明

キャリブレーション関連の関数

日付:

\$Date:: 2011-09-21 18:38:52 +0900 #

7.1.2 関数

7.1.2.1 void `backprojectPoint` (`Data_2D * icPos`, `Data_2D iPos`, `CameraParam * cameraParam`)

7.1.2.2 void `distortPosition` (`Data_2D * iPos2D`, `Data_2D icPos2D`, `CameraParam * cameraParam`)

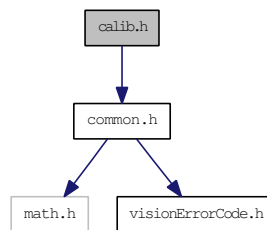
7.1.2.3 void `projectPoint` (`Data_2D * iPos2D`, `Data_2D icPos2D`, `CameraParam * cameraParam`)

7.1.2.4 void `undistortPosition` (`Data_2D * icPos`, `Data_2D iPos`, `CameraParam * cameraParam`)

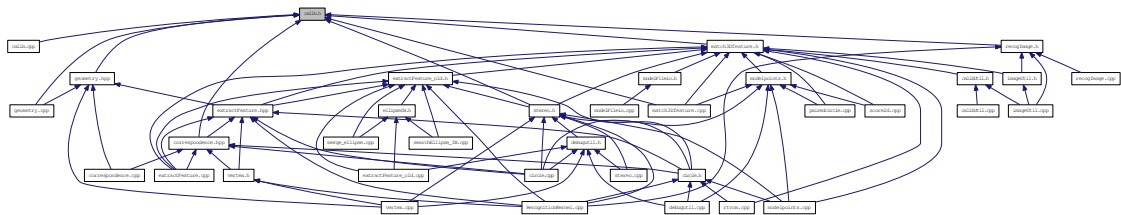
7.2 calib.h

キャリブレーション関連の関数 `#include "common.h"`

calib.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [DistortionParam](#)
歪みパラメータ
- struct [CameraParam](#)
カメラパラメータ
- struct [CalibParam](#)
キャリブレーションパラメータ

関数

- void [undistortPosition](#) ([Data_2D](#) *icPos, [Data_2D](#) iPos, [CameraParam](#) *cameraParam)

- void `distortPosition` (`Data_2D` *iPos2D, `Data_2D` icPos2D, `CameraParam` *cameraParam)
- void `backprojectPoint` (`Data_2D` *icPos, `Data_2D` iPos, `CameraParam` *cameraParam)
- void `projectPoint` (`Data_2D` *iPos2D, `Data_2D` icPos2D, `CameraParam` *cameraParam)

7.2.1 説明

キャリブレーション関連の関数

日付:

\$Date:: 2011-09-09 14:00:23 +0900 #

7.2.2 関数

7.2.2.1 void `backprojectPoint` (`Data_2D` * *icPos*, `Data_2D` *iPos*, `CameraParam` * *cameraParam*)

7.2.2.2 void `distortPosition` (`Data_2D` * *iPos2D*, `Data_2D` *icPos2D*, `CameraParam` * *cameraParam*)

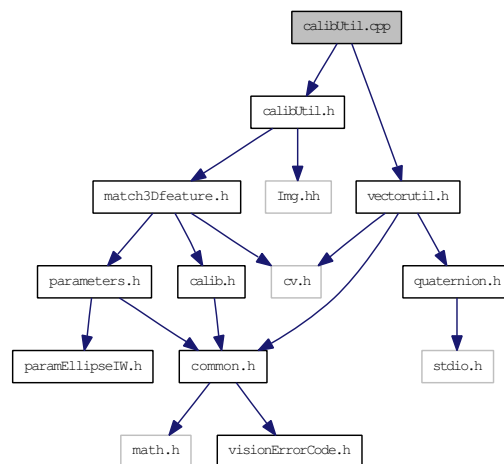
7.2.2.3 void `projectPoint` (`Data_2D` * *iPos2D*, `Data_2D` *icPos2D*, `CameraParam` * *cameraParam*)

7.2.2.4 void `undistortPosition` (`Data_2D` * *icPos*, `Data_2D` *iPos*, `CameraParam` * *cameraParam*)

7.3 calibUtil.cpp

```
#include "calibUtil.h"  
#include "vectorutil.h"
```

calibUtil.cpp のインクルード依存関係図



関数

- void **setCalibFromCameraImage** (const Img::CameraImage &image, CameraParam &camera)

7.3.1 関数

7.3.1.1 void setCalibFromCameraImage (const Img::CameraImage &image, CameraParam &camera)

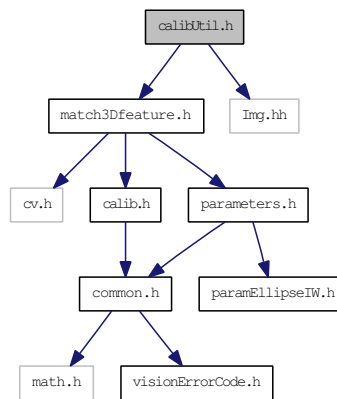
CameraImage 内のキャリブレーションデータを、Calib 構造体にセットする。

7.4 calibUtil.h

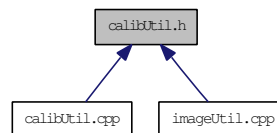
キャリブレーションデータの変換関連 `#include "match3Dfeature.h"`

`#include "Img.hh"`

calibUtil.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- void [setCalibFromCameraImage](#) (const [Img::CameraImage](#) &image, [CameraParam](#) &camera)

7.4.1 説明

キャリブレーションデータの変換関連

7.4.2 関数

7.4.2.1 void setCalibFromCameraImage (const Img::CameraImage & *image*, CameraParam & *camera*)

CameraImage 内のキャリブレーションデータを、 Calib 構造体にセットする。

7.5 circle.cpp

3次元円特徴生成関連関数 `#include "stereo.h"`

`#include "vectorutil.h"`

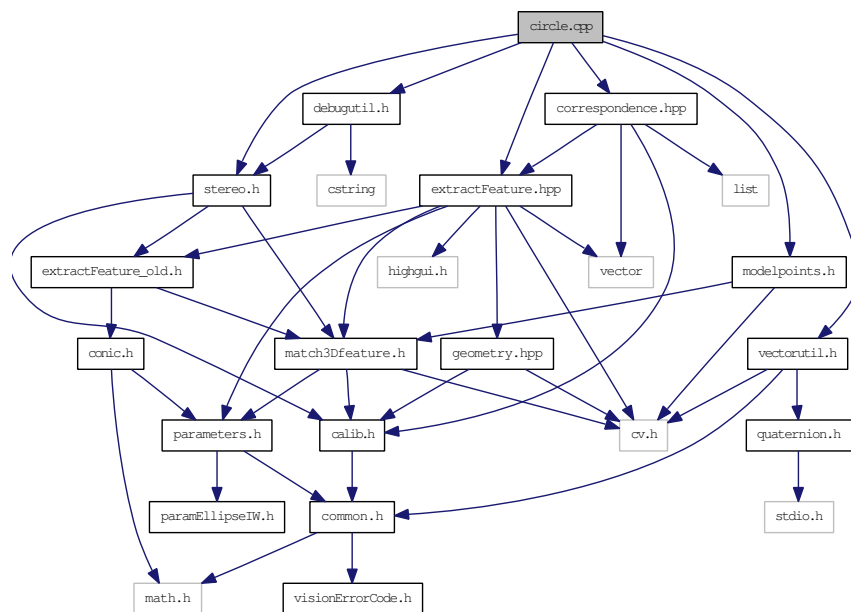
`#include "debugutil.h"`

`#include "modelpoints.h"`

`#include "correspondence.hpp"`

`#include "extractFeature.hpp"`

circle.cpp のインクルード依存関係図



関数

- void [calc_3d_axes_of_circle](#) (double major_axis[3], double minor_axis[3], const double normal[3], const [CameraParam](#) *cp)

画像上の楕円の長軸・短軸に対応する 3次元単位ベクトルの算出

- void [reconstruct_ellipse2D_to_circle3D](#) (std::vector< const [ovgr::Features2D](#) * > &feature, const [ovgr::CorrespondingSet](#) &cs, const [CameraParam](#) *camParam[3], const unsigned char *edge[3], [Features3D](#) *scene, const [Parameters](#) ¶meters)

7.5.1 説明

3次元円特徴生成関連関数

日付:

\$Date:: 2011-10-21 14:49:57 +0900 #

7.5.2 関数

7.5.2.1 void calc_3d_axes_of_circle (double *major_axis*[3], double *minor_axis*[3], const double *normal*[3], const CameraParam * *cp*)

画像上の楕円の長軸・短軸に対応する3次元単位ベクトルの算出 カメラパラメータ

引数:

major_axis 長軸

minor_axis 短軸

normal 3次元円の法線

cp カメラパラメータ

7.5.2.2 void reconstruct_ellipse2D_to_circle3D (std::vector< const ovgr::Features2D * > & *feature*, const ovgr::CorrespondingSet & *cs*, const CameraParam * *camParam*[3], const unsigned char * *edge*[3], Features3D * *scene*, const Parameters & *parameters*)

7.6 circle.h

3次元円特徴生成関連関数

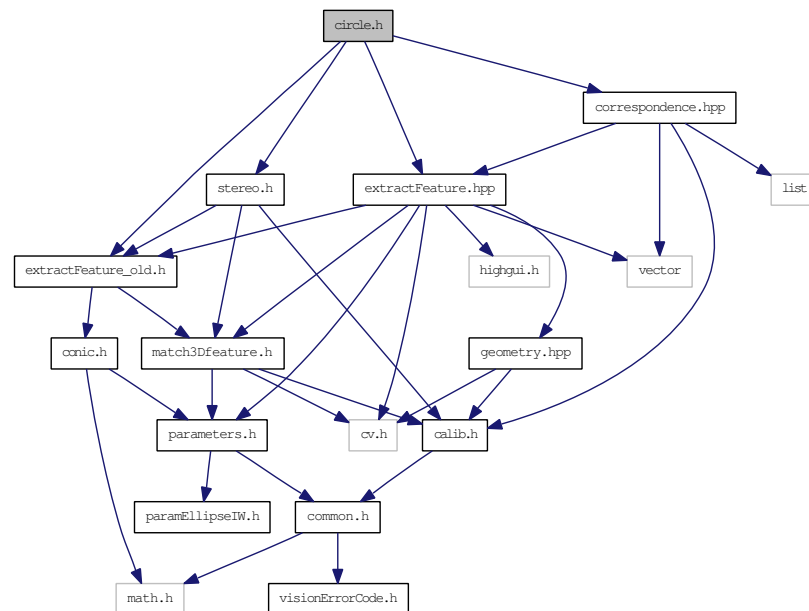
```
#include "extractFeature_old.h"
```

```
#include "stereo.h"
```

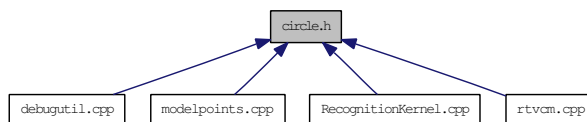
```
#include "extractFeature.hpp"
```

```
#include "correspondence.hpp"
```

circle.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- void [reconstruct_ellipse2D_to_circle3D](#) (std::vector< const [ovgr::Features2D](#) * > &feature, const [ovgr::CorrespondingSet](#) &cs, const [CameraParam](#)

- `*camParam[3], const unsigned char *edge[3], Features3D *scene, const Parameters ¶meters)`
- `void calc_3d_axes_of_circle (double major_axis[3], double minor_axis[3], const double normal[3], const CameraParam *cp)`
画像上の楕円の長軸・短軸に対応する 3 次元単位ベクトルの算出

7.6.1 説明

3 次元円特徴生成関連関数

日付:

\$Date:: 2011-09-15 17:59:25 +0900 # \$

7.6.2 関数

7.6.2.1 void calc_3d_axes_of_circle (double *major_axis*[3], double *minor_axis*[3], const double *normal*[3], const CameraParam * *cp*)

画像上の楕円の長軸・短軸に対応する 3 次元単位ベクトルの算出 カメラパラメータ

引数:

major_axis 長軸
minor_axis 短軸
normal 3 次元円の法線
cp カメラパラメータ

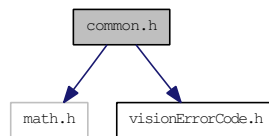
7.6.2.2 void reconstruct_ellipse2D_to_circle3D (std::vector< const ovgr::Features2D * > & *feature*, const ovgr::CorrespondingSet & *cs*, const CameraParam * *camParam*[3], const unsigned char * *edge*[3], Features3D * *scene*, const Parameters & *parameters*)

7.7 common.h

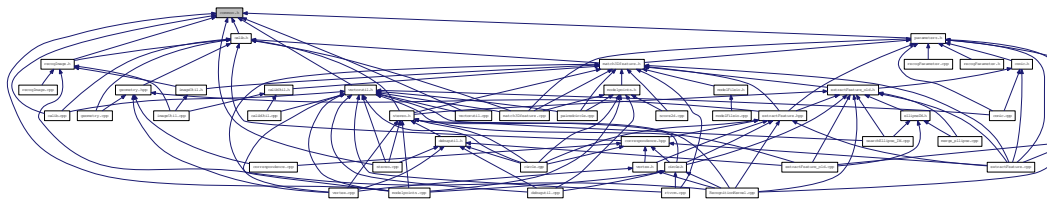
各種の共通定義 `#include <math.h>`

`#include "visionErrorCode.h"`

common.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [Data_2D](#)

マクロ定義

- `#define` [VISION_EPS](#) 1.0e-10
- `#define` [VISIBLE](#) 1
- `#define` [INVISIBLE](#) 0
- `#define` [NO_SEARCH_VERTEX](#) 1
- `#define` [NO_SEARCH_ELLIPSE](#) 2

列挙型

- enum [StereoPairing](#) {
[DBL_LR](#), [DBL_LV](#), [DBL_RV](#), [TBL_OR](#),
[TBL_AND](#) }

7.7.1 説明

各種の共通定義

日付:

\$Date:: 2011-09-30 18:33:01 +0900 #

7.7.2 マクロ定義

7.7.2.1 `#define INVISIBLE 0`

7.7.2.2 `#define NO_SEARCH_ELLIPSE 2`

7.7.2.3 `#define NO_SEARCH_VERTEX 1`

7.7.2.4 `#define VISIBLE 1`

7.7.2.5 `#define VISION_EPS 1.0e-10`

7.7.3 列挙型

7.7.3.1 `enum StereoPairing`

列挙型の値:

DBL_LR

DBL_LV

DBL_RV

TBL_OR

TBL_AND

7.8 conic.cpp

二次曲線特徴抽出関連関数 `#include <math.h>`

`#include <string.h>`

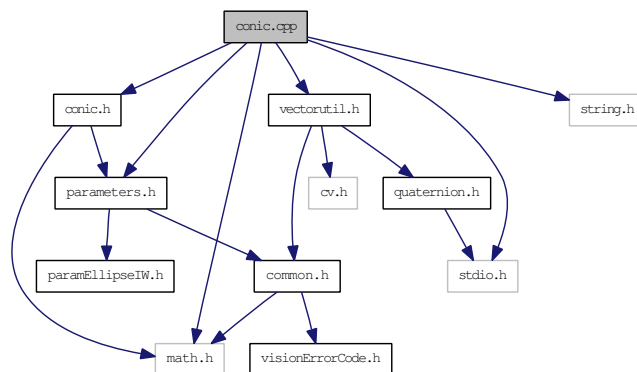
`#include <stdio.h>`

`#include "vectorutil.h"`

`#include "parameters.h"`

`#include "conic.h"`

conic.cpp のインクルード依存関係図



関数

- void [clearConicSum](#) (double sum[5][5])
- void [addConicSum](#) (double sum[5][5], int *point, double *offset)
- void [subConicSum](#) (double sum[5][5], int *point, double *offset)
- double [distanceConic](#) (double coef[6], int *point)
- [ConicType](#) [getConicType](#) (double coef[6])
- void [getConicProperty](#) (double coef[6], [ConicType](#) *type, double center[2], double axis[2][2], double *Laxis, double *Saxis)
- int [fitConic](#) (double sum[5][5], double coef[3][6], double *offset)
- [ConicType](#) [fitConicAny](#) (double retcoef[6], double *retError, double sum[5][5], int *point, const int nPoint, const int start, const int end, [Parameters](#) parameters, int line_detect_flag, double *offset)

7.8.1 説明

二次曲線特徴抽出関連関数

日付:

\$Date:: 2011-12-16 18:15:07 +0900 #

7.8.2 関数

7.8.2.1 void addConicSum (double *sum*[5][5], int * *point*, double * *offset*)

7.8.2.2 void clearConicSum (double *sum*[5][5])

7.8.2.3 double distanceConic (double *coef*[6], int * *point*)

7.8.2.4 int fitConic (double *sum*[5][5], double *coef*[3][6], double * *offset*)

7.8.2.5 ConicType fitConicAny (double *retcoef*[6], double * *retError*, double *sum*[5][5], int * *point*, const int *nPoint*, const int *start*, const int *end*, Parameters *parameters*, int *line_detect_flag*, double * *offset*)

7.8.2.6 void getConicProperty (double *coef*[6], ConicType * *type*, double *center*[2], double *axis*[2][2], double * *Laxis*, double * *Saxis*)

7.8.2.7 ConicType getConicType (double *coef*[6])

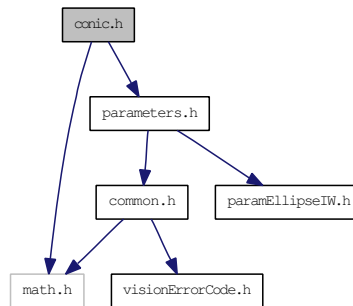
7.8.2.8 void subConicSum (double *sum*[5][5], int * *point*, double * *offset*)

7.9 conic.h

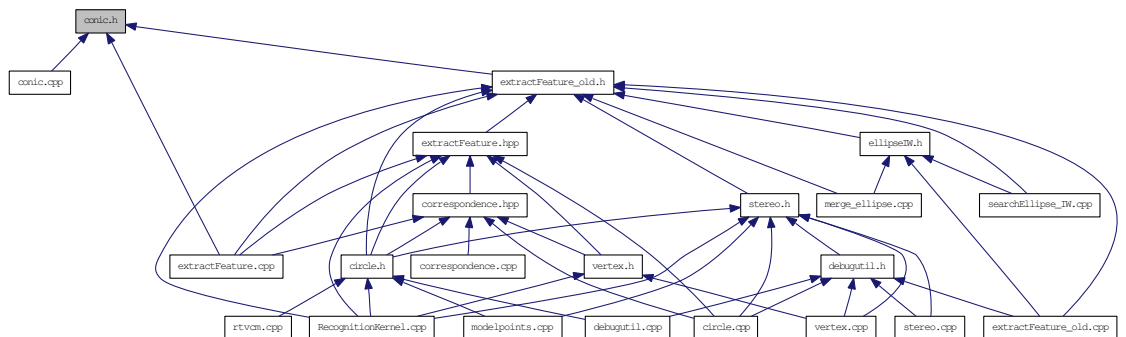
二次曲線特徴抽出関連関数 `#include <math.h>`

`#include "parameters.h"`

conic.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



列挙型

- enum `ConicType` {
`ConicType_Unknown`, `ConicType_Line`, `ConicType_Ellipse`,
`ConicType_Hyperbola`,
`ConicType_Parabola` }

関数

- void `clearConicSum` (double sum[5][5])

- void [addConicSum](#) (double sum[5][5], int *point, double *offset)
- void [subConicSum](#) (double sum[5][5], int *point, double *offset)
- double [distanceConic](#) (double coef[6], int *point)
- [ConicType](#) [getConicType](#) (double coef[6])
- void [getConicProperty](#) (double coef[6], [ConicType](#) *type, double center[2], double axis[2][2], double *Laxis, double *Saxis)
- int [fitConic](#) (double sum[5][5], double coef[3][6], double *offset)
- [ConicType](#) [fitConicAny](#) (double retcoef[6], double *retError, double sum[5][5], int *point, const int nPoint, const int start, const int end, [Parameters](#) parameters, int line_detect_flag, double *offset)

7.9.1 説明

二次曲線特徴抽出関連関数

日付:

\$Date:: 2011-09-09 14:00:23 +0900 #

7.9.2 列挙型

7.9.2.1 enum ConicType

列挙型の値:

ConicType_Unknown

ConicType_Line

ConicType_Ellipse

ConicType_Hyperbola

ConicType_Parabola

7.9.3 関数

7.9.3.1 void addConicSum (double sum[5][5], int * point, double * offset)

7.9.3.2 void clearConicSum (double sum[5][5])

7.9.3.3 `double distanceConic (double coef[6], int * point)`

7.9.3.4 `int fitConic (double sum[5][5], double coef[3][6], double * offset)`

7.9.3.5 `ConicType fitConicAny (double retcoef[6], double * retError, double sum[5][5], int * point, const int nPoint, const int start, const int end, Parameters parameters, int line_detect_flag, double * offset)`

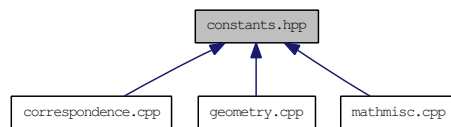
7.9.3.6 `void getConicProperty (double coef[6], ConicType * type, double center[2], double axis[2][2], double * Laxis, double * Saxis)`

7.9.3.7 `ConicType getConicType (double coef[6])`

7.9.3.8 `void subConicSum (double sum[5][5], int * point, double * offset)`

7.10 constants.hpp

このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



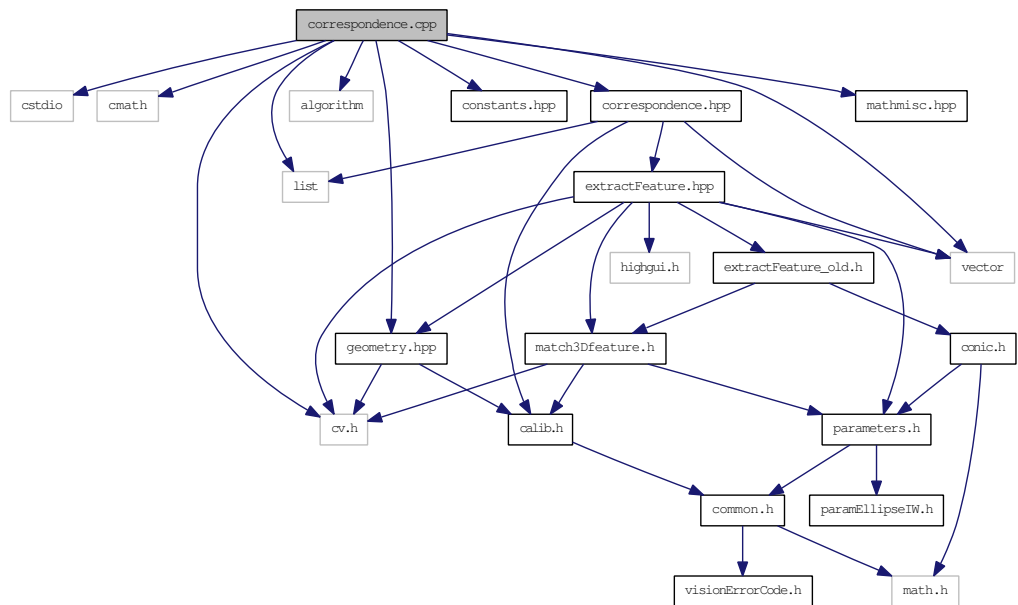
ネームスペース

- namespace [ovgr](#)

7.11 correspondence.cpp

```
#include <cstdio>
#include <cmath>
#include <vector>
#include <list>
#include <algorithm>
#include <cv.h>
#include "constants.hpp"
#include "correspondence.hpp"
#include "geometry.hpp"
#include "mathmisc.hpp"
```

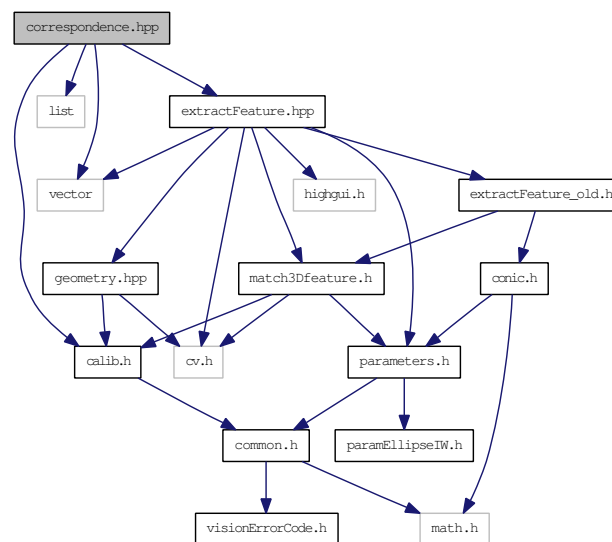
correspondence.cpp のインクルード依存関係図



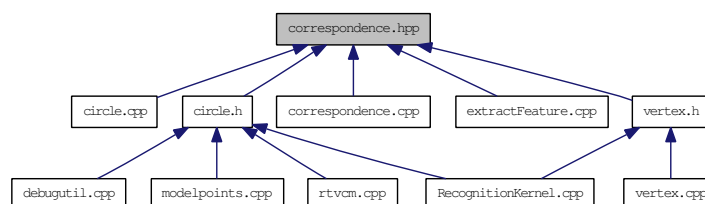
7.12 correspondence.hpp

```
#include <vector>
#include <list>
#include "calib.h"
#include "extractFeature.hpp"
```

correspondence.hpp のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [ovgr::CorrespondingPair](#)
- struct [ovgr::CorrespondingSet](#)
- struct [ovgr::CorrespondenceThresholds](#)

ネームスペース

- namespace [ovgr](#)

型定義

- typedef std::list< int > [ovgr::feature_index_list_t](#)
- typedef std::vector< [feature_index_list_t](#) > [ovgr::feature_map_t](#)
- typedef std::vector< int > [ovgr::feature_indexes_t](#)
- typedef std::list< [feature_indexes_t](#) > [ovgr::feature_list_t](#)

列挙型

- enum [ovgr::CorrespondingCriteria](#) { [ovgr::CorresOr](#), [ovgr::CorresAnd](#) }

関数

- [CorrespondingPair](#) [ovgr::make_corresponding_pairs](#) (const [Features2D](#) &feature1, const [CameraParam](#) ¶m1, const [Features2D](#) &feature2, const [CameraParam](#) ¶m2, const [CorrespondenceThresholds](#) &thres=[CorrespondenceThresholds](#)())
- [CorrespondingSet](#) [ovgr::filter_corresponding_set](#) (const std::vector< const [CorrespondingPair](#) * > &corres_pair, const [CorrespondingCriteria](#) criteria=[CorresOr](#))

7.13 debugutil.cpp

デバッグ用関数 `#include <stdio.h>`

`#include <highgui.h>`

`#include "debugutil.h"`

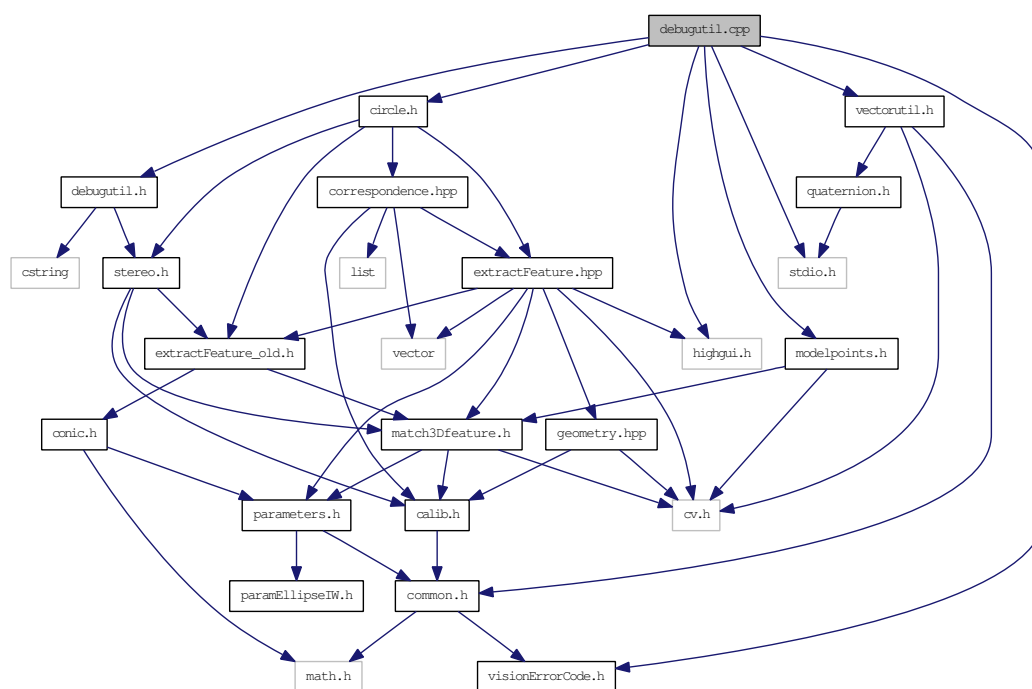
`#include "vectorutil.h"`

`#include "visionErrorCode.h"`

`#include "modelpoints.h"`

`#include "circle.h"`

debugutil.cpp のインクルード依存関係図



関数

- `int drawInputImage (const uchar *src, const Parameters ¶meters, const int id)`
- `int drawEdgeImage (const uchar *edge, const Parameters ¶meters, const int id)`

- int `drawDetectedLines` (const uchar *edge, const `Features2D_old` *lineFeatures, const `Parameters` ¶meters, const int id)
- int `drawDetectedVertices` (const `Features2D_old` *features, const `Parameters` ¶meters, const int id)
- int `drawTrackPoints` (const `Features2D_old` *features, const `Parameters` ¶meters, const int id)
- int `drawDetectedEllipses` (const uchar *edge, const `Features2D_old` *features, const `Parameters` ¶meters, const int id)
- int `drawCircleCandidate` (const uchar *edge, const std::vector< `CircleCandidate` > &candidates, int pairing, const `Parameters` ¶meters, const `CameraParam` *cameraParam)
- int `printVertex` (const std::vector< `Vertex` > &vertex)

7.13.1 説明

デバッグ用関数

日付:

\$Date:: 2011-10-25 19:47:47 +0900 # \$

7.13.2 関数

7.13.2.1 int `drawCircleCandidate` (const uchar * *edge*, const std::vector< `CircleCandidate` > & *candidates*, int *pairing*, const `Parameters` & *parameters*, const `CameraParam` * *cameraParam*)

7.13.2.2 int `drawDetectedEllipses` (const uchar * *edge*, const `Features2D_old` * *features*, const `Parameters` & *parameters*, const int *id*)

7.13.2.3 int `drawDetectedLines` (const uchar * *edge*, const `Features2D_old` * *lineFeatures*, const `Parameters` & *parameters*, const int *id*)

7.13.2.4 int `drawDetectedVertices` (const `Features2D_old` * *features*, const `Parameters` & *parameters*, const int *id*)

7.13.2.5 `int drawEdgeImage (const uchar * edge, const Parameters & parameters, const int id)`

7.13.2.6 `int drawInputImage (const uchar * src, const Parameters & parameters, const int id)`

7.13.2.7 `int drawTrackPoints (const Features2D_old * features, const Parameters & parameters, const int id)`

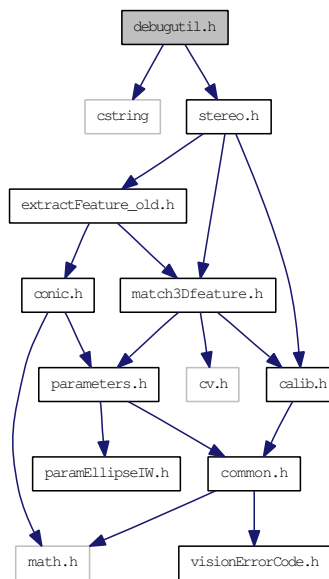
7.13.2.8 `int printVertex (const std::vector< ::Vertex > & vertex)`

7.14 debugutil.h

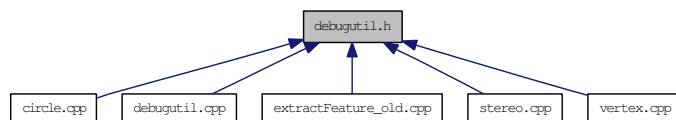
デバッグ用関数 `#include <cstring>`

`#include "stereo.h"`

debugutil.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct `ovgr::EqualOp< T >`
- class `ovgr::VariableWatcher< T, Equal >`

ネームスペース

- namespace `ovgr`

関数

- int [drawInputImage](#) (const uchar *src, const [Parameters](#) ¶meters, const int id)
- int [drawEdgeImage](#) (const uchar *edge, const [Parameters](#) ¶meters, const int id)
- int [drawDetectedLines](#) (const uchar *edge, const [Features2D_old](#) *lineFeatures, const [Parameters](#) ¶meters, const int id)
- int [drawDetectedVertices](#) (const [Features2D_old](#) *features, const [Parameters](#) ¶meters, const int id)
- int [drawDetectedEllipses](#) (const uchar *edge, const [Features2D_old](#) *features, const [Parameters](#) ¶meters, const int id)
- int [drawCircleCandidate](#) (const uchar *edge, const std::vector< [CircleCandidate](#) > &candidates, int pairing, const [Parameters](#) ¶meters, const [CameraParam](#) *cameraParam)
- int [printVertex](#) (const std::vector< [Vertex](#) > &vertex)

7.14.1 説明

デバッグ用関数

日付:

\$Date:: 2011-10-24 14:02:38 +0900 #

7.14.2 関数

7.14.2.1 int [drawCircleCandidate](#) (const uchar * *edge*, const std::vector< [CircleCandidate](#) > & *candidates*, int *pairing*, const [Parameters](#) & *parameters*, const [CameraParam](#) * *cameraParam*)

7.14.2.2 int [drawDetectedEllipses](#) (const uchar * *edge*, const [Features2D_old](#) * *features*, const [Parameters](#) & *parameters*, const int *id*)

7.14.2.3 int [drawDetectedLines](#) (const uchar * *edge*, const [Features2D_old](#) * *lineFeatures*, const [Parameters](#) & *parameters*, const int *id*)

7.14.2.4 `int drawDetectedVertices (const Features2D_old * features, const Parameters & parameters, const int id)`

7.14.2.5 `int drawEdgeImage (const uchar * edge, const Parameters & parameters, const int id)`

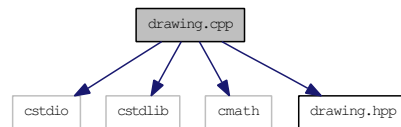
7.14.2.6 `int drawInputImage (const uchar * src, const Parameters & parameters, const int id)`

7.14.2.7 `int printVertex (const std::vector< ::Vertex > & vertex)`

7.15 drawing.cpp

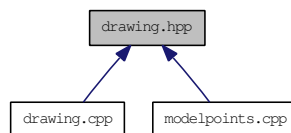
```
#include <cstdio>
#include <cstdlib>
#include <cmath>
#include "drawing.hpp"
```

drawing.cpp のインクルード依存関係図



7.16 drawing.hpp

このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- class `ovgr::PointsOnLine`
- class `ovgr::PointsOnEllipse`

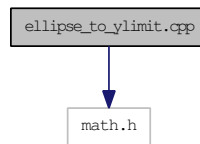
ネームスペース

- namespace `ovgr`

7.17 ellipse_to_ylimit.cpp

```
#include <math.h>
```

ellipse_to_ylimit.cpp のインクルード依存関係図



関数

- int `ellipse_to_ylimit` (const double `a`[6], double `ylimit`[2])

7.17.1 関数

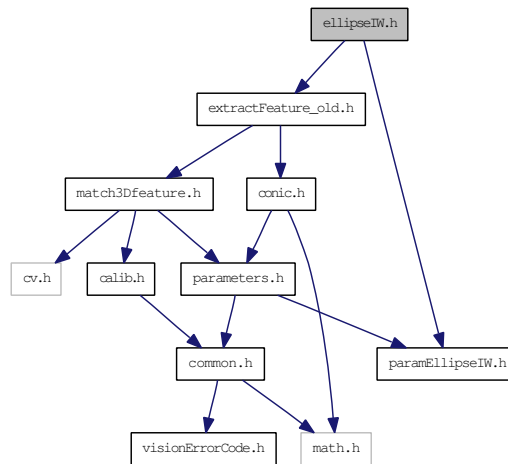
7.17.1.1 int `ellipse_to_ylimit` (const double `a`[6], double `ylimit`[2])

7.18 ellipseIW.h

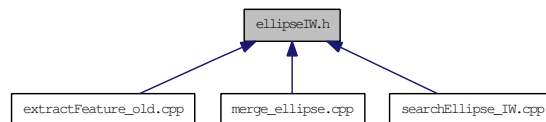
```
#include "extractFeature_old.h"
```

```
#include "paramEllipseIW.h"
```

ellipseIW.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [_Ellipse_](#)
- struct [_SumSet_](#)
- struct [_offset_prop_](#)

マクロ定義

- #define [NDIM3](#) (3)
- #define [NDIM2](#) (2)
- #define [NAXIS](#) (2)

- #define [NCOEFCUBIC](#) (4)
- #define [NDIM_CONIC_HALF](#) (3)
- #define [NDIM_CONIC_FULL](#) (6)
- #define [SEARCH_FEATURES2_OK](#) (1)
- #define [SEARCH_FEATURES2_NG](#) (0)
- #define [CHECK_ELLIPSE_NG](#) (0)
- #define [CHECK_ELLIPSE_OK](#) (1)
- #define [MERGE_ELLIPSE_OK](#) (0)
- #define [MERGE_ELLIPSE_NG](#) (1)

型定義

- typedef struct [_Ellipse](#) [Ellipse](#)
- typedef struct [_SumSet](#) [SumSet](#)
- typedef struct [_offset_prop](#) [OffsetProp](#)

関数

- void [addArcSum](#) ([SumSet](#) *sum, const int *pointX, const double *offsetD)
- int [searchEllipseIW](#) ([Features2D_old](#) *f2D, int iTrack, const [ParamEllipseIW](#) *paramE)
- void [sum_to_P_dynamic](#) (const [SumSet](#) *sum, [Ellipse](#) *ellipse, [OffsetProp](#) *offsetProp)
- void [P_to_avec_and_fix](#) ([Ellipse](#) *ellipse, const [ParamEllipseIW](#) *paramE)
- void [avec_to_ellipse](#) (int k_min_error, [Ellipse](#) *ellipse)
- int [check_ellipse_cond](#) ([Ellipse](#) *ellipse, const [ParamEllipseIW](#) *paramE)
- int [mod_nPoint](#) (int n, int nPoint)
- double [distanceAConic](#) (const double coef[6], const int *point)
- int [merge_ellipse](#) ([Features2D_old](#) *f2D, const [ParamEllipseIW](#) *paramE)

7.18.1 マクロ定義

7.18.1.1 #define [CHECK_ELLIPSE_NG](#) (0)

7.18.1.2 #define [CHECK_ELLIPSE_OK](#) (1)

7.18.1.3 #define [MERGE_ELLIPSE_NG](#) (1)

7.18.1.4 #define MERGE_ELLIPSE_OK (0)

7.18.1.5 #define NAXIS (2)

7.18.1.6 #define NCOEFCUBIC (4)

7.18.1.7 #define NDIM2 (2)

7.18.1.8 #define NDIM3 (3)

7.18.1.9 #define NDIM_CONIC_FULL (6)

7.18.1.10 #define NDIM_CONIC_HALF (3)

7.18.1.11 #define SEARCH_FEATURES2_NG (0)

7.18.1.12 #define SEARCH_FEATURES2_OK (1)

7.18.2 型定義

7.18.2.1 typedef struct _Ellipse_ Ellipse

7.18.2.2 `typedef struct _offset_prop_ OffsetProp`

7.18.2.3 `typedef struct _SumSet_ SumSet`

7.18.3 関数

7.18.3.1 `void addArcSum (SumSet * sum, const int * pointX, const double * offsetD)`

7.18.3.2 `void avec_to_ellipse (int k_min_error, Ellipse * ellipse)`

7.18.3.3 `int check_ellipse_cond (Ellipse * ellipse, const ParamEllipseIW * paramE)`

7.18.3.4 `double distanceAConic (const double coef[6], const int * point)`

7.18.3.5 `int merge_ellipse (Features2D_old * f2D, const ParamEllipseIW * paramE)`

7.18.3.6 `int mod_nPoint (int n, int nPoint)`

7.18.3.7 `void P_to_avec_and_fix (Ellipse * ellipse, const ParamEllipseIW * paramE)`

7.18.3.8 `int searchEllipseIW (Features2D_old * f2D, int iTrack, const ParamEllipseIW * paramE)`

7.18.3.9 `void sum_to_P_dynamic (const SumSet * sum, Ellipse * ellipse, OffsetProp * offsetProp)`

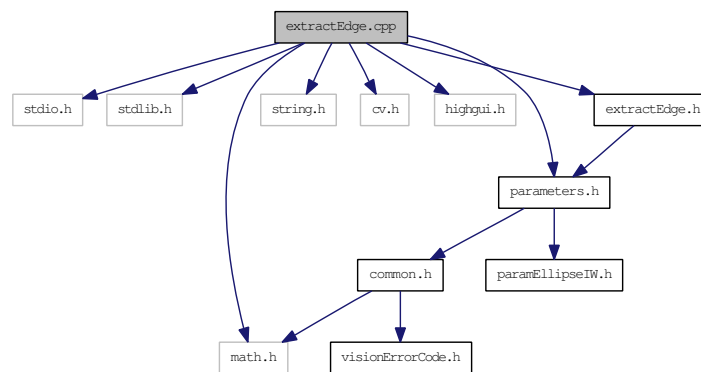
7.19 extractEdge.cpp

```

エッジ抽出関連関数#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <cv.h>
#include <highgui.h>
#include "parameters.h"
#include "extractEdge.h"

```

extractEdge.cpp のインクルード依存関係図



マクロ定義

- #define [Gray](#)(col, row) (gray[(row)*colsize+(col)])
- #define [Edge](#)(col, row) (edge[(row)*colsize+(col)])

関数

- int [extractEdge](#) (unsigned char *edge, unsigned char *gray, const int threshold, [Parameters](#) parameters)
- void [extractEdge_new](#) (unsigned char *edge, unsigned char *gray, const int threshold, [Parameters](#) parameters)

7.19.1 説明

エッジ抽出関連関数

日付:

\$Date:: 2011-10-21 14:49:57 +0900 #

7.19.2 マクロ定義

7.19.2.1 `#define Edge(col, row) (edge[(row)*colsize+(col)])`

7.19.2.2 `#define Gray(col, row) (gray[(row)*colsize+(col)])`

7.19.3 関数

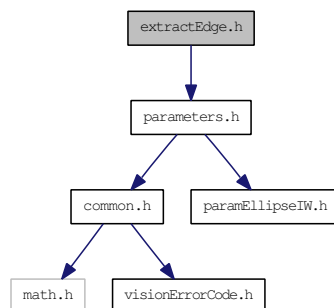
7.19.3.1 `int extractEdge (unsigned char * edge, unsigned char * gray, const int threshold, Parameters parameters)`

7.19.3.2 `void extractEdge_new (unsigned char * edge, unsigned char * gray, const int threshold, Parameters parameters)`

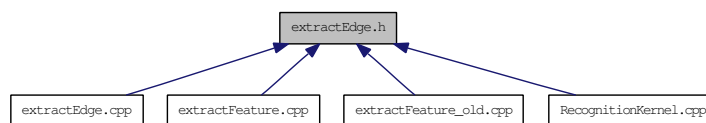
7.20 extractEdge.h

エッジ抽出関連関数 `#include "parameters.h"`

extractEdge.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



マクロ定義

- `#define EEnotEdge (0)`
- `#define EEerasedThin (1)`
- `#define EEcandidate (2)`
- `#define EEnotSearched (2)`
- `#define EEsearchedSmall (3)`
- `#define EEsearchedLarge (4)`
- `#define EEextended (5)`
- `#define EE6 (6)`
- `#define EE7 (7)`

関数

- `int extractEdge (unsigned char *edge, unsigned char *gray, const int threshold, Parameters parameters)`
- `void extractEdge_new (unsigned char *edge, unsigned char *gray, const int threshold, Parameters parameters)`

7.20.1 説明

エッジ抽出関連関数

日付:

\$Date:: 2011-09-09 14:00:23 +0900 #

7.20.2 マクロ定義

7.20.2.1 `#define EE6 (6)`

7.20.2.2 `#define EE7 (7)`

7.20.2.3 `#define EEcandidate (2)`

7.20.2.4 `#define EEerasedThin (1)`

7.20.2.5 `#define EEextended (5)`

7.20.2.6 `#define EEnotEdge (0)`

7.20.2.7 `#define EEnotSearched (2)`

7.20.2.8 `#define EEsearchedLarge (4)`

7.20.2.9 `#define EEsearchedSmall (3)`

7.20.3 関数

7.20.3.1 `int extractEdge (unsigned char * edge, unsigned char * gray, const int threshold, Parameters parameters)`

7.20.3.2 `void extractEdge_new (unsigned char * edge, unsigned char * gray, const int threshold, Parameters parameters)`

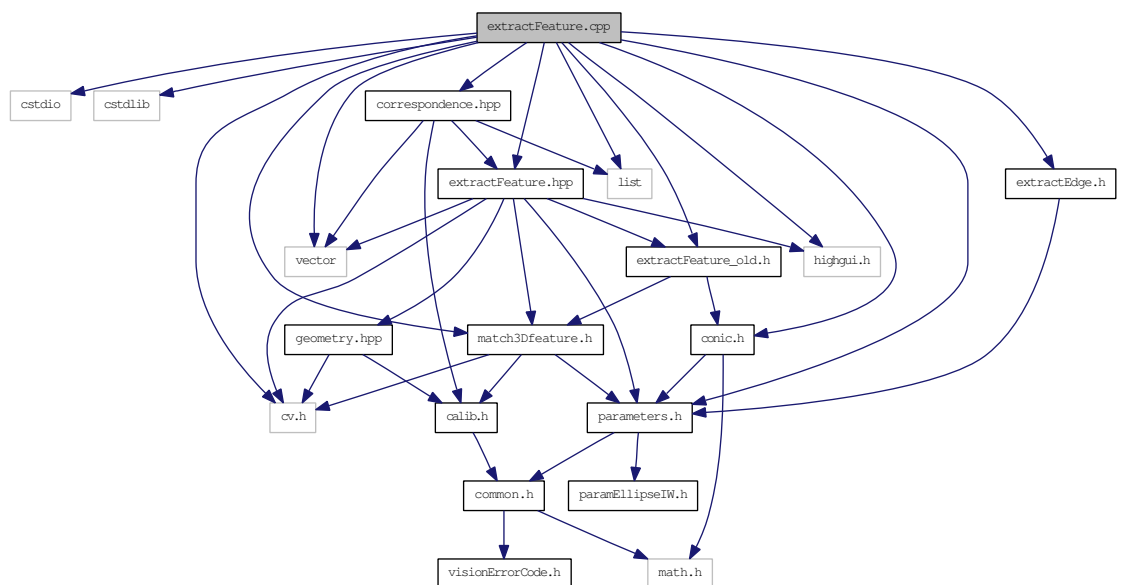
7.21 extractFeature.cpp

```

2次元特徴抽出関連関数
#include <cstdio>
#include <cstdlib>
#include <vector>
#include <list>
#include <cv.h>
#include <highgui.h>
#include "parameters.h"
#include "match3Dfeature.h"
#include "conic.h"
#include "extractFeature_old.h"
#include "extractEdge.h"
#include "extractFeature.hpp"
#include "correspondence.hpp"

```

extractFeature.cpp のインクルード依存関係図



型定義

- typedef std::vector< cv::Point > [Points](#)
- typedef std::vector< [Points](#) > [PointSet](#)
- typedef std::vector< bool > [support_index_t](#)

7.21.1 説明

2次元特徴抽出関連関数

日付:

\$Date:: 2011-12-16 18:15:07 +0900 #

7.21.2 型定義

7.21.2.1 typedef std::vector<cv::Point> **Points**

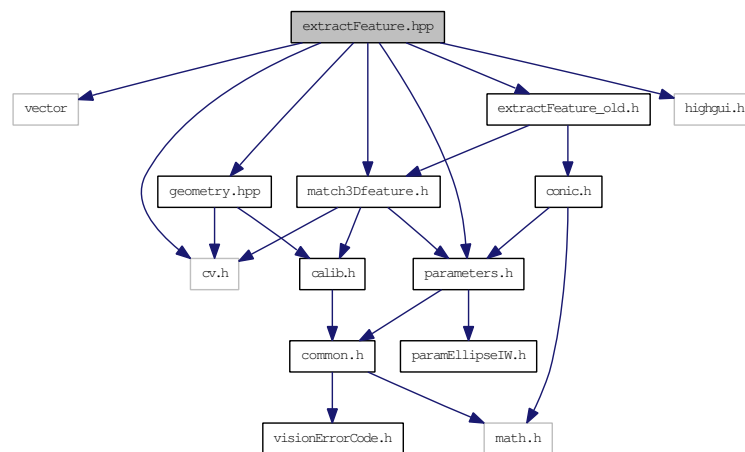
7.21.2.2 typedef std::vector<Points> **PointSet**

7.21.2.3 typedef std::vector<bool> **support_index_t**

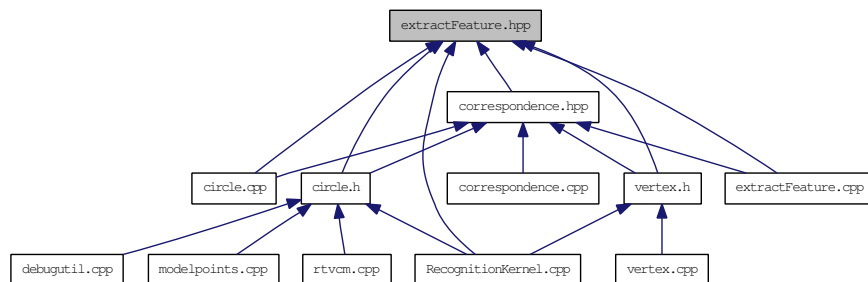
7.22 extractFeature.hpp

```
#include <vector>
#include <cv.h>
#include <highgui.h>
#include "parameters.h"
#include "match3Dfeature.h"
#include "extractFeature_old.h"
#include "geometry.hpp"
```

extractFeature.hpp のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [ovgr::Feature2D](#)
- struct [ovgr::LineFeature2D](#)
- struct [ovgr::ConicFeature2D](#)
- struct [ovgr::VertexFeature](#)
- struct [ovgr::EllipseFeature](#)
- struct [ovgr::Features2D](#)

ネームスペース

- namespace [ovgr](#)

型定義

- typedef std::vector< Point2D > [ovgr::Segment2D](#)

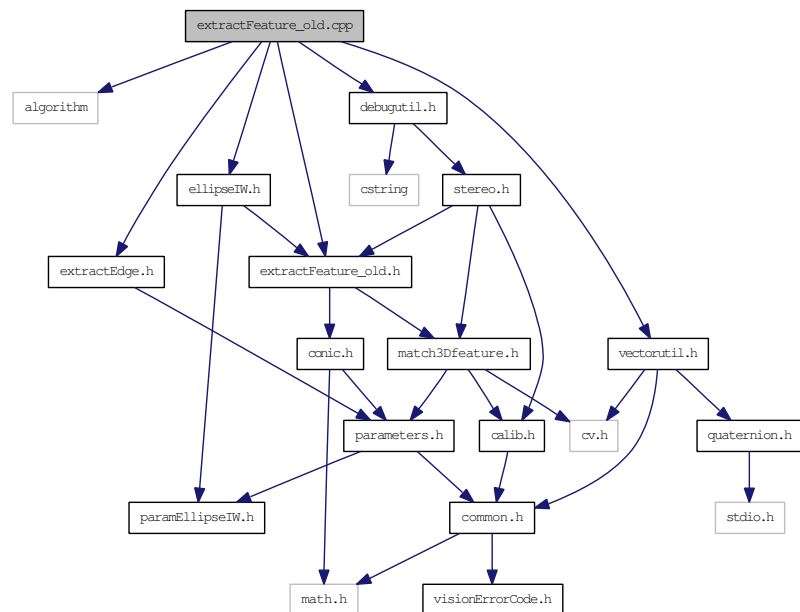
関数

- [Features2D ovgr::extractFeatures](#) (const unsigned char *edge, const [Parameters](#) ¶meters, const [Features3D](#) &model)
- [Features2D ovgr::ImageToFeature2D](#) (unsigned char *src, unsigned char *edge, const [Parameters](#) ¶meters, const [Features3D](#) &model)
- [Features2D_old * ovgr::create_old_features_from_new_one](#) (const [Features2D](#) &features)
- [Features2D ovgr::create_new_features_from_old_one](#) (const [Features2D_old](#) *old_features, unsigned char *img=NULL, const [Parameters](#) *parameters=NULL)
- cv::RotatedRect [ovgr::calc_ellipse_rr](#) (const double coeff[6])
- template<class VF >
void [ovgr::draw_VertexFeatures](#) (cv::Mat &img, const VF &vf)
- template<class EF >
void [ovgr::draw_EllipseFeatures](#) (cv::Mat &img, const EF &ef)

7.23 extractFeature_old.cpp

```
#include <algorithm>
#include "extractEdge.h"
#include "extractFeature_old.h"
#include "vectorutil.h"
#include "debugutil.h"
#include "ellipseIW.h"
```

extractFeature_old.cpp のインクルード依存関係図



マクロ定義

- #define [Work](#)(col, row) (work[(row)*colsize+(col)])
- #define [CONIC_MATCH_OK](#) (0)
- #define [CONIC_MATCH_NG](#) (1)

型定義

- typedef struct _feature_list_ [Feature_List](#)

関数

- void `destructFeatures` (`Features2D_old` *features)
2次元特徴情報のメモリ解放
- `Features2D_old` * `expandFeatures` (`Features2D_old` *features)
2次元特徴データのメモリ拡張
- void `mark_similar_lines` (`Features2D_old` *lineFeatures, const double tolerance)
- `Features2D_old` * `extractFeatures_old` (unsigned char *edge, `Parameters` parameters, const int id, `Features3D` model)
- `Features2D_old` * `ImageToFeature2D_old` (unsigned char *src, unsigned char *edge, `Parameters` parameters, const int id, `Features3D` model)
ステレオ画像の一枚から二次元特徴の抽出

7.23.1 マクロ定義

7.23.1.1 `#define CONIC_MATCH_NG (1)`

7.23.1.2 `#define CONIC_MATCH_OK (0)`

7.23.1.3 `#define Work(col, row) (work[(row)*colsize+(col)])`

7.23.2 型定義

7.23.2.1 `typedef struct _feature_list_ Feature_List`

7.23.3 関数

7.23.3.1 void `destructFeatures` (`Features2D_old` **features*)

2次元特徴情報のメモリ解放

7.23.3.2 Features2D_old* expandFeatures (Features2D_old * *features*)

2次元特徴データのメモリ拡張

7.23.3.3 Features2D_old* extractFeatures_old (unsigned char * *edge*, Parameters *parameters*, const int *id*, Features3D *model*)**7.23.3.4 Features2D_old* ImageToFeature2D_old (unsigned char * *src*, unsigned char * *edge*, Parameters *parameters*, const int *id*, Features3D *model*)**

ステレオ画像の一枚から二次元特徴の抽出

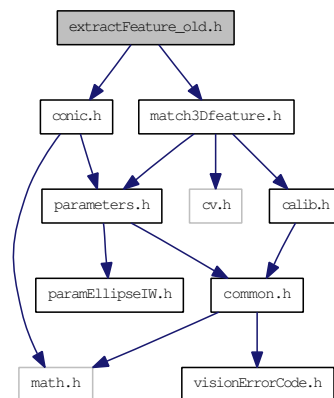
7.23.3.5 void mark_similar_lines (Features2D_old * *lineFeatures*, const double *tolerance*)

7.24 extractFeature_old.h

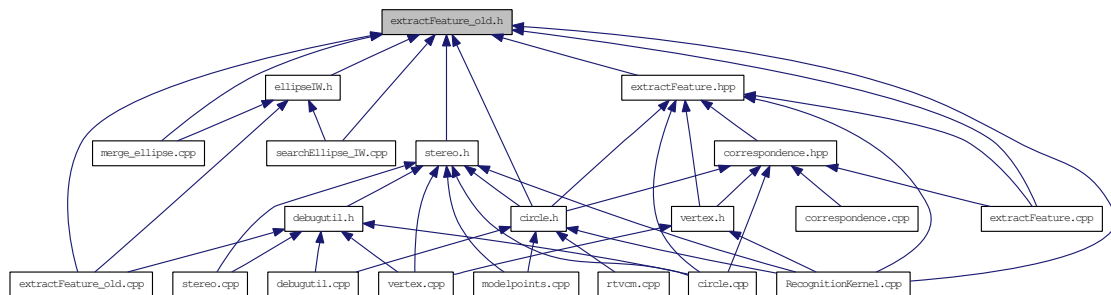
2次元特徴抽出関連関数 `#include "conic.h"`

`#include "match3Dfeature.h"`

extractFeature_old.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [_ellipse_arc_](#)
- struct [_ellipse_arc_list_](#)
- struct [Feature2D_old](#)
各 2 次元特徴
- struct [Track](#)

輪郭情報

- struct [Features2D_old](#)
2次元特徴情報
- struct [EllipseGroup](#)
楕円重複除去用グループ情報

型定義

- typedef struct [_ellipse_arc_](#) [EllipseArc](#)
- typedef struct [_ellipse_arc_list_](#) [EllipseArcList](#)

関数

- void [destructFeatures](#) ([Features2D_old](#) *features)
2次元特徴情報のメモリ解放
- [Features2D_old](#) * [expandFeatures](#) ([Features2D_old](#) *features)
2次元特徴データのメモリ拡張
- [Features2D_old](#) * [ImageToFeature2D_old](#) (unsigned char *src, unsigned char *edge, [Parameters](#) parameters, const int id, [Features3D](#) model)
ステレオ画像の一枚から二次元特徴の抽出

7.24.1 説明

2次元特徴抽出関連関数

日付:

\$Date:: 2011-12-12 16:32:35 +0900 # \$

7.24.2 型定義

7.24.2.1 typedef struct [_ellipse_arc_](#) [EllipseArc](#)

7.24.2.2 `typedef struct _ellipse_arc_list_ EllipseArcList`

7.24.3 関数

7.24.3.1 `void destructFeatures (Features2D_old * features)`

2次元特徴情報のメモリ解放

7.24.3.2 `Features2D_old* expandFeatures (Features2D_old * features)`

2次元特徴データのメモリ拡張

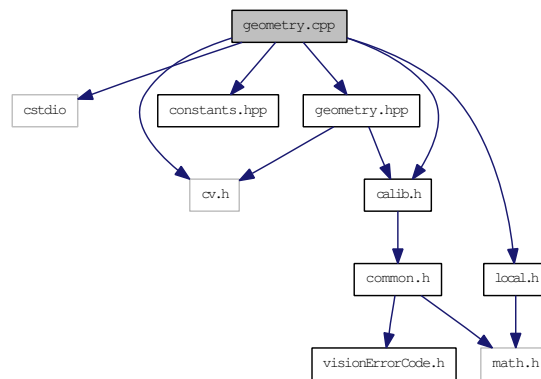
7.24.3.3 `Features2D_old* ImageToFeature2D_old (unsigned char * src,
unsigned char * edge, Parameters parameters, const int id,
Features3D model)`

ステレオ画像の一枚から二次元特徴の抽出

7.25 geometry.cpp

```
#include <cstdio>
#include <cv.h>
#include "constants.hpp"
#include "geometry.hpp"
#include "calib.h"
#include "local.h"
```

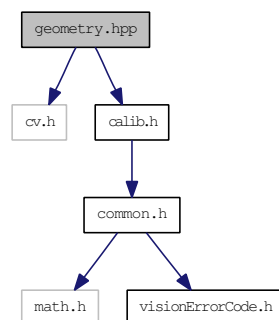
geometry.cpp のインクルード依存関係図



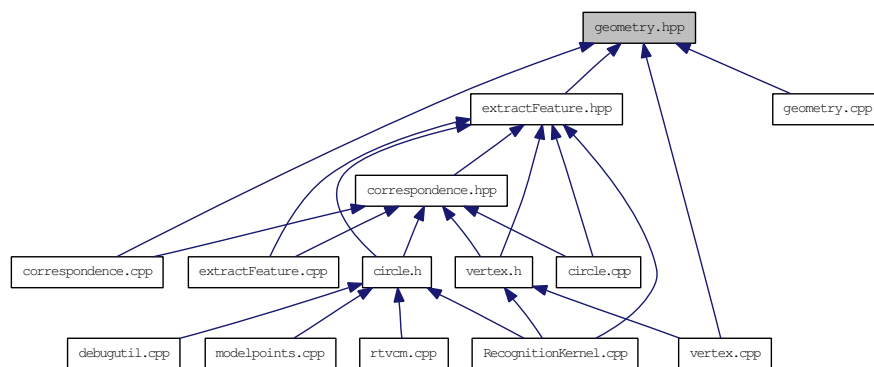
7.26 geometry.hpp

```
#include <cv.h>
#include "calib.h"
```

geometry.hpp のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct `ovgr::Array< T, N >`
- struct `ovgr::Point2D`

ネームスペース

- namespace `ovgr`

関数

- void `ovgr::calc_crossing_point` (double p[3], const double l1[3], const double l2[3])
- cv::Mat `ovgr::calc_homography` (const std::vector< Array< double, 3 > > &point1, const std::vector< Array< double, 3 > > &point2, const std::vector< Array< double, 3 > > &line1=std::vector< Array< double, 3 > >(), const std::vector< Array< double, 3 > > &line2=std::vector< Array< double, 3 > >())
- cv::Mat `ovgr::calc_essential_matrix` (const `CameraParam` &cp1, const `CameraParam` &cp2)
- cv::Mat `ovgr::calc_fundamental_matrix` (const `CameraParam` &cp1, const `CameraParam` &cp2)
- void `ovgr::calc_epipolar_line` (double line_coef[3], const cv::Mat &E, const Point2D &point, const bool inv=false)
- void `ovgr::calc_epipole` (double e[3], const `CameraParam` &cp1, const `CameraParam` &cp2)
- cv::Mat `ovgr::calc_infinite_homography` (const `CameraParam` &cp1, const `CameraParam` &cp2)
- double `ovgr::distance_from_line` (const double line_coef[3], const Point2D &point)
- void `ovgr::calc_line_joining_points` (double line_coef[3], const double p1[3], const double p2[3])

7.27 imageUtil.cpp

画像入出力関数 `#include <stdio.h>`

`#include <stdlib.h>`

`#include <math.h>`

`#include <string.h>`

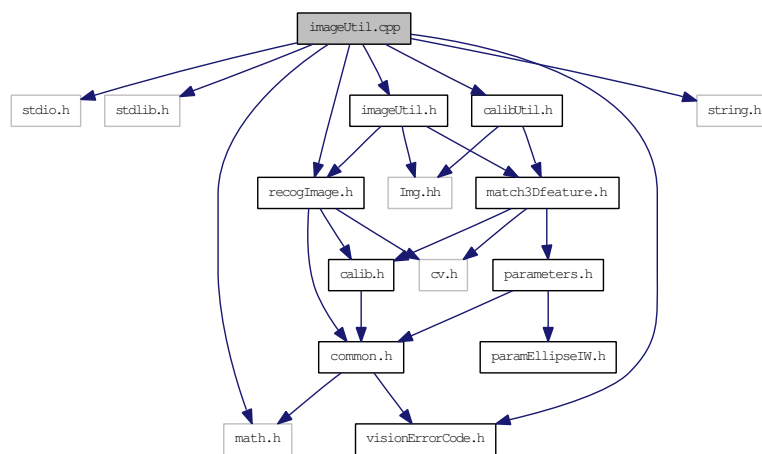
`#include "recogImage.h"`

`#include "imageUtil.h"`

`#include "calibUtil.h"`

`#include "visionErrorCode.h"`

imageUtil.cpp のインクルード依存関係図



関数

- `RecogImage ** convertTimedMultiCameraImageToRecogImage (const Img::TimedMultiCameraImage &frame, const bool color_mode)`
- `void freeConvertedRecogImage (RecogImage **recogImage, int imageNum)`

7.27.1 説明

画像入出力関数

日付:

\$Date:: 2011-09-29 08:56:58 +0900 # \$

7.27.2 関数

7.27.2.1 **RecogImage** convertTimedMultiCameraImageToRecogImage (const Img::TimedMultiCameraImage & *frame*, const bool *color_mode*)**

TimedMultiCameraImage 画像データを、RecogImage 構造体形式に変換する。
color_mode == false のときカラー画像をグレイ画像に変換する。

7.27.2.2 **void freeConvertedRecogImage (RecogImage ** *recogImage*, int *imageNum*)**

convertTimedMultiCameraImageToRecogImage によって 確保されたメモリを開放
する

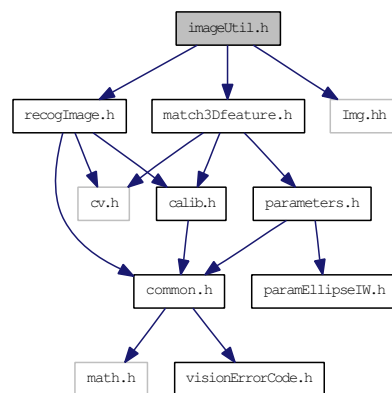
7.28 imageUtil.h

画像入出力関連 `#include "recogImage.h"`

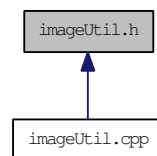
`#include "match3Dfeature.h"`

`#include "Img.hh"`

imageUtil.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- `RecogImage ** convertTimedMultiCameraImageToRecogImage (const Img::TimedMultiCameraImage &frame, const bool color_mode=false)`
- `void freeConvertedRecogImage (RecogImage **recogImage, int imageNum)`

7.28.1 説明

画像入出力関連

日付:

\$Date:: 2011-09-29 08:56:58 +0900 #

7.28.2 関数

7.28.2.1 **RecogImage** convertTimedMultiCameraImageToRecogImage (const Img::TimedMultiCameraImage & *frame*, const bool *color_mode*)**

TimedMultiCameraImage 画像データを、RecogImage 構造体形式に変換する。画像は 1 枚ごとに個別の RecogImage 構造体を作成する。

TimedMultiCameraImage 画像データを、RecogImage 構造体形式に変換する。
color_mode == false のときカラー画像をグレー画像に変換する。

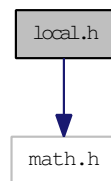
7.28.2.2 **void freeConvertedRecogImage (RecogImage ** *recogImage*, int *imageNum*)**

convertTimedMultiCameraImageToRecogImage によって確保されたメモリを開放する

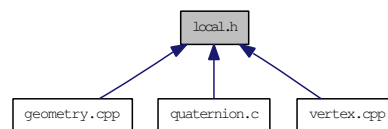
7.29 local.h

```
#include <math.h>
```

local.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。

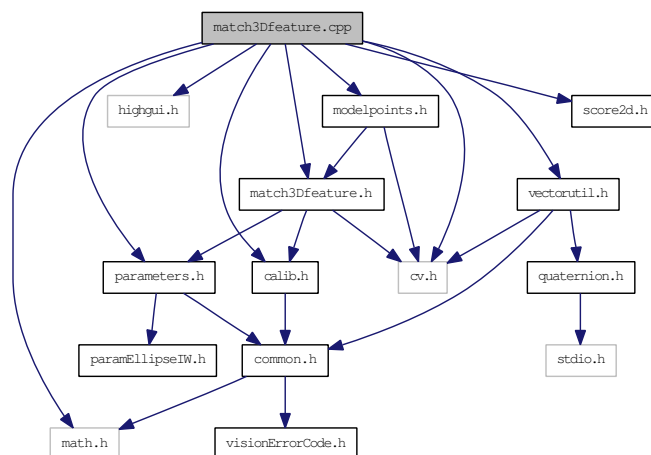


7.30 match3Dfeature.cpp

3次元特徴による認識関連関数 `#include <math.h>`

```
#include <cv.h>
#include <highgui.h>
#include "parameters.h"
#include "calib.h"
#include "vectorutil.h"
#include "modelpoints.h"
#include "score2d.h"
#include "match3Dfeature.h"
```

match3Dfeature.cpp のインクルード依存関係図



関数

- void `freeFeatures3D` (`Features3D *feature`)
3次元特徴データのメモリ解放
- void `freeMatch3Dresults` (`Match3Dresults *holder`)
認識結果データのメモリ解除
- void `shrinkMatch3Dresults` (`Match3Dresults *Match`)
不要になった認識結果データの開放

- [Match3Dresults](#) [matchFeatures3D](#) ([Features3D](#) &scene, [Features3D](#) &model, [Parameters](#) ¶meters)

7.30.1 説明

3次元特徴による認識関連関数

日付:

\$Date:: 2011-11-22 12:02:19 +0900 #

7.30.2 関数

7.30.2.1 void freeFeatures3D (Features3D * *feature*)

3次元特徴データのメモリ解放

7.30.2.2 void freeMatch3Dresults (Match3Dresults * *holder*)

認識結果データのメモリ解除

7.30.2.3 Match3Dresults matchFeatures3D (Features3D & *scene*, Features3D & *model*, Parameters & *parameters*)

認識：シーン特徴とモデル特徴の照合 戻り値：認識結果

7.30.2.4 void shrinkMatch3Dresults (Match3Dresults * *Match*)

不要になった認識結果データの開放

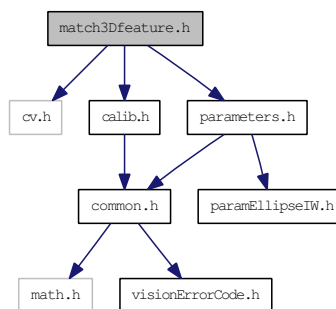
7.31 match3Dfeature.h

3次元特徴による認識関連関数 `#include <cv.h>`

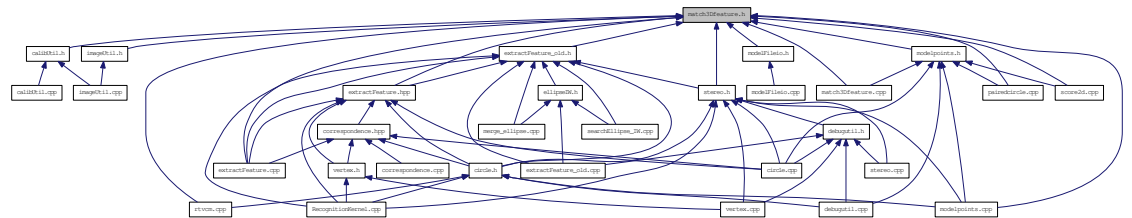
`#include "parameters.h"`

`#include "calib.h"`

match3Dfeature.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [Trace](#)
認識結果評価用サンプリング点列情報
- struct [P3D](#)
3次元位置情報
- struct [P2D](#)
2次元位置情報
- struct [tag_Wireframe](#)

ワイヤフレームモデル

- struct [tag_Wireframe::Segment](#)
 < 線分
- struct [tag_Wireframe::Face](#)
 < 面
- struct [Vertex](#)
 3次元頂点情報
- struct [Circle](#)
 3次元円情報
- struct [Features3D](#)
 3次元特徴情報
- struct [MatchResult](#)
 各認識結果情報
- struct [Match3Dresults](#)
 全認識結果

型定義

- typedef struct [tag_Wireframe](#) [Wireframe](#)
 ワイヤフレームモデル

列挙型

- enum [m3df_side](#) { [M3DF_FRONT](#) = 0, [M3DF_BACK](#) = 1 }
- 表裏を表す定数
- enum [m3df_feature_label](#) { [M3DF_LABEL_NONE](#) = 0, [M3DF_LABEL_NOEVAL](#) = 1 }
- 3次元特徴のラベル

関数

- void [freeMatch3Dresults](#) ([Match3Dresults](#) *holder)
 認識結果データのメモリ解除

- void `shrinkMatch3Dresults` (`Match3Dresults` *Match)
不要になった認識結果データの開放
- void `freeFeatures3D` (`Features3D` *feature)
3次元特徴データのメモリ解放
- `Match3Dresults matchFeatures3D` (`Features3D` &scene, `Features3D` &model, `Parameters` ¶meters)
- `Match3Dresults matchPairedCircles` (`Features3D` &scene, `Features3D` &model, const std::vector< cv::Mat > &dstImages, double tolerance, `StereoPairing` &pairing)

7.31.1 説明

3次元特徴による認識関連関数

日付:

\$Date:: 2011-11-22 12:02:19 +0900 # \$

7.31.2 型定義

7.31.2.1 typedef struct tag_Wireframe Wireframe

ワイヤフレームモデル

7.31.3 列挙型

7.31.3.1 enum m3df_feature_label

3次元特徴のラベル

列挙型の値:

M3DF_LABEL_NONE

M3DF_LABEL_NOEVAL

7.31.3.2 enum m3df_side

表裏を表す定数

列挙型の値:

M3DF_FRONT

M3DF_BACK

7.31.4 関数

7.31.4.1 void freeFeatures3D (Features3D * *feature*)

3次元特徴データのメモリ解放

7.31.4.2 void freeMatch3Dresults (Match3Dresults * *holder*)

認識結果データのメモリ解除

7.31.4.3 Match3Dresults matchFeatures3D (Features3D & *scene*, Features3D & *model*, Parameters & *parameters*)

認識: シーン特徴とモデル特徴の照合 戻り値: 認識結果

7.31.4.4 Match3Dresults matchPairedCircles (Features3D & *scene*, Features3D & *model*, const std::vector< cv::Mat > & *dstImages*, double *tolerance*, StereoPairing & *pairing*)

2円を使った照合 戻り値: 認識結果

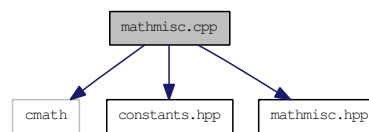
7.31.4.5 void shrinkMatch3Dresults (Match3Dresults * *Match*)

不要になった認識結果データの開放

7.32 mathmisc.cpp

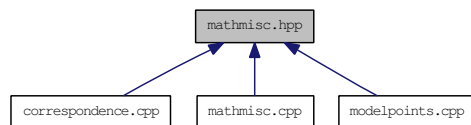
```
#include <cmath>
#include "constants.hpp"
#include "mathmisc.hpp"
```

mathmisc.cpp のインクルード依存関係図



7.33 mathmisc.hpp

このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



ネームスペース

- namespace `ovgr`

関数

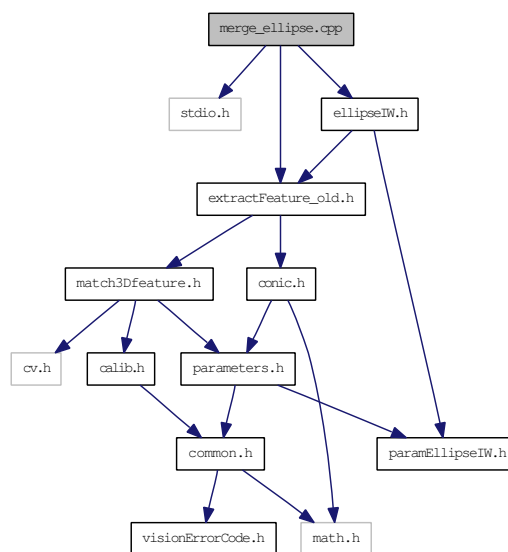
- `int ovgr::solve_quad_eq` (double x[2], const double a, const double b, const double c)

2 次方程式 $ax^2 + bx + c = 0$ を解く

7.34 merge_ellipse.cpp

```
#include <stdio.h>
#include "extractFeature_old.h"
#include "ellipseIW.h"
```

merge_ellipse.cpp のインクルード依存関係図



マクロ定義

- #define INITIAL_ARRAYS_OK (0)
- #define INITIAL_ARRAYS_NG (1)
- #define SEARCH_ANOTHER_ARC_OK (0)
- #define SEARCH_ANOTHER_ARC_NG (1)
- #define REF_SELF (-1)
- #define REF_OK (0)
- #define REF_NG (1)
- #define NOELLIPSE (0)
- #define ELLIPSE_TERMINAL_PART (1)
- #define ELLIPSE_TERMINAL_WHOLE (2)
- #define SIGN_UNDEF (0)
- #define SIGN_INC (1)
- #define SIGN_DEC (2)
- #define SIGN_ZERO (3)

- #define [NTERMINAL](#) (2)
- #define [ANOTHER_EXIST_NG](#) (0)
- #define [ANOTHER_EXIST_OK](#) (1)
- #define [ANOTHER_EXIST_ERROR](#) (2)
- #define [ALL_REF_OK](#) (1)
- #define [ALL_REF_NG](#) (0)
- #define [TRY_ELLIPSE_TERMINATE](#) (0)
- #define [TRY_ELLIPSE_CONTINUE](#) (1)
- #define [TRY_ELLIPSE_REGISTER](#) (2)
- #define [ADD_NEW_MULTI_ELLIPSE_OK](#) (1)
- #define [ADD_NEW_MULTI_ELLIPSE_NG](#) (0)

型定義

- typedef struct _EllipseTerminal_ [EllipseTerminal](#)
- typedef struct _MergeEllipseArrays_ [MergeEllipseArrays](#)

関数

- int [merge_ellipse](#) ([Features2D_old](#) *f2D, const [ParamEllipseIW](#) *paramE)

7.34.1 マクロ定義

7.34.1.1 #define [ADD_NEW_MULTI_ELLIPSE_NG](#) (0)

7.34.1.2 #define [ADD_NEW_MULTI_ELLIPSE_OK](#) (1)

7.34.1.3 #define [ALL_REF_NG](#) (0)

7.34.1.4 #define [ALL_REF_OK](#) (1)

7.34.1.5 #define [ANOTHER_EXIST_ERROR](#) (2)

7.34.1.6 #define ANOTHER_EXIST_NG (0)

7.34.1.7 #define ANOTHER_EXIST_OK (1)

7.34.1.8 #define ELLIPSE_TERMINAL_PART (1)

7.34.1.9 #define ELLIPSE_TERMINAL_WHOLE (2)

7.34.1.10 #define INITIAL_ARRAYS_NG (1)

7.34.1.11 #define INITIAL_ARRAYS_OK (0)

7.34.1.12 #define NOELLIPSE (0)

7.34.1.13 #define NTERMINAL (2)

7.34.1.14 #define REF_NG (1)

7.34.1.15 #define REF_OK (0)

7.34.1.16 #define REF_SELF (-1)

7.34.1.17 **#define SEARCH_ANOTHER_ARC_NG (1)**

7.34.1.18 **#define SEARCH_ANOTHER_ARC_OK (0)**

7.34.1.19 **#define SIGN_DEC (2)**

7.34.1.20 **#define SIGN_INC (1)**

7.34.1.21 **#define SIGN_UNDEF (0)**

7.34.1.22 **#define SIGN_ZERO (3)**

7.34.1.23 **#define TRY_ELLIPSE_CONTINUE (1)**

7.34.1.24 **#define TRY_ELLIPSE_REGISTER (2)**

7.34.1.25 **#define TRY_ELLIPSE_TERMINATE (0)**

7.34.2 型定義

7.34.2.1 **typedef struct _EllipseTerminal_ EllipseTerminal**

7.34.2.2 `typedef struct _MergeEllipseArrays_ MergeEllipseArrays`

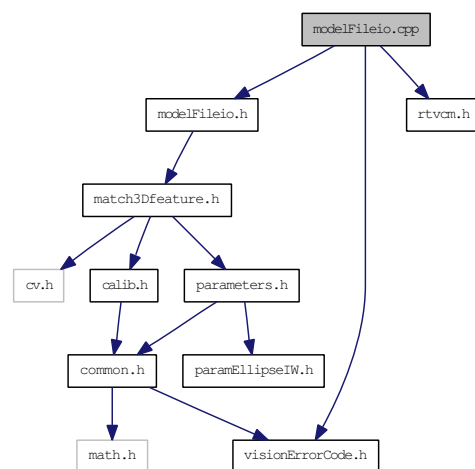
7.34.3 関数

7.34.3.1 `int merge_ellipse (Features2D_old *f2D, const ParamEllipseIW *paramE)`

7.35 modelFileio.cpp

```
#include "modelFileio.h"  
#include "rtvcm.h"  
#include "visionErrorCode.h"
```

modelFileio.cpp のインクルード依存関係図



関数

- int `loadModelFile` (char *path, `Features3D` &model)
モデルデータをファイルから読み込む。

7.35.1 関数

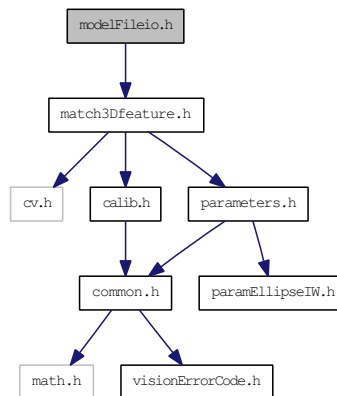
7.35.1.1 int loadModelFile (char * path, `Features3D` & model)

モデルデータをファイルから読み込む。

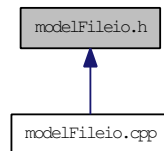
7.36 modelFileio.h

モデルをファイルから読み込む。#include "match3Dfeature.h"

modelFileio.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- int **loadModelFile** (char *path, **Features3D** &model)
モデルデータをファイルから読み込む。

7.36.1 説明

モデルをファイルから読み込む。

7.36.2 関数

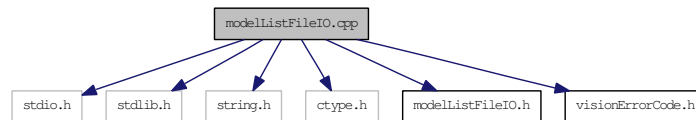
7.36.2.1 int loadModelFile (char * *path*, Features3D & *model*)

モデルデータをファイルから読み込む。

7.37 modelListFileIO.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "modelListFileIO.h"
#include "visionErrorCode.h"
```

modelListFileIO.cpp のインクルード依存関係図



関数

- `int loadModelListFile (char *filename, ModelFileInfo *mfInfo)`
- `void clearModelFileInfo (ModelFileInfo *mfInfo)`
モデルリストをクリアする。

7.37.1 関数

7.37.1.1 void clearModelFileInfo (ModelFileInfo * mfInfo)

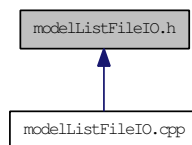
モデルリストをクリアする。

7.37.1.2 int loadModelListFile (char *filename, ModelFileInfo * mfInfo)

モデル一覧ファイルを読み込んで、モデル ID とモデルファイル名の対応リストを保持する。

7.38 modelListFileIO.h

モデルファイルの入出力関連このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [ModelFileInfoNode](#)
モデルリストのノード
- struct [ModelFileInfo](#)
モデルファイルリスト

マクロ定義

- #define [MAX_PATH](#) 256

関数

- int [loadModelListFile](#) (char *filename, [ModelFileInfo](#) *mfInfo)
- void [clearModelFileInfo](#) ([ModelFileInfo](#) *mfInfo)
モデルリストをクリアする。

7.38.1 説明

モデルファイルの入出力関連

7.38.2 マクロ定義

7.38.2.1 `#define MAX_PATH 256`

7.38.3 関数

7.38.3.1 `void clearModelFileInfo (ModelFileInfo * mfInfo)`

モデルリストをクリアする。

7.38.3.2 `int loadModelListFile (char *filename, ModelFileInfo * mfInfo)`

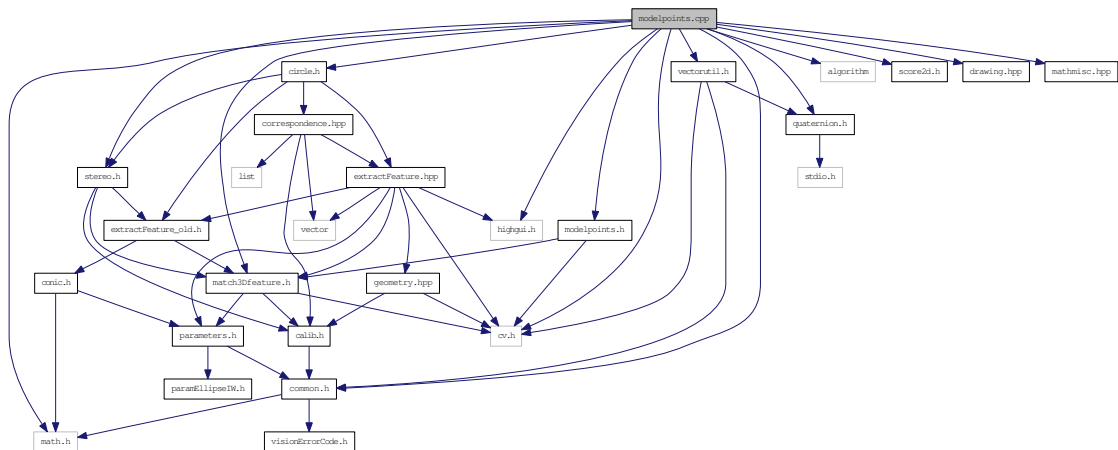
モデル一覧ファイルを読み込んで、モデル ID とモデルファイル名の対応リストを保持する。

7.39 modelpoints.cpp

モデル評価点生成関連関数 `#include <math.h>`

```
#include <cv.h>
#include <highgui.h>
#include <algorithm>
#include "common.h"
#include "quaternion.h"
#include "circle.h"
#include "stereo.h"
#include "match3Dfeature.h"
#include "score2d.h"
#include "vectorutil.h"
#include "modelpoints.h"
#include "drawing.hpp"
#include "mathmisc.hpp"
```

modelpoints.cpp のインクルード依存関係図



関数

- void `drawModelPoints` (`Features3D *model`, `double matrix[4][4]`, `char *filename`, `int p_camera`, `unsigned char *img`, `int lineThickness`)

モデル評価点の描画（認識結果確認表示用）

- int `makeModelPoints` (`Features3D` **model*, double *pdist*)
- void `getPropertyVector` (double *mat*[4][4], double *vec*[7])
 合同変換行列を位置ベクトルと回転ベクトルを合わせた 7 次元ベクトルに変換する
- double `calcEvaluationValue2DMultiCameras` (`Features3D` **model*,
`StereoPairing` &*pairing*, `MatchResult` **result*, `plot_t` **plot*, const std::vector<
 cv::Mat > &*dstImages*)

7.39.1 説明

モデル評価点生成関連関数

日付:

\$Date:: 2011-11-22 12:02:19 +0900 # \$

7.39.2 関数

7.39.2.1 double `calcEvaluationValue2DMultiCameras` (`Features3D` **model*,
`StereoPairing` &*pairing*, `MatchResult` **result*, `plot_t` **plot*, const
 std::vector< cv::Mat > &*dstImages*)

使用した全画像を用いた 2 次元評価値計算。距離変換画像の利用 戻り値：2 次元
 評価値

7.39.2.2 void `drawModelPoints` (`Features3D` **model*, double *matrix*[4][4], char
 **filename*, int *p_camera*, unsigned char **img*, int *lineThickness*)

モデル評価点の描画（認識結果確認表示用）

7.39.2.3 void `getPropertyVector` (double *mat*[4][4], double *vec*[7])

合同変換行列を位置ベクトルと回転ベクトルを合わせた 7 次元ベクトルに変換する

7.39.2.4 int `makeModelPoints` (`Features3D` **model*, double *pdist*)

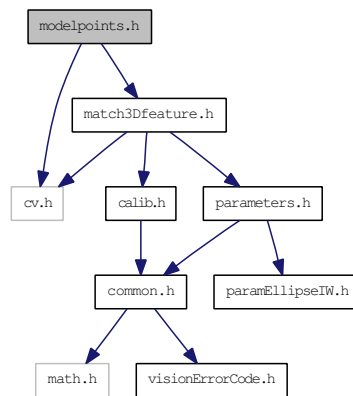
モデルの評価点の生成 戻り値：総評価点数

7.40 modelpoints.h

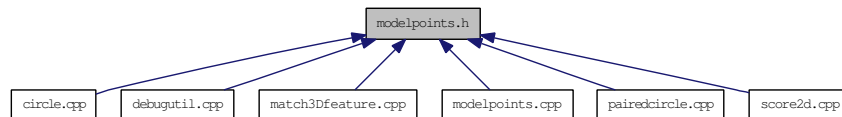
モデル評価点生成関連関数 `#include <cv.h>`

`#include "match3Dfeature.h"`

modelpoints.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



型定義

- `typedef std::vector< cv::Point > plot_t`

関数

- `void drawModelPoints (Features3D *model, double matrix[4][4], char *filename, int p_camera, unsigned char *img, int lineThickness)`
モデル評価点の描画（認識結果確認表示用）
- `int makeModelPoints (Features3D *model, double pdist)`
- `void getPropertyVector (double mat[4][4], double vec[7])`
合同変換行列を位置ベクトルと回転ベクトルを合わせた 7 次元ベクトルに変換する

- double `calcEvaluationValue2DMultiCameras` (`Features3D` **model*, `StereoPairing` &*pairing*, `MatchResult` **result*, `plot_t` **plot*, const std::vector< cv::Mat > &*dstImages*)
- int `isValidPixelPosition` (int *col*, int *row*, `Features3D` **finfo*)

7.40.1 説明

モデル評価点生成関連関数

日付:

\$Date:: 2011-10-20 13:55:43 +0900 # \$

7.40.2 型定義

7.40.2.1 `typedef std::vector<cv::Point> plot_t`

7.40.3 関数

7.40.3.1 `double calcEvaluationValue2DMultiCameras (Features3D * model, StereoPairing & pairing, MatchResult * result, plot_t * plot, const std::vector< cv::Mat > & dstImages)`

使用した全画像を用いた 2 次元評価値計算。距離変換画像の利用 戻り値：2 次元評価値

7.40.3.2 `void drawModelPoints (Features3D * model, double matrix[4][4], char * filename, int p_camera, unsigned char * img, int lineThickness)`

モデル評価点の描画（認識結果確認表示用）

7.40.3.3 `void getPropertyVector (double mat[4][4], double vec[7])`

合同変換行列を位置ベクトルと回転ベクトルを合わせた 7 次元ベクトルに変換する

7.40.3.4 `int isValidPixelPosition (int col, int row, Features3D * finfo)`
[inline]

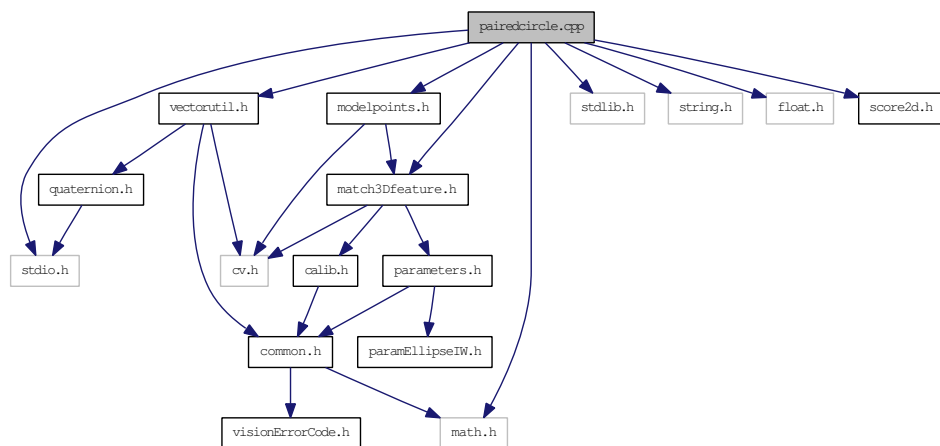
7.40.3.5 int makeModelPoints (Features3D * *model*, double *pdist*)

モデルの評価点の生成 戻り値：総評価点数

7.41 pairedcircle.cpp

```
2 円照合関連関数#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <float.h>
#include "match3Dfeature.h"
#include "score2d.h"
#include "vectorutil.h"
#include "modelpoints.h"
```

pairedcircle.cpp のインクルード依存関係図



関数

- [Match3Dresults matchPairedCircles](#) ([Features3D](#) &scene, [Features3D](#) &model, const std::vector< cv::Mat > &dstImages, double tolerance, [StereoPairing](#) &pairing)

7.41.1 説明

2 円照合関連関数

日付:

\$Date:: 2011-09-14 18:26:27 +0900 #

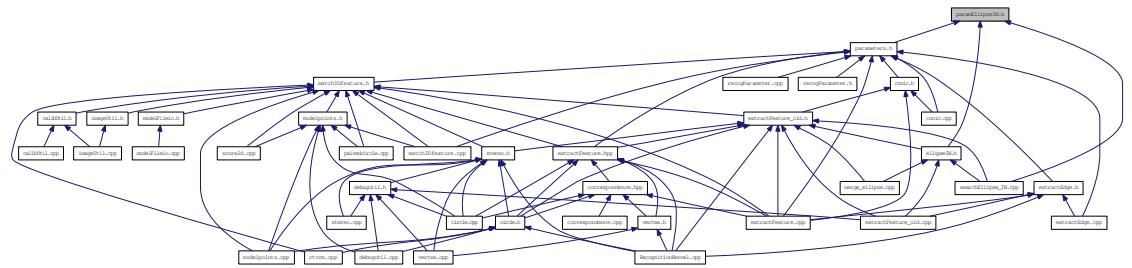
7.41.2 関数

7.41.2.1 Match3Dresults matchPairedCircles (Features3D & *scene*, Features3D & *model*, const std::vector< cv::Mat > & *dstImages*, double *tolerance*, StereoPairing & *pairing*)

2 円を使った照合 戻り値 : 認識結果

7.42 paramEllipseIW.h

このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [_param_ellipse_IW_](#)

マクロ定義

- #define [MINIMUM_MIN_LENGTH](#) (5)
- #define [MINIMUM_MIN_SHORT_RAD](#) (0.0)
- #define [MINIMUM_TH_MEAN_ERROR](#) (0.0)
- #define [MINIMUM_TH_MAX_ERROR](#) (0.0)
- #define [DEF_PARAME_CONDITION](#) (ELLIPSE_CONDITION_MEAN)
- #define [DEF_PARAME_MIN_LENGTH](#) (20)
- #define [DEF_PARAME_POST_MIN_LENGTH](#) (50)
- #define [DEF_PARAME_MIN_SHORT_RAD_PREV](#) (1.0)
- #define [DEF_PARAME_MIN_SHORT_RAD_POST](#) (2.0)
- #define [DEF_PARAME_TH_MEAN_ERROR](#) (0.4)
- #define [DEF_PARAME_TH_MAX_ERROR](#) (1.0)
- #define [DEF_PARAME_TH_MEAN_ERROR_MERGING](#) (0.44)
- #define [DEF_PARAME_TH_MAX_ERROR_MERGING](#) (1.1)
- #define [DEF_PARAME_MIN_SD](#) (0.5)
- #define [DEF_PARAME_OFFSET_MODE](#) (ELLIPSE_OFFSET_DYNAMIC)
- #define [DEF_PARAME_SW_LINE_ELLIPSE](#)
- #define [DEF_PARAME_SW_OLD_MERGE_FUNC](#) (ENABLE_OLD_MERGE_FUNC)
- #define [DEF_SHORTEN_ELLIPSE_MERGING](#) (0)

型定義

- typedef struct `_param_ellipse_IW_ ParamEllipseIW`

列挙型

- enum `paramEllipseIW_Ellipse_with_line_key` { `ENABLE_ELLIPSE_NONE`, `ENABLE_ELLIPSE_WITH_LINE`, `ENABLE_ELLIPSE_WITHOUT_LINE` }
- enum `paramEllipseIW_Old_Merge_func_key` { `DISABLE_OLD_MERGE_FUNC`, `ENABLE_OLD_MERGE_FUNC` }
- enum `paramEllipseIW_ErrCond_key` { `ELLIPSE_CONDITION_MEAN`, `ELLIPSE_CONDITION_MAX` }
- enum `paramEllipseIW_OffsetMode_key` { `ELLIPSE_OFFSET_STATIC`, `ELLIPSE_OFFSET_DYNAMIC` }

7.42.1 マクロ定義

7.42.1.1 `#define DEF_PARAME_CONDITION (ELLIPSE_CONDITION_MEAN)`

7.42.1.2 `#define DEF_PARAME_MIN_LENGTH (20)`

7.42.1.3 `#define DEF_PARAME_MIN_SD (0.5)`

7.42.1.4 `#define DEF_PARAME_MIN_SHORT_RAD_POST (2.0)`

7.42.1.5 `#define DEF_PARAME_MIN_SHORT_RAD_PREV (1.0)`

7.42.1.6 `#define DEF_PARAME_OFFSET_MODE (ELLIPSE_OFFSET_DYNAMIC)`

7.42.1.7 #define DEF_PARAME_POST_MIN_LENGTH (50)

7.42.1.8 #define DEF_PARAME_SW_LINE_ELLIPSE

値:

```
(ENABLE_ELLIPSE_WITH_LINE | \
                                ENABLE_ELLIPSE_WITHOUT_LINE)
```

**7.42.1.9 #define DEF_PARAME_SW_OLD_MERGE_FUNC (ENABLE_-
OLD_MERGE_FUNC)**

7.42.1.10 #define DEF_PARAME_TH_MAX_ERROR (1.0)

7.42.1.11 #define DEF_PARAME_TH_MAX_ERROR_MERGING (1.1)

7.42.1.12 #define DEF_PARAME_TH_MEAN_ERROR (0.4)

7.42.1.13 #define DEF_PARAME_TH_MEAN_ERROR_MERGING (0.44)

7.42.1.14 #define DEF_SHORTEN_ELLIPSE_MERGING (0)

7.42.1.15 #define MINIMUM_MIN_LENGTH (5)

7.42.1.16 `#define MINIMUM_MIN_SHORT_RAD (0.0)`

7.42.1.17 `#define MINIMUM_TH_MAX_ERROR (0.0)`

7.42.1.18 `#define MINIMUM_TH_MEAN_ERROR (0.0)`

7.42.2 型定義

7.42.2.1 `typedef struct _param_ellipse_IW_ ParamEllipseIW`

7.42.3 列挙型

7.42.3.1 `enum paramEllipseIW_Ellipse_with_line_key`

列挙型の値:

ENABLE_ELLIPSE_NONE
ENABLE_ELLIPSE_WITH_LINE
ENABLE_ELLIPSE_WITHOUT_LINE

7.42.3.2 `enum paramEllipseIW_ErrCond_key`

列挙型の値:

ELLIPSE_CONDITION_MEAN
ELLIPSE_CONDITION_MAX

7.42.3.3 `enum paramEllipseIW_OffsetMode_key`

列挙型の値:

ELLIPSE_OFFSET_STATIC
ELLIPSE_OFFSET_DYNAMIC

7.42.3.4 enum paramEllipseIW_Old_Merge_func_key

列挙型の値:

DISABLE_OLD_MERGE_FUNC

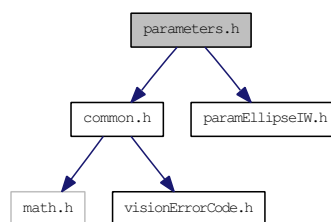
ENABLE_OLD_MERGE_FUNC

7.43 parameters.h

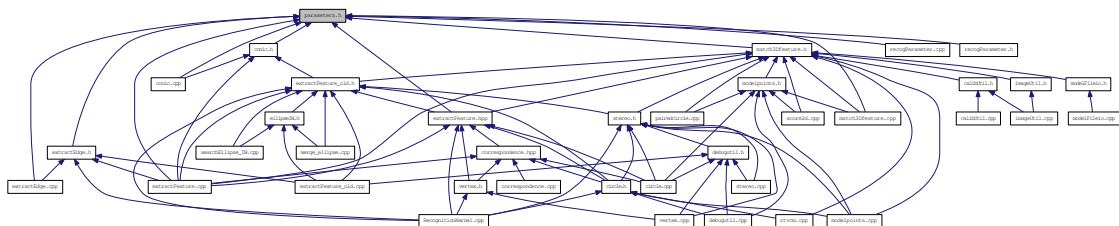
処理パラメータ設定関連関数 `#include "common.h"`

`#include "paramEllipseIW.h"`

parameters.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [ParametersFeature2D](#)
2次元特徴抽出用パラメータ
- struct [ParametersStereo](#)
ステレオ対応処理用パラメータ
- struct [ParametersMatch](#)
認識用パラメータ
- struct [Parameters](#)
全パラメータ

型定義

- typedef struct [ParametersStereo](#) [ParametersStereo](#)
ステレオ対応処理用パラメータ

7.43.1 説明

処理パラメータ設定関連関数

日付:

\$Date:: 2011-11-24 08:44:03 +0900 #

7.43.2 型定義

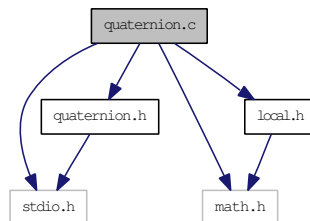
7.43.2.1 typedef struct [ParametersStereo](#) [ParametersStereo](#)

ステレオ対応処理用パラメータ

7.44 quaternion.c

```
#include <stdio.h>
#include <math.h>
#include "quaternion.h"
#include "local.h"
```

quaternion.c のインクルード依存関係図



関数

- void [quat_fprintf](#) (FILE *fp, const char *fmt, const char sep, const [quaternion_t](#) q)
- void [quat_copy](#) ([quaternion_t](#) dst, const [quaternion_t](#) src)
- void [quat_mult](#) ([quaternion_t](#) result, const [quaternion_t](#) q1, const [quaternion_t](#) q2)
- void [quat_conj](#) ([quaternion_t](#) result, const [quaternion_t](#) q)
- double [quat_norm2](#) (const [quaternion_t](#) q)
- double [quat_normalize](#) ([quaternion_t](#) q)
- void [quat_rot](#) (double result[3], const [quaternion_t](#) q, const double x[3])
- void [quat_ivot](#) (double result[3], const [quaternion_t](#) q, const double x[3])
- void [quat_make_from_rvec](#) ([quaternion_t](#) result, const double theta, const double x, const double y, const double z)
- void [quat_R_from_q](#) (double *R, const int ldim, const [quaternion_t](#) q)
- void [quat_q_from_R](#) ([quaternion_t](#) q, const double *R, const int ldim)

7.44.1 関数

7.44.1.1 void [quat_conj](#) ([quaternion_t](#) result, const [quaternion_t](#) q)

- 7.44.1.2 `void quat_copy (quaternion_t dst, const quaternion_t src)`
- 7.44.1.3 `void quat_fprintf (FILE *fp, const char *fmt, const char sep, const quaternion_t q)`
- 7.44.1.4 `void quat_irot (double result[3], const quaternion_t q, const double x[3])`
- 7.44.1.5 `void quat_make_from_rvec (quaternion_t result, const double theta, const double x, const double y, const double z)`
- 7.44.1.6 `void quat_mult (quaternion_t result, const quaternion_t q1, const quaternion_t q2)`
- 7.44.1.7 `double quat_norm2 (const quaternion_t q)`
- 7.44.1.8 `double quat_normalize (quaternion_t q)`
- 7.44.1.9 `void quat_q_from_R (quaternion_t q, const double *R, const int ldim)`
- 7.44.1.10 `void quat_R_from_q (double *R, const int ldim, const quaternion_t q)`
- 7.44.1.11 `void quat_rot (double result[3], const quaternion_t q, const double x[3])`

- void `quat_mult` (`quaternion_t` result, const `quaternion_t` q1, const `quaternion_t` q2)
- void `quat_conj` (`quaternion_t` result, const `quaternion_t` q)
- double `quat_norm2` (const `quaternion_t` q)
- double `quat_normalize` (`quaternion_t` q)
- void `quat_rot` (double result[3], const `quaternion_t` q, const double x[3])
- void `quat_irot` (double result[3], const `quaternion_t` q, const double x[3])
- void `quat_make_from_rvec` (`quaternion_t` result, const double theta, const double x, const double y, const double z)
- void `quat_R_from_q` (double *R, const int ldim, const `quaternion_t` q)
- void `quat_q_from_R` (`quaternion_t` q, const double *R, const int ldim)

7.45.1 マクロ定義

7.45.1.1 `#define QUAT_EPS 1.0e-15`

7.45.1.2 `#define quat_fprint(fp, q) quat_fprintf((fp), "% 10.3g", ' ', (q))`

7.45.1.3 `#define quat_im(q, i) (q)[(i)]`

7.45.1.4 `#define QUAT_INIT_ONE {0.0, 0.0, 0.0, 1.0}`

7.45.1.5 `#define QUAT_INIT_ZERO {0.0, 0.0, 0.0, 0.0}`

7.45.1.6 `#define quat_print(q) quat_fprintf(stdout, "% 10.3g", ' ', (q))`

7.45.1.7 `#define quat_printf(fmt, sep, q) quat_fprintf(stdout, (fmt), (sep), (q))`

7.45.1.8 `#define quat_re(q) (q)[3]`

7.45.2 型定義

7.45.2.1 `typedef double quaternion_t[4]`

7.45.3 関数

7.45.3.1 `void quat_conj (quaternion_t result, const quaternion_t q)`

7.45.3.2 `void quat_copy (quaternion_t dst, const quaternion_t src)`

7.45.3.3 `void quat_fprintf (FILE *fp, const char *fmt, const char sep, const quaternion_t q)`

7.45.3.4 `void quat_irot (double result[3], const quaternion_t q, const double x[3])`

7.45.3.5 `void quat_make_from_rvec (quaternion_t result, const double theta, const double x, const double y, const double z)`

7.45.3.6 `void quat_mult (quaternion_t result, const quaternion_t q1, const quaternion_t q2)`

7.45.3.7 `double quat_norm2 (const quaternion_t q)`

7.45.3.8 `double quat_normalize (quaternion_t q)`

7.45.3.9 `void quat_q_from_R (quaternion_t q, const double * R, const int ldim)`

7.45.3.10 `void quat_R_from_q (double * R, const int ldim, const quaternion_t q)`

7.45.3.11 `void quat_rot (double result[3], const quaternion_t q, const double x[3])`

7.46 recogImage.cpp

画像入出力関数 `#include <stdlib.h>`

`#include <string.h>`

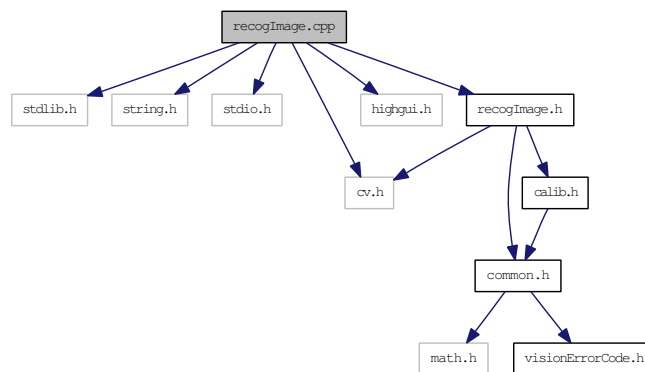
`#include <stdio.h>`

`#include <cv.h>`

`#include <highgui.h>`

`#include "recogImage.h"`

recogImage.cpp のインクルード依存関係図



関数

- `RecogImage * constructImage` (const int colsize, const int rowsize, const int bytePerPixel)
画像メモリの確保と初期化
- `RecogImage * convertImage` (const cv::Mat &cvimg)
OpenCV 画像からの変換.
- void `destructImage` (RecogImage *image)
画像メモリの解放
- void `rgb2grayImage` (RecogImage *target, RecogImage *source)
RGB 画像から *Grey* 画像への変換.
- void `undistortImage` (const RecogImage *src, const CameraParam *cp, RecogImage *dst, CameraParam *cp_dst)
- void `writeRecogImage` (const char *filename, const RecogImage *img)

画像メモリのファイル出力 デバッグ用

7.46.1 説明

画像入出力関数

日付:

\$Date:: 2011-10-21 14:49:57 +0900 #

7.46.2 関数

7.46.2.1 **RecogImage* constructImage (const int *colsize*, const int *rowsize*, const int *bytePerPixel*)**

画像メモリの確保と初期化

7.46.2.2 **RecogImage* convertImage (const cv::Mat & *cving*)**

OpenCV 画像からの変換.

7.46.2.3 **void destructImage (RecogImage * *image*)**

画像メモリの解放

7.46.2.4 **void rgb2grayImage (RecogImage * *target*, RecogImage * *source*)**

RGB 画像から Grey 画像への変換.

7.46.2.5 **void undistortImage (const RecogImage * *src*, const CameraParam * *cp*, RecogImage * *dst*, CameraParam * *cp_dst*)**

7.46.2.6 **void writeRecogImage (const char * *filename*, const RecogImage * *img*)**

画像メモリのファイル出力 デバッグ用

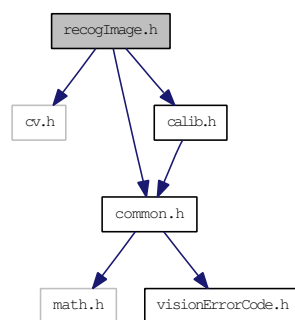
7.47 recogImage.h

画像入出力関数 `#include <cv.h>`

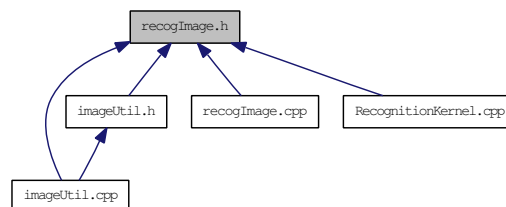
`#include "common.h"`

`#include "calib.h"`

recogImage.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [_recogImage](#)

型定義

- typedef struct [_recogImage](#) [RecogImage](#)

関数

- [RecogImage](#) * [constructImage](#) (const int colsize, const int rowsize, const int bytePerPixel)

画像メモリの確保と初期化

- `RecogImage * convertImage (const cv::Mat &cving)`
OpenCV 画像からの変換.
- `void destructImage (RecogImage *image)`
画像メモリの解放
- `void rgb2grayImage (RecogImage *target, RecogImage *source)`
RGB 画像から Grey 画像への変換.
- `void undistortImage (const RecogImage *src, const CameraParam *cp, RecogImage *dst, CameraParam *cp_dst)`
- `void writeRecogImage (const char *filename, const RecogImage *img)`
画像メモリのファイル出力 デバッグ用

7.47.1 説明

画像入出力関数

日付:

\$Date:: 2011-09-29 08:56:58 +0900 # \$

7.47.2 型定義

7.47.2.1 `typedef struct _recogImage RecogImage`

7.47.3 関数

7.47.3.1 `RecogImage* constructImage (const int colsize, const int rowsize, const int bytePerPixel)`

画像メモリの確保と初期化

7.47.3.2 `RecogImage* convertImage (const cv::Mat & cving)`

OpenCV 画像からの変換.

7.47.3.3 void destructImage (RecogImage * *image*)

画像メモリの解放

7.47.3.4 void rgb2grayImage (RecogImage * *target*, RecogImage * *source*)

RGB 画像から Grey 画像への変換.

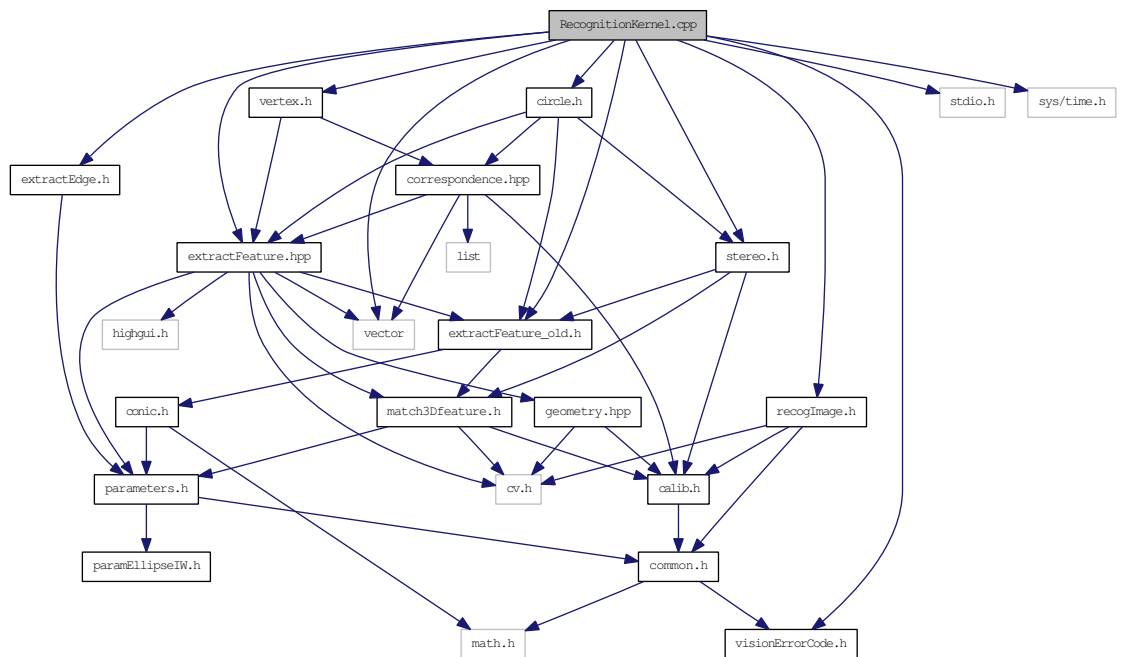
7.47.3.5 void undistortImage (const RecogImage * *src*, const CameraParam * *cp*, RecogImage * *dst*, CameraParam * *cp_dst*)**7.47.3.6 void writeRecogImage (const char * *filename*, const RecogImage * *img*)**

画像メモリのファイル出力 デバッグ用

7.48 RecognitionKernel.cpp

```
#include <vector>
#include <stdio.h>
#include "recogImage.h"
#include "stereo.h"
#include "circle.h"
#include "vertex.h"
#include "extractEdge.h"
#include "extractFeature_old.h"
#include "extractFeature.hpp"
#include "visionErrorCode.h"
#include <sys/time.h>
```

RecognitionKernel.cpp のインクルード依存関係図



関数

- [Match3Dresults RecognitionKernel](#) ([RecogImage](#) **image, [CalibParam](#) &calib, [Features3D](#) &model, [Parameters](#) ¶m)
三次元物体認識の実行

7.48.1 関数

7.48.1.1 [Match3Dresults RecognitionKernel](#) ([RecogImage](#) ** *image*, [CalibParam](#) & *calib*, [Features3D](#) & *model*, [Parameters](#) & *param*)

三次元物体認識の実行 画像データから 三次元特徴を抽出し、モデルとマッチングを行い、結果を返す。 引数: [RecogImage](#)** image : カメラ画像 [CalibParam](#)& calib : カメラキャリブレーションデータ [Features3D](#)& model : 対象物体モデル [Parameters](#)& param : 認識パラメータ

7.49 RecognitionKernel.h

3次元物体認識の中核処理

関数

- [Match3Dresults RecognitionKernel](#) ([RecogImage](#) **image, [CalibParam](#) &calib, [Features3D](#) &model, [Parameters](#) ¶m)

三次元物体認識の実行

7.49.1 説明

3次元物体認識の中核処理

7.49.2 関数

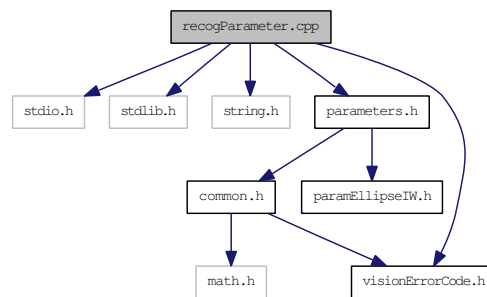
7.49.2.1 [Match3Dresults RecognitionKernel](#) ([RecogImage](#) ** *image*, [CalibParam](#) & *calib*, [Features3D](#) & *model*, [Parameters](#) & *param*)

三次元物体認識の実行 画像データから 三次元特徴を抽出し、モデルとマッチングを行い、結果を返す。引数: [RecogImage](#)** image : カメラ画像 [CalibParam](#)& calib : カメラキャリブレーションデータ [Features3D](#)& model : 対象物体モデル [Parameters](#)& param : 認識パラメータ

7.50 recogParameter.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "parameters.h"
#include "visionErrorCode.h"
```

recogParameter.cpp のインクルード依存関係図



列挙型

- enum paramKey {
 - eStereoPair, eOutputCandNum, eEdgeDetectFunction, eEdgeStrength,
 - eMaxErrorOfLineFit, eMaxDistanceSimilarLine, eMaxErrorOfConicFit,
 - eOverlapRatioLine,
 - eOverlapRatioCircle, eMinLengthLine2D, eHDMMax, eNoSearchFeatures,
 - eDepN, eDepF, eAMin, eAMax,
 - eStereoError, eIwCondition, eIwMinLength, eIwPostMinLength,
 - eIwMinShortRadPrev, eIwMinShortRadPost, eIwThMeanErrorMerging,
 - eIwThMaxErrorMerging,
 - eIwThMeanError, eIwThMaxError, eIwMinSD, eIwOffsetMode,
 - eIwSwLineEllipse, eIwSwOldMergeFunc, eIwShortenEllipseMerging,
 - eParamSentinel }

関数

- void setDefaultRecogParameter (Parameters ¶m)
 - Parameters 構造体にデフォルト値をセットする。.

- int `loadRecogParameter` (char *path, `Parameters` ¶m)
ファイルから、`Parameters` 構造体に設定値を読み込む。
- int `loadDebugParameter` (int text, int image, int display, `Parameters` ¶m)

7.50.1 列挙型

7.50.1.1 enum paramKey

列挙型の値:

eStereoPair
eOutputCandNum
eEdgeDetectFunction
eEdgeStrength
eMaxErrorOfLineFit
eMaxDistanceSimilarLine
eMaxErrorOfConicFit
eOverlapRatioLine
eOverlapRatioCircle
eMinLengthLine2D
eHDMax
eNoSearchFeatures
eDepN
eDepF
eAMin
eAMax
eStereoError
eIwCondition
eIwMinLength
eIwPostMinLength
eIwMinShortRadPrev
eIwMinShortRadPost
eIwThMeanErrorMerging
eIwThMaxErrorMerging
eIwThMeanError
eIwThMaxError

eIwMinSD
eIwOffsetMode
eIwSwLineEllipse
eIwSwOldMergeFunc
eIwShortenEllipseMerging
eParamSentinel

7.50.2 関数

7.50.2.1 `int loadDebugParameter (int text, int image, int display, Parameters & param)`

7.50.2.2 `int loadRecogParameter (char * path, Parameters & param)`

ファイルから、Parameters 構造体に設定値を読み込む。

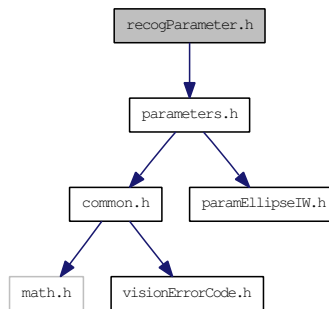
7.50.2.3 `void setDefaultRecogParameter (Parameters & param)`

Parameters 構造体にデフォルト値をセットする。.

7.51 recogParameter.h

認識パラメータ設定関連 `#include "parameters.h"`

recogParameter.h のインクルード依存関係図



関数

- void `setDefaultRecogParameter` (`Parameters` ¶m)
Parameters 構造体にデフォルト値をセットする。.
- int `loadRecogParameter` (char *path, `Parameters` ¶m)
ファイルから、*Parameters* 構造体に設定値を読み込む。
- int `loadDebugParameter` (int text, int image, int display, `Parameters` ¶m)

7.51.1 説明

認識パラメータ設定関連

7.51.2 関数

7.51.2.1 int loadDebugParameter (int text, int image, int display, `Parameters` & param)

7.51.2.2 int loadRecogParameter (char * *path*, Parameters & *param*)

ファイルから、Parameters 構造体に設定値を読み込む。

7.51.2.3 void setDefaultRecogParameter (Parameters & *param*)

Parameters 構造体にデフォルト値をセットする。 .

7.52 recogResult.h

認識結果の定義

マクロ定義

- #define `RecogResultElementNum` 20

列挙型

- enum `RecogResultElement` {
 `eRRCameraID`, `eRRModelID`, `eRRCandNo`, `eRRCoordNo`,
 `eRRRecogReliability`, `eRRErrorCode`, `eRRReserve1`, `eRRReserve2`,
 `eRRR00`, `eRRR01`, `eRRR02`, `eRRTx`,
 `eRRR10`, `eRRR11`, `eRRR12`, `eRRTy`,
 `eRRR20`, `eRRR21`, `eRRR22`, `eRRTz` }

7.52.1 説明

認識結果の定義

7.52.2 マクロ定義

7.52.2.1 #define `RecogResultElementNum` 20

7.52.3 列挙型

7.52.3.1 enum `RecogResultElement`

列挙型の値:

`eRRCameraID`
`eRRModelID`
`eRRCandNo`
`eRRCoordNo`

eRRRecogReliability

eRRErrorCode

eRRReserve1

eRRReserve2

eRRR00

eRRR01

eRRR02

eRRTx

eRRR10

eRRR11

eRRR12

eRRTy

eRRR20

eRRR21

eRRR22

eRRTz

7.53 rtvcm.cpp

モデル入出力関連関数 `#include <iostream>`

`#include <fstream>`

`#include <string>`

`#include <cmath>`

`#include "match3Dfeature.h"`

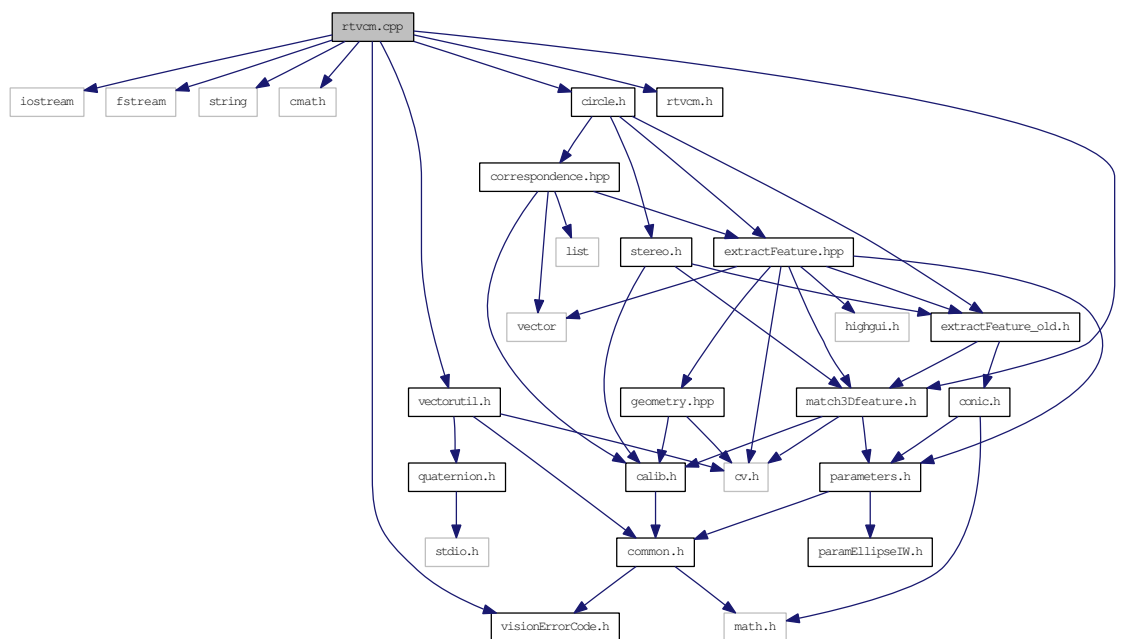
`#include "vectorutil.h"`

`#include "rtvcm.h"`

`#include "visionErrorCode.h"`

`#include "circle.h"`

rtvcm.cpp のインクルード依存関係図



関数

- void [freeRTVCM](#) ([RTVCM](#) &rtvcm)
モデルデータのメモリ解放
- int [readRTVCMModel](#) (char *filename, [RTVCM](#) &rtvcm)

- void `reverseVertex` (`Vertex` src, `Vertex` &dst)
3次元頂点データの裏データ作成
- void `reverseCircle` (`Circle` src, `Circle` &dst)
3次元円データの裏データ作成
- void `create_wireframe_model` (`Features3D` *feature)
- int `convertRTVCMtoFeatures3D` (`RTVCM` rtvcm, `Features3D` &feature)
モデルデータから3次元特徴データへの変換

7.53.1 説明

モデル入出力関連関数

日付:

\$Date:: 2011-11-22 12:02:19 +0900 #

7.53.2 関数

7.53.2.1 int `convertRTVCMtoFeatures3D` (`RTVCM` rtvcm, `Features3D` & *feature*)

モデルデータから3次元特徴データへの変換

7.53.2.2 void `create_wireframe_model` (`Features3D` **feature*)

7.53.2.3 void `freeRTVCM` (`RTVCM` & *rtvcm*)

モデルデータのメモリ解放

7.53.2.4 int `readRTVCMModel` (`char` **filename*, `RTVCM` & *rtvcm*)

モデルデータの読み込み 戻り値: エラーコード

7.53.2.5 void `reverseCircle` (`Circle` *src*, `Circle` & *dst*)

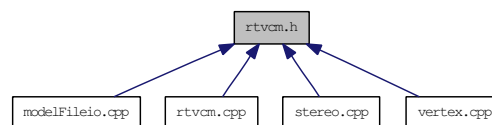
3次元円データの裏データ作成

7.53.2.6 void reverseVertex (Vertex *src*, Vertex & *dst*)

3次元頂点データの裏データ作成

7.54 rtvcm.h

モデル入出力関連関数このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [RTVCM_Vertex](#)
モデル内の頂点データ
- struct [RTVCM_Circle](#)
モデル内の円データ
- struct [RTVCM_Box](#)
モデル内の立方体データ
- struct [RTVCM_Cylinder](#)
モデル内の円筒データ
- struct [RTVertexCircleModel](#)
モデルデータ構造体

型定義

- typedef int [RTVCM_Label](#)
- typedef struct [RTVertexCircleModel](#) [RTVCM](#)
モデルデータ構造体

関数

- void [freeRTVCM](#) ([RTVCM](#) &rtvcm)
モデルデータのメモリ解放
- int [readRTVCMModel](#) (char *filename, [RTVCM](#) &rtvcm)

- void `reverseVertex` (`Vertex` src, `Vertex` &dst)
3次元頂点データの裏データ作成
- void `reverseCircle` (`Circle` src, `Circle` &dst)
3次元円データの裏データ作成
- int `convertRTVCMtoFeatures3D` (`RTVCM` rtvcm, `Features3D` &feature)
モデルデータから3次元特徴データへの変換

7.54.1 説明

モデル入出力関連関数

日付:

\$Date:: 2011-09-09 14:00:23 +0900 # \$

7.54.2 型定義

7.54.2.1 typedef struct RTVertexCircleModel RTVCM

モデルデータ構造体

7.54.2.2 typedef int RTVCM_Label

7.54.3 関数

7.54.3.1 int `convertRTVCMtoFeatures3D` (`RTVCM` rtvcm, `Features3D` &feature)

モデルデータから3次元特徴データへの変換

7.54.3.2 void `freeRTVCM` (`RTVCM` & rtvcm)

モデルデータのメモリ解放

7.54.3.3 int `readRTVCMModel` (char *filename, `RTVCM` & rtvcm)

モデルデータの読み込み 戻り値: エラーコード

7.54.3.4 void reverseCircle (Circle *src*, Circle & *dst*)

3次元円データの裏データ作成

7.54.3.5 void reverseVertex (Vertex *src*, Vertex & *dst*)

3次元頂点データの裏データ作成

7.55 score2d.cpp

2次元評価関連関数 `#include <stdio.h>`

`#include <assert.h>`

`#include <stdlib.h>`

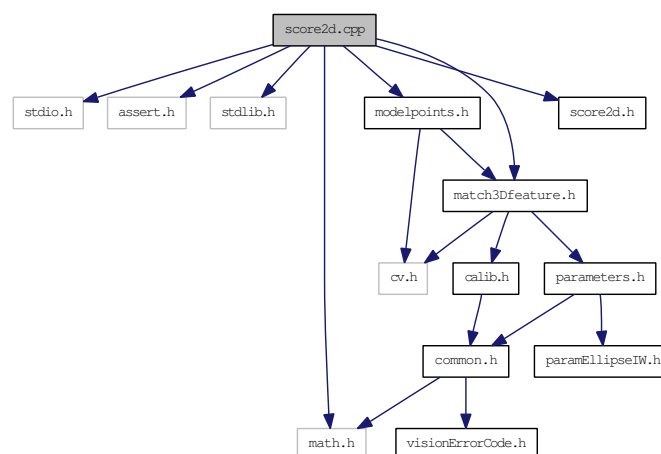
`#include <math.h>`

`#include "modelpoints.h"`

`#include "match3Dfeature.h"`

`#include "score2d.h"`

score2d.cpp のインクルード依存関係図



関数

- `int compareResultScore (const void *c1, const void *c2)`
- `void getResultScore (MatchResult *results, int numOfResults, Features3D *model, StereoPairing &pairing, const std::vector< cv::Mat > &dstImages, double weight)`

7.55.1 説明

2次元評価関連関数

日付:

\$Date:: 2011-10-25 13:49:23 +0900 # \$

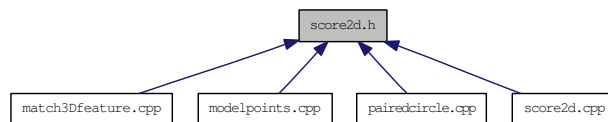
7.55.2 関数

7.55.2.1 `int compareResultScore (const void * c1, const void * c2)`

7.55.2.2 `void getResultScore (MatchResult * results, int numOfResults,
Features3D * model, StereoPairing & pairing, const std::vector<
cv::Mat > & dstImages, double weight)`

7.56 score2d.h

2次元評価関連関数このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- int [compareResultScore](#) (const void *c1, const void *c2)
- void [getResultScore](#) ([MatchResult](#) *results, int numOfResults, [Features3D](#) *model, [StereoPairing](#) &pairing, const std::vector< cv::Mat > &dstImages, double weight)

7.56.1 説明

2次元評価関連関数

日付:

\$Date:: 2011-09-14 19:02:24 +0900 #\$

7.56.2 関数

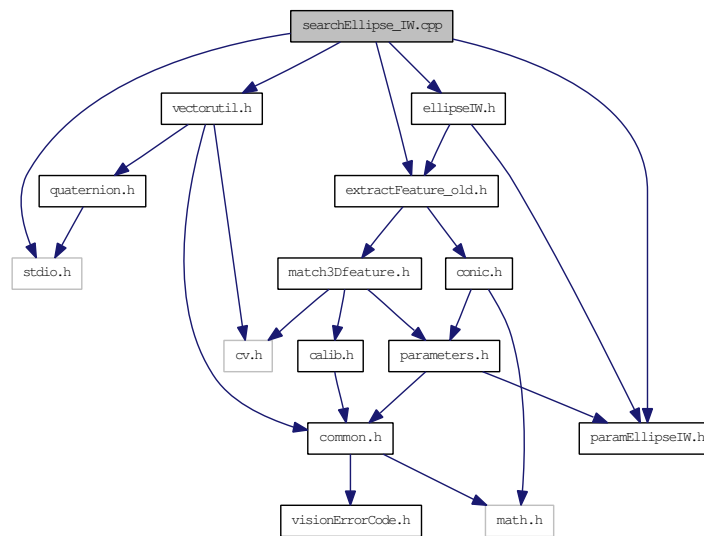
7.56.2.1 int [compareResultScore](#) (const void *c1, const void *c2)

7.56.2.2 void [getResultScore](#) ([MatchResult](#) *results, int numOfResults, [Features3D](#) *model, [StereoPairing](#) &pairing, const std::vector< cv::Mat > &dstImages, double weight)

7.57 searchEllipse_IW.cpp

```
#include <stdio.h>
#include "extractFeature_old.h"
#include "vectorutil.h"
#include "ellipseIW.h"
#include "paramEllipseIW.h"
```

searchEllipse_IW.cpp のインクルード依存関係図



マクロ定義

- #define MAX_CURVE_LEN_UNDEFINED (-1)
- #define TRACKING_OFF (0)
- #define TRACKING_ON (1)
- #define NDIR4 (4)
- #define DIR_START_DEC (1)
- #define DIR_START_INC (3)
- #define DIR_GOAL_DEC (2)
- #define DIR_GOAL_INC (0)
- #define TURN_LEFT (1)
- #define TURN_FORWARD (0)
- #define TURN_RIGHT (3)
- #define TURN_BACKWARD (2)

- `#define DS_SHIFT (1)`
- `#define DG_SHIFT (1)`
- `#define CHECK_NEXT_FAIL (0)`
- `#define CHECK_NEXT_SUCCESS (1)`
- `#define CHECK_TRACK_FAIL (0)`
- `#define CHECK_TRACK_SUCCESS (1)`
- `#define POINT_TO_ELLIPSE_FAIL (0)`
- `#define POINT_TO_ELLIPSE_SUCCESS (1)`
- `#define LOOP_CONTINUE (0)`
- `#define LOOP_EXIT_WHOLE (1)`
- `#define LOOP_EXIT_NORMAL (2)`
- `#define CHECK_SA_OK (0)`
- `#define CHECK_SA_NG (1)`
- `#define ADD_NEW_ELLIPSE_OK (1)`
- `#define ADD_NEW_ELLIPSE_NG (0)`
- `#define DS_SKIP (0)`
- `#define DG_SKIP (1)`
- `#define SKIP_LOOP_STOP (0)`
- `#define SKIP_LOOP_FULL (1)`
- `#define SKIP_LOOP_CONTINUE (2)`

関数

- `int mod_nPoint (int n, int nPoint)`
- `void addArcSum (SumSet *sum, const int *pointX, const double *offsetD)`
- `void avec_to_ellipse (int k_min_error, Ellipse *ellipse)`
- `double distanceAConic (const double coef[6], const int *point)`
- `void sum_to_P_dynamic (const SumSet *sum, Ellipse *ellipse, OffsetProp *offsetProp)`
- `void P_to_avec_and_fix (Ellipse *ellipse, const ParamEllipseIW *paramE)`
- `int check_ellipse_cond (Ellipse *ellipse, const ParamEllipseIW *paramE)`
- `int searchEllipseIW (Features2D_old *f2D, int iTrack, const ParamEllipseIW *paramE)`

7.57.1 マクロ定義

7.57.1.1 `#define ADD_NEW_ELLIPSE_NG (0)`

7.57.1.2 `#define ADD_NEW_ELLIPSE_OK (1)`

7.57.1.3 #define CHECK_NEXT_FAIL (0)

7.57.1.4 #define CHECK_NEXT_SUCCESS (1)

7.57.1.5 #define CHECK_SA_NG (1)

7.57.1.6 #define CHECK_SA_OK (0)

7.57.1.7 #define CHECK_TRACK_FAIL (0)

7.57.1.8 #define CHECK_TRACK_SUCCESS (1)

7.57.1.9 #define DG_SHIFT (1)

7.57.1.10 #define DG_SKIP (1)

7.57.1.11 #define DIR_GOAL_DEC (2)

7.57.1.12 #define DIR_GOAL_INC (0)

7.57.1.13 #define DIR_START_DEC (1)

7.57.1.14 **#define DIR_START_INC (3)**

7.57.1.15 **#define DS_SHIFT (1)**

7.57.1.16 **#define DS_SKIP (0)**

7.57.1.17 **#define LOOP_CONTINUE (0)**

7.57.1.18 **#define LOOP_EXIT_NORMAL (2)**

7.57.1.19 **#define LOOP_EXIT_WHOLE (1)**

7.57.1.20 **#define MAX_CURVE_LEN_UNDEFINED (-1)**

7.57.1.21 **#define NDIR4 (4)**

7.57.1.22 **#define POINT_TO_ELLIPSE_FAIL (0)**

7.57.1.23 **#define POINT_TO_ELLIPSE_SUCCESS (1)**

7.57.1.24 **#define SKIP_LOOP_CONTINUE (2)**

7.57.1.25 `#define SKIP_LOOP_FULL (1)`

7.57.1.26 `#define SKIP_LOOP_STOP (0)`

7.57.1.27 `#define TRACKING_OFF (0)`

7.57.1.28 `#define TRACKING_ON (1)`

7.57.1.29 `#define TURN_BACKWARD (2)`

7.57.1.30 `#define TURN_FORWARD (0)`

7.57.1.31 `#define TURN_LEFT (1)`

7.57.1.32 `#define TURN_RIGHT (3)`

7.57.2 関数

7.57.2.1 `void addArcSum (SumSet * sum, const int * pointX, const double * offsetD)`

7.57.2.2 `void avec_to_ellipse (int k_min_error, Ellipse * ellipse)`

7.57.2.3 `int check_ellipse_cond (Ellipse * ellipse, const ParamEllipseIW * paramE)`

7.57.2.4 `double distanceAConic (const double coef[6], const int * point)`

7.57.2.5 `int mod_nPoint (int n, int nPoint)`

7.57.2.6 `void P_to_avec_and_fix (Ellipse * ellipse, const ParamEllipseIW * paramE)`

7.57.2.7 `int searchEllipseIW (Features2D_old * f2D, int iTrack, const ParamEllipseIW * paramE)`

7.57.2.8 `void sum_to_P_dynamic (const SumSet * sum, Ellipse * ellipse, OffsetProp * offsetProp)`

7.58 stereo.cpp

ステレオ処理関連関数 `#include "common.h"`

`#include "stereo.h"`

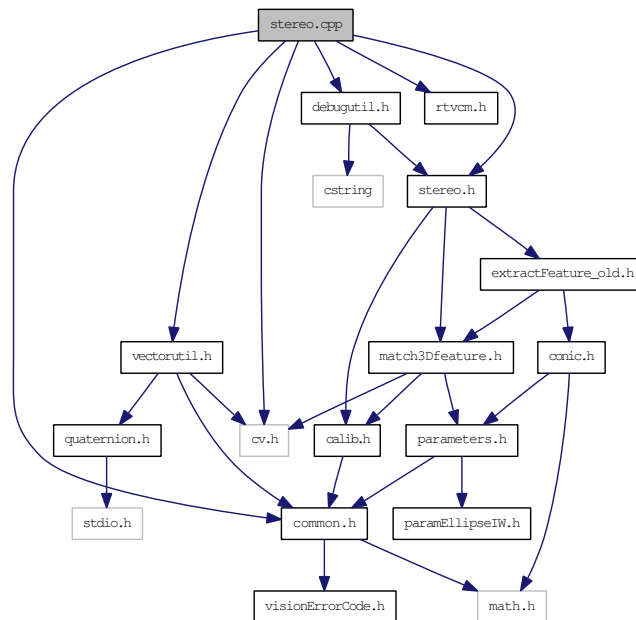
`#include "vectorutil.h"`

`#include "rtvcm.h"`

`#include "debugutil.h"`

`#include <cv.h>`

stereo.cpp のインクルード依存関係図



関数

- double `calculateLR2XYZ` (double position3D[3], Data_2D posL, Data_2D posR, CameraParam *camParamL, CameraParam *camParamR)
- double `calculatePlane3D` (double plane3D[4], const double l11[3], const double l12[3], const double l21[3], const double l22[3], const CameraParam *camParamL, const CameraParam *camParamR)
- void `projectXYZ2LR` (Data_2D *pos2D, double position[3], CameraParam *cameraParam)

3次元点の2次元画像上への投影点座標を求める

- bool `set_circle_to_OldFeature3D` (const std::vector< [CircleCandidate](#) > &candidates, [Features3D](#) *feature)

ステレオ処理結果を旧 3 次元特徴構造体へセットする

- bool `set_vertex_to_OldFeature3D` (const std::vector< [VertexCandidate](#) > &candidates, [Features3D](#) *feature)

頂点のステレオ処理結果を 3 次元特徴構造体へセットする

7.58.1 説明

ステレオ処理関連関数

日付:

\$Date:: 2011-09-15 18:11:51 +0900 # \$

7.58.2 関数

7.58.2.1 double `calculateLR2XYZ` (double *position3D*[3], *Data_2D posL*, *Data_2D posR*, *CameraParam * camParamL*, *CameraParam * camParamR*)

ステレオ対応点から 3 次元座標を計算する 戻り値 : 復元誤差 = 2 つの視線 (エピポーラ線) 間の距離

7.58.2.2 double `calculatePlane3D` (double *plane3D*[4], const double *l11*[3], const double *l12*[3], const double *l21*[3], const double *l22*[3], const *CameraParam * camParamL*, const *CameraParam * camParamR*)

7.58.2.3 void `projectXYZ2LR` (*Data_2D * pos2D*, double *position*[3], *CameraParam * cameraParam*)

3 次元点の 2 次元画像上への投影点座標を求める

7.58.2.4 bool `set_circle_to_OldFeature3D` (const std::vector< [CircleCandidate](#) > & *candidates*, [Features3D](#) * *feature*)

ステレオ処理結果を旧 3 次元特徴構造体へセットする

7.58.2.5 `bool set_vertex_to_OldFeature3D (const std::vector< VertexCandidate
> & candidates, Features3D *feature)`

頂点のステレオ処理結果を 3 次元特徴構造体へセットする

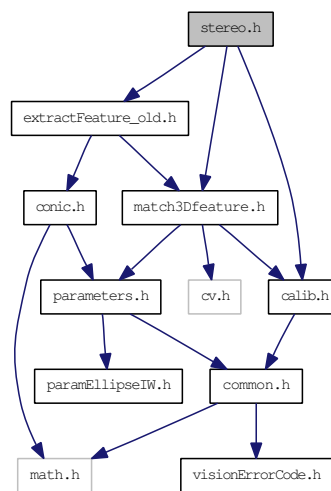
7.59 stereo.h

ステレオ処理関連関数 `#include "calib.h"`

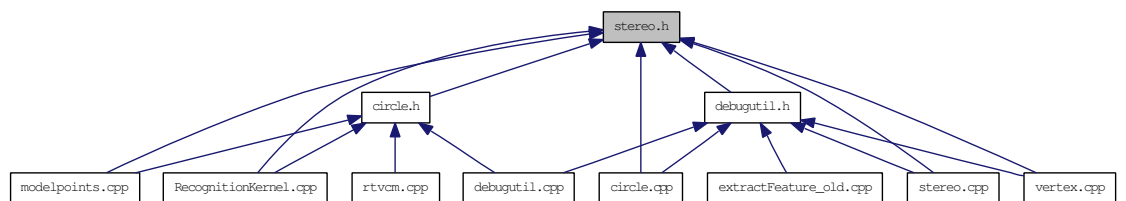
`#include "extractFeature_old.h"`

`#include "match3Dfeature.h"`

stereo.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



データ構造

- struct [StereoCalib](#)
ステレオカメラキャリブレーションデータ
- struct [VertexCandidate](#)
三次元頂点特徴候補データ

- struct [CircleCandidate](#)
三次元円特徴候補データ

関数

- double [calculateLR2XYZ](#) (double position3D[3], [Data_2D](#) posL, [Data_2D](#) posR, [CameraParam](#) *camParamL, [CameraParam](#) *camParamR)
- double [calculatePlane3D](#) (double plane3D[4], const double l11[3], const double l12[3], const double l21[3], const double l22[3], const [CameraParam](#) *camParamL, const [CameraParam](#) *camParamR)
- void [projectXYZ2LR](#) ([Data_2D](#) *pos2D, double position[3], [CameraParam](#) *cameraParam)
3次元点の2次元画像上への投影点座標を求める
- bool [set_circle_to_OldFeature3D](#) (const std::vector< [CircleCandidate](#) > &candidates, [Features3D](#) *feature)
ステレオ処理結果を旧3次元特徴構造体へセットする
- bool [set_vertex_to_OldFeature3D](#) (const std::vector< [VertexCandidate](#) > &candidates, [Features3D](#) *feature)
頂点のステレオ処理結果を3次元特徴構造体へセットする

7.59.1 説明

ステレオ処理関連関数

日付:

\$Date:: 2011-09-16 08:26:14 +0900 # \$

7.59.2 関数

7.59.2.1 double [calculateLR2XYZ](#) (double *position3D*[3], [Data_2D](#) *posL*, [Data_2D](#) *posR*, [CameraParam](#) * *camParamL*, [CameraParam](#) * *camParamR*)

ステレオ対応点から3次元座標を計算する 戻り値: 復元誤差 = 2つの視線 (エピポーラ線) 間の距離

7.59.2.2 `double calculatePlane3D (double plane3D[4], const double l11[3],
const double l12[3], const double l21[3], const double l22[3], const
CameraParam * camParamL, const CameraParam * camParamR)`

7.59.2.3 `void projectXYZ2LR (Data_2D * pos2D, double position[3],
CameraParam * cameraParam)`

3次元点の2次元画像上への投影点座標を求める

7.59.2.4 `bool set_circle_to_OldFeature3D (const std::vector< CircleCandidate
> & candidates, Features3D * feature)`

ステレオ処理結果を旧3次元特徴構造体へセットする

7.59.2.5 `bool set_vertex_to_OldFeature3D (const std::vector< VertexCandidate
> & candidates, Features3D * feature)`

頂点のステレオ処理結果を3次元特徴構造体へセットする

7.60 vectorutil.cpp

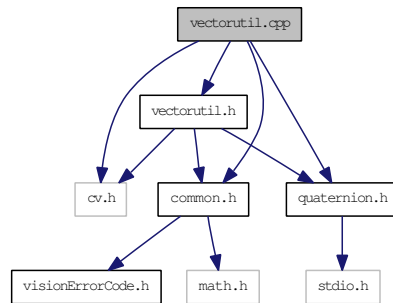
ベクトル処理、行列処理 ユーティリティ関数 `#include <cv.h>`

```
#include "vectorutil.h"
```

```
#include "common.h"
```

```
#include "quaternion.h"
```

vectorutil.cpp のインクルード依存関係図



関数

- `int isZero` (double value)
- `void copyV2` (V2 in, V2 out)
- `void mulV2S` (V2 in, const double s, V2 out)
- `double getNormV2` (V2 in)
- `void normalizeV2` (V2 in, V2 out)
- `double getDistanceV2` (V2 in1, V2 in2)
- `double getAngle2D` (double vec1[2], double vec2[2])
- `int eigenM22` (double e[2], double ev[2][2], double m[2][2], const double rankDiag)
- `void zeroV3` (V3 out)
- `void copyV3` (V3 in, V3 out)
- `void addV3` (V3 in1, V3 in2, V3 out)
- `void subV3` (V3 in1, V3 in2, V3 out)
- `void mulV3S` (const double s, V3 in, V3 out)
- `void normalizeV3` (V3 in, V3 out)
- `double getNormV3` (V3 in)
- `double getInnerProductV3` (V3 in1, V3 in2)
- `void getCrossProductV3` (V3 in1, V3 in2, V3 out)
- `double getDistanceV3` (V3 in1, V3 in2)
- `void subM33` (M33 in1, M33 in2, M33 out)

- void `transposeM33` (`M33` in, `M33` out)
- void `mulM33` (`M33` in1, `M33` in2, `M33` out)
- void `mulM33V3` (`M33` in1, `V3` in2, `V3` out)
- int `inverseM33` (`M33` in, `M33` out)
- void `getDirectionVector` (double tail[3], double head[3], double data[3], CvMat *vec)
- int `getOrthogonalDir` (double axis[3], double normal[3], double dir[3])
- int `getOrthogonalDir` (CvMat *axis, CvMat *normal, CvMat *dir)
- void `quaternion_rotation` (`quaternion_t` q, const double radian, double axis[3])

7.60.1 説明

ベクトル処理、行列処理 ユーティリティ関数

日付:

\$Date:: 2011-10-21 14:49:57 +0900 #

7.60.2 関数

7.60.2.1 void addV3 (`V3` in1, `V3` in2, `V3` out)

7.60.2.2 void copyV2 (`V2` in, `V2` out)

7.60.2.3 void copyV3 (`V3` in, `V3` out)

7.60.2.4 int eigenM22 (double *e*[2], double *ev*[2][2], double *m*[2][2], const double *rankDiag*)

7.60.2.5 double getAngle2D (double *vec1*[2], double *vec2*[2])

7.60.2.6 void getCrossProductV3 (V3 *in1*, V3 *in2*, V3 *out*)

7.60.2.7 void getDirectionVector (double *tail*[3], double *head*[3], double *data*[3], CvMat * *vec*)

7.60.2.8 double getDistanceV2 (V2 *in1*, V2 *in2*)

7.60.2.9 double getDistanceV3 (V3 *in1*, V3 *in2*)

7.60.2.10 double getInnerProductV3 (V3 *in1*, V3 *in2*)

7.60.2.11 double getNormV2 (V2 *in*)

7.60.2.12 double getNormV3 (V3 *in*)

7.60.2.13 int getOrthogonalDir (CvMat * *axis*, CvMat * *normal*, CvMat * *dir*)

7.60.2.14 int getOrthogonalDir (double *axis*[3], double *normal*[3], double *dir*[3])

7.60.2.15 int inverseM33 (M33 *in*, M33 *out*)

7.60.2.16 `int isZero (double value)`

7.60.2.17 `void mulM33 (M33 in1, M33 in2, M33 out)`

7.60.2.18 `void mulM33V3 (M33 in1, V3 in2, V3 out)`

7.60.2.19 `void mulV2S (V2 in, const double s, V2 out)`

7.60.2.20 `void mulV3S (const double s, V3 in, V3 out)`

7.60.2.21 `void normalizeV2 (V2 in, V2 out)`

7.60.2.22 `void normalizeV3 (V3 in, V3 out)`

7.60.2.23 `void quaternion_rotation (quaternion_t q, const double radian,
double axis[3])`

7.60.2.24 `void subM33 (M33 in1, M33 in2, M33 out)`

7.60.2.25 `void subV3 (V3 in1, V3 in2, V3 out)`

7.60.2.26 void transposeM33 (*M33 in*, *M33 out*)

7.60.2.27 void zeroV3 (*V3 out*)

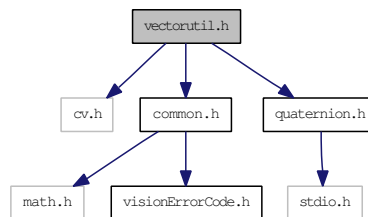
7.61 vectorutil.h

ベクトル処理、行列処理 ユーティリティ関数 `#include <cv.h>`

```
#include "common.h"
```

```
#include "quaternion.h"
```

vectorutil.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



型定義

- typedef double **V2** [2]
2 dimensional vector
- typedef double **V3** [3]
3 dimensional vector
- typedef double **M33** [3][3]
3x3 matrix

関数

- int **isZero** (double value)
- void **copyV2** (V2 in, V2 out)
- void **mulV2S** (V2 in, const double s, V2 out)
- void **normalizeV2** (V2 in, V2 out)
- double **getNormV2** (V2 in)

- double `getDistanceV2` (`V2` in1, `V2` in2)
- double `getAngle2D` (double `vec1`[2], double `vec2`[2])
- int `eigenM22` (double `e`[2], double `ev`[2][2], double `m`[2][2], const double `rankDiag`)
- void `copyV3` (`V3` in, `V3` out)
- void `addV3` (`V3` in1, `V3` in2, `V3` out)
- void `subV3` (`V3` in1, `V3` in2, `V3` out)
- void `mulV3S` (const double `s`, `V3` in, `V3` out)
- void `normalizeV3` (`V3` in, `V3` out)
- double `getInnerProductV3` (`V3` in1, `V3` in2)
- double `getNormV3` (`V3` in)
- void `getCrossProductV3` (`V3` in1, `V3` in2, `V3` out)
- double `getDistanceV3` (`V3` in1, `V3` in2)
- void `subM33` (`M33` in1, `M33` in2, `M33` out)
- void `transposeM33` (`M33` in, `M33` out)
- void `mulM33` (`M33` in1, `M33` in2, `M33` out)
- void `mulM33V3` (`M33` in1, `V3` in2, `V3` out)
- int `inverseM33` (`M33` in, `M33` out)
- void `getDirectionVector` (double `tail`[3], double `head`[3], double `data`[3], `CvMat` *`vec`)
- int `getOrthogonalDir` (double `axis`[3], double `normal`[3], double `dir`[3])
- int `getOrthogonalDir` (`CvMat` *`axis`, `CvMat` *`normal`, `CvMat` *`dir`)
- void `quaternion_rotation` (`quaternion_t` `q`, const double `radian`, double `axis`[3])

7.61.1 説明

ベクトル処理、行列処理 ユーティリティ関数

日付:

\$Date:: 2011-10-21 11:50:52 +0900 #\$

7.61.2 型定義

7.61.2.1 typedef double M33[3][3]

3x3 matrix

7.61.2.2 typedef double V2[2]

2 dimensional vector

7.61.2.3 typedef double V3[3]

3 dimensional vector

7.61.3 関数

7.61.3.1 void addV3 (V3 *in1*, V3 *in2*, V3 *out*)

7.61.3.2 void copyV2 (V2 *in*, V2 *out*)

7.61.3.3 void copyV3 (V3 *in*, V3 *out*)

7.61.3.4 int eigenM22 (double *e*[2], double *ev*[2][2], double *m*[2][2], const double *rankDiag*)

7.61.3.5 double getAngle2D (double *vec1*[2], double *vec2*[2])

7.61.3.6 void getCrossProductV3 (V3 *in1*, V3 *in2*, V3 *out*)

7.61.3.7 void getDirectionVector (double *tail*[3], double *head*[3], double *data*[3], CvMat * *vec*)

7.61.3.8 double getDistanceV2 (V2 *in1*, V2 *in2*)

7.61.3.9 double getDistanceV3 (V3 *in1*, V3 *in2*)

7.61.3.10 double getInnerProductV3 (V3 *in1*, V3 *in2*)

7.61.3.11 double getNormV2 (V2 *in*)

7.61.3.12 double getNormV3 (V3 *in*)

7.61.3.13 int getOrthogonalDir (CvMat * *axis*, CvMat * *normal*, CvMat * *dir*)

7.61.3.14 int getOrthogonalDir (double *axis*[3], double *normal*[3], double *dir*[3])

7.61.3.15 int inverseM33 (M33 *in*, M33 *out*)

7.61.3.16 int isZero (double *value*)

7.61.3.17 void mulM33 (M33 *in1*, M33 *in2*, M33 *out*)

7.61.3.18 void mulM33V3 (M33 *in1*, V3 *in2*, V3 *out*)

7.61.3.19 void mulV2S (*V2 in*, const double *s*, *V2 out*)

7.61.3.20 void mulV3S (const double *s*, *V3 in*, *V3 out*)

7.61.3.21 void normalizeV2 (*V2 in*, *V2 out*)

7.61.3.22 void normalizeV3 (*V3 in*, *V3 out*)

7.61.3.23 void quaternion_rotation (quaternion_t *q*, const double *radian*,
double *axis*[3])

7.61.3.24 void subM33 (*M33 in1*, *M33 in2*, *M33 out*)

7.61.3.25 void subV3 (*V3 in1*, *V3 in2*, *V3 out*)

7.61.3.26 void transposeM33 (*M33 in*, *M33 out*)

7.62 vertex.cpp

3次元頂点特徴生成関連関数 `#include <iostream>`

`#include "stereo.h"`

`#include "vectorutil.h"`

`#include "debugutil.h"`

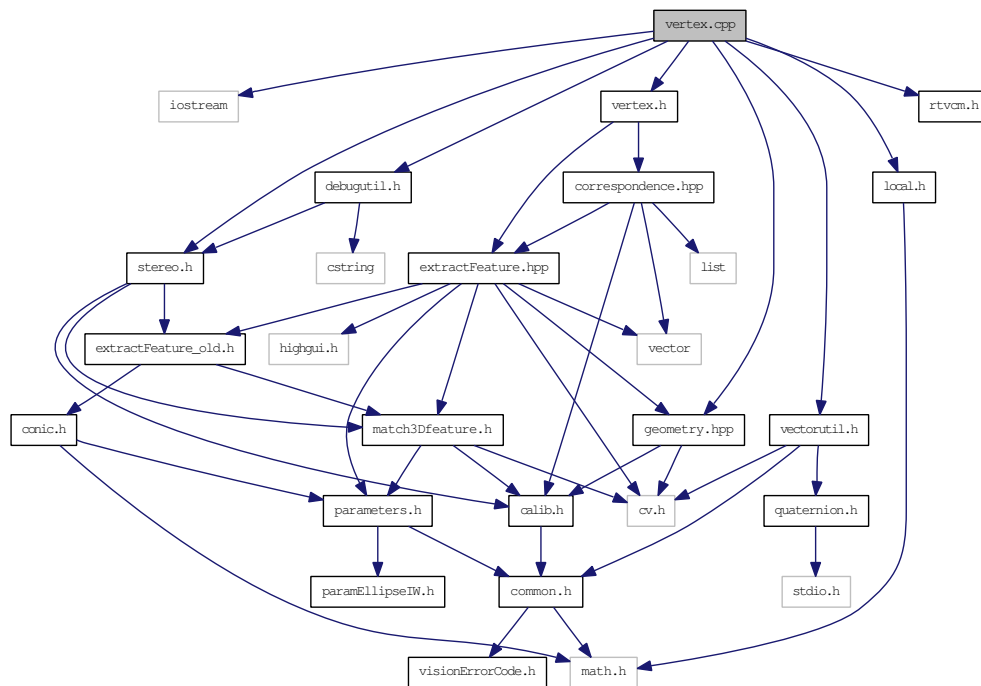
`#include "vertex.h"`

`#include "rtvcm.h"`

`#include "local.h"`

`#include "geometry.hpp"`

vertex.cpp のインクルード依存関係図



関数

- `int isValidPixelPosition` (double col, double row, const `Parameters` ¶meters)
- `void reconstruct_hyperbola_to_vertex3D` (const `std::vector< const ovgr::Features2D * >` feature, const `ovgr::CorrespondingSet` &cs, const

```
CameraParam *camParam[3], const unsigned char *edge[3], Features3D  
*scene, const Parameters &parameters)
```

二次元頂点データから三次元頂点データを生成

7.62.1 説明

3次元頂点特徴生成関連関数

日付:

\$Date:: 2011-10-21 14:49:57 +0900 #

7.62.2 関数

7.62.2.1 `int isValidPixelPosition (double col, double row, const Parameters &
parameters) [inline]`

7.62.2.2 `void reconstruct_hyperbola_to_vertex3D (const std::vector< const
ovgr::Features2D * > feature, const ovgr::CorrespondingSet & cs,
const CameraParam * camParam[3], const unsigned char * edge[3],
Features3D * scene, const Parameters & parameters)`

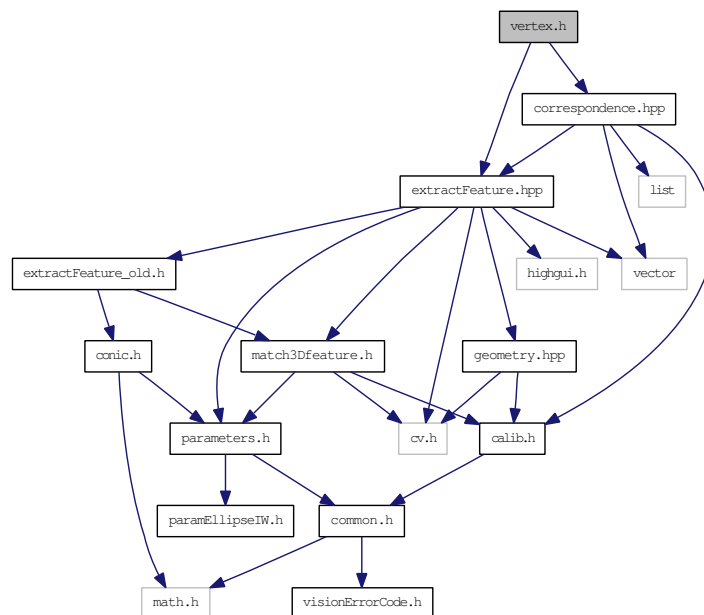
二次元頂点データから三次元頂点データを生成

7.63 vertex.h

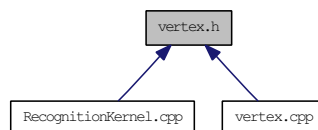
3次元頂点特徴生成関連関数 `#include "extractFeature.hpp"`

`#include "correspondence.hpp"`

vertex.h のインクルード依存関係図



このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



関数

- void `reconstruct_hyperbola_to_vertex3D` (const std::vector< const `ovgr::Features2D` * > feature, const `ovgr::CorrespondingSet` &cs, const `CameraParam` *camParam[3], const unsigned char *edge[3], `Features3D` *scene, const `Parameters` ¶meters)

二次元頂点データから三次元頂点データを生成

7.63.1 説明

3次元頂点特徴生成関連関数

日付:

\$Date:: 2011-09-15 17:51:29 +0900 #

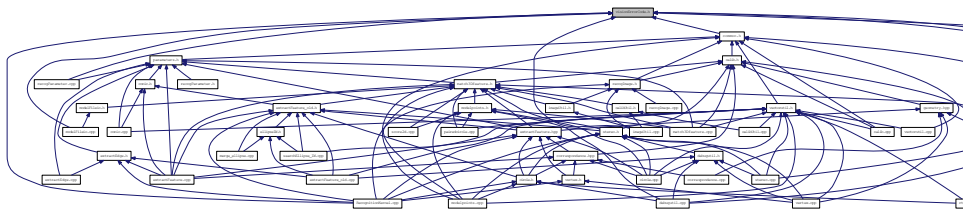
7.63.2 関数

7.63.2.1 void reconstruct_hyperbola_to_vertex3D (const std::vector< const ovgr::Features2D * > *feature*, const ovgr::CorrespondingSet & *cs*, const CameraParam * *camParam*[3], const unsigned char * *edge*[3], Features3D * *scene*, const Parameters & *parameters*)

二次元頂点データから三次元頂点データを生成

7.64 visionErrorCode.h

返り値の定義このグラフは、どのファイルから直接、間接的にインクルードされているかを示しています。



列挙型

```
enum VisionErrorCode {
    VISION_PARAM_ERROR = -1,  VISION_MALLOC_ERROR = -2,
    VISION_FILE_OPEN_ERROR = -3, VISION_FILE_FORMAT_ERROR
    = -4,
    VISION_ILLEGAL_IMAGE_SIZE = -101, VISION_INPUT_NOIMAGE
    = -102, VISION_DIFF_IMAGE_SIZE = -103,
    VISION_DIFF_IMAGE_COLOR_MODEL = -104,
    VISION_NO_MODEL_FILE = -105 }
```

7.64.1 説明

返り値の定義

7.64.2 列挙型

7.64.2.1 enum VisionErrorCode

列挙型の値:

VISION_PARAM_ERROR 関数の引数エラー
VISION_MALLOC_ERROR メモリ領域確保エラー
VISION_FILE_OPEN_ERROR ファイルオープンエラー
VISION_FILE_FORMAT_ERROR ファイルフォーマットエラー
VISION_ILLEGAL_IMAGE_SIZE 入力画像の幅または高さが 0

VISION_INPUT_NOIMAGE 入力された画像が 1 枚以下
VISION_DIFF_IMAGE_SIZE 入力された画像のサイズが同一でない
VISION_DIFF_IMAGE_COLOR_MODEL 入力された画像のカラーモデル
が同一でない
VISION_NO_MODEL_FILE 入力されたモデル番号に該当するモデルデー
タがない