

libopenvgr
0.8.1

作成 : Doxygen 1.6.1

Tue Aug 9 17:16:17 2011

Contents

1	データ構造索引	1
1.1	データ構造	1
2	ファイル索引	3
2.1	ファイル一覧	3
3	データ構造	5
3.1	クラス <code>_impl_CameraCaptureService</code>	5
3.1.1	コンストラクタとデストラクタ	5
3.1.1.1	<code>~_impl_CameraCaptureService</code>	5
3.1.2	関数	5
3.1.2.1	<code>_dispatch</code>	5
3.1.2.2	<code>take_one_frame</code>	6
3.2	クラス <code>_objref_CameraCaptureService</code>	7
3.2.1	コンストラクタとデストラクタ	7
3.2.1.1	<code>_objref_CameraCaptureService</code>	7
3.2.1.2	<code>_objref_CameraCaptureService</code>	7
3.2.1.3	<code>~_objref_CameraCaptureService</code>	7
3.2.2	関数	7
3.2.2.1	<code>take_one_frame</code>	8
3.2.3	フレンドと関連する関数	8
3.2.3.1	<code>CameraCaptureService</code>	8
3.3	クラス <code>_pof_CameraCaptureService</code>	9
3.3.1	コンストラクタとデストラクタ	9
3.3.1.1	<code>_pof_CameraCaptureService</code>	9
3.3.1.2	<code>~_pof_CameraCaptureService</code>	9

3.3.2	関数	9
3.3.2.1	is_a	9
3.3.2.2	newObjRef	9
3.4	構造体 _recogImage	10
3.4.1	構造体	10
3.4.1.1	bytePerPixel	10
3.4.1.2	colsize	10
3.4.1.3	pixel	10
3.4.1.4	rowsize	10
3.5	構造体 CalibParam	11
3.5.1	説明	12
3.5.2	構造体	12
3.5.2.1	CameraL	12
3.5.2.2	CameraR	12
3.5.2.3	CameraV	12
3.5.2.4	colsize	12
3.5.2.5	numOfCameras	12
3.5.2.6	rowsize	12
3.6	クラス CameraCaptureService	13
3.6.1	型定義	13
3.6.1.1	_ptr_type	13
3.6.1.2	_var_type	13
3.6.2	コンストラクタとデストラクタ	13
3.6.2.1	~CameraCaptureService	14
3.6.3	関数	14
3.6.3.1	_duplicate	14
3.6.3.2	_marshalObjRef	14
3.6.3.3	_narrow	14
3.6.3.4	_nil	14
3.6.3.5	_this	14
3.6.3.6	_unchecked_narrow	14
3.6.3.7	_unmarshalObjRef	14
3.6.4	構造体	14
3.6.4.1	_PD_repoId	15

3.7	クラス CameraCaptureService_Helper	16
3.7.1	型定義	16
3.7.1.1	_ptr_type	16
3.7.2	関数	16
3.7.2.1	_nil	16
3.7.2.2	duplicate	16
3.7.2.3	is_nil	16
3.7.2.4	marshalObjRef	17
3.7.2.5	release	17
3.7.2.6	unmarshalObjRef	17
3.8	構造体 CameraImage	18
3.8.1	型定義	18
3.8.1.1	_var_type	18
3.8.2	関数	18
3.8.2.1	operator<<=	19
3.8.2.2	operator>>=	19
3.8.3	構造体	19
3.8.3.1	captured_time	19
3.8.3.2	extrinsic	19
3.8.3.3	image	19
3.8.3.4	intrinsic	19
3.9	構造体 CameraIntrinsicParameter	20
3.9.1	型定義	20
3.9.1.1	_ORL_matrix_element	20
3.9.1.2	_distortion_coefficient_seq	20
3.9.1.3	_matrix_element_slice	21
3.9.1.4	_var_type	21
3.9.2	関数	21
3.9.2.1	operator<<=	21
3.9.2.2	operator>>=	21
3.9.3	構造体	21
3.9.3.1	distortion_coefficient	21
3.9.3.2	matrix_element	21
3.10	構造体 CameraParam	22

3.10.1	説明	23
3.10.2	構造体	23
3.10.2.1	cx	23
3.10.2.2	cy	23
3.10.2.3	Distortion	23
3.10.2.4	fx	23
3.10.2.5	fy	23
3.10.2.6	intrinsicMatrix	23
3.10.2.7	Position	23
3.10.2.8	Rotation	24
3.10.2.9	rRotation	24
3.10.2.10	Translation	24
3.11	構造体 Circle	25
3.11.1	説明	26
3.11.2	構造体	26
3.11.2.1	center	26
3.11.2.2	label	26
3.11.2.3	n	26
3.11.2.4	normal	26
3.11.2.5	numOfTracePoints	26
3.11.2.6	orientation	26
3.11.2.7	projected	26
3.11.2.8	radius	27
3.11.2.9	side	27
3.11.2.10	tracepoints	27
3.11.2.11	transformed	27
3.12	構造体 CircleCandidate	28
3.12.1	説明	28
3.12.2	構造体	28
3.12.2.1	center	28
3.12.2.2	normal	28
3.12.2.3	radius	28
3.12.2.4	valid	29
3.13	構造体 Data_2D	30

3.13.1 構造体	30
3.13.1.1 col	30
3.13.1.2 row	30
3.14 構造体 DistortionParam	31
3.14.1 説明	31
3.14.2 構造体	31
3.14.2.1 k1	31
3.14.2.2 k2	31
3.14.2.3 k3	31
3.14.2.4 p1	32
3.14.2.5 p2	32
3.15 構造体 EllipseGroup	33
3.15.1 説明	33
3.15.2 構造体	33
3.15.2.1 groupCenter	33
3.15.2.2 groupNums	33
3.15.2.3 nCurrNum	33
3.16 構造体 Feature2D	34
3.16.1 説明	35
3.16.2 構造体	35
3.16.2.1 all	35
3.16.2.2 axis	35
3.16.2.3 center	36
3.16.2.4 coef	36
3.16.2.5 direction	36
3.16.2.6 end	36
3.16.2.7 endPoint	36
3.16.2.8 endSPoint	36
3.16.2.9 error	36
3.16.2.10 ev	36
3.16.2.11 lineAngle	36
3.16.2.12 lineLength	36
3.16.2.13 lineLength1	37
3.16.2.14 lineLength2	37

3.16.2.15	middleSPoint	37
3.16.2.16	nPoints	37
3.16.2.17	nTrack	37
3.16.2.18	start	37
3.16.2.19	startPoint	37
3.16.2.20	startSPoint	37
3.16.2.21	type	37
3.17	構造体 Features2D	38
3.17.1	説明	38
3.17.2	構造体	38
3.17.2.1	feature	39
3.17.2.2	nAlloc	39
3.17.2.3	nFeature	39
3.17.2.4	nTrack	39
3.17.2.5	track	39
3.18	構造体 Features3D	40
3.18.1	説明	41
3.18.2	構造体	41
3.18.2.1	calib	41
3.18.2.2	Circles	41
3.18.2.3	edge	41
3.18.2.4	image	41
3.18.2.5	numOfCircles	42
3.18.2.6	numOfVertices	42
3.18.2.7	pointCounts	42
3.18.2.8	trace_edge	42
3.18.2.9	trace_pdist	42
3.18.2.10	trace_search	42
3.18.2.11	traceCounts	42
3.18.2.12	Vertices	42
3.19	構造体 ImageData	43
3.19.1	型定義	43
3.19.1.1	_raw_data_seq	43
3.19.1.2	_var_type	43

3.19.2 関数	43
3.19.2.1 operator<<=	43
3.19.2.2 operator>>=	44
3.19.3 構造体	44
3.19.3.1 format	44
3.19.3.2 height	44
3.19.3.3 raw_data	44
3.19.3.4 width	44
3.20 クラス Mat44_copyHelper	45
3.20.1 関数	45
3.20.1.1 alloc	45
3.20.1.2 dup	45
3.20.1.3 free	45
3.21 構造体 Match3Dresults	46
3.21.1 説明	46
3.21.2 構造体	46
3.21.2.1 error	46
3.21.2.2 numOfResults	47
3.21.2.3 Results	47
3.22 構造体 MatchResult	48
3.22.1 説明	48
3.22.2 構造体	48
3.22.2.1 mat	48
3.22.2.2 model	49
3.22.2.3 n	49
3.22.2.4 scene	49
3.22.2.5 score	49
3.22.2.6 type	49
3.22.2.7 vec	49
3.23 構造体 ModelFileInfo	50
3.23.1 説明	50
3.23.2 構造体	50
3.23.2.1 model	50
3.23.2.2 modelNum	51

3.24	構造体 ModelFileInfoNode	52
3.24.1	説明	52
3.24.2	構造体	52
3.24.2.1	id	52
3.24.2.2	path	52
3.25	構造体 MultiCameraImage	53
3.25.1	型定義	53
3.25.1.1	_image_seq_seq	53
3.25.1.2	_var_type	53
3.25.2	関数	53
3.25.2.1	operator<=<=	53
3.25.2.2	operator>>=	54
3.25.3	構造体	54
3.25.3.1	camera_set_id	54
3.25.3.2	image_seq	54
3.26	構造体 P2D	55
3.26.1	説明	55
3.26.2	構造体	55
3.26.2.1	colrow	55
3.27	構造体 P3D	56
3.27.1	説明	56
3.27.2	構造体	56
3.27.2.1	xyz	56
3.28	構造体 Parameters	57
3.28.1	説明	58
3.28.2	構造体	58
3.28.2.1	colsize	58
3.28.2.2	dbgdisp	58
3.28.2.3	dbgimag	58
3.28.2.4	dbgtxt	58
3.28.2.5	feature2D	58
3.28.2.6	imgsize	58
3.28.2.7	match	58
3.28.2.8	outputCandNum	58

3.28.2.9	pairing	58
3.28.2.10	rowsize	59
3.28.2.11	stereo	59
3.29	構造体 ParametersFeature2D	60
3.29.1	説明	61
3.29.2	構造体	61
3.29.2.1	edgeDetectFunction	61
3.29.2.2	edgeStrength	62
3.29.2.3	id	62
3.29.2.4	max_distance_ellipse_grouping	62
3.29.2.5	max_distance_ellipse_pairing	62
3.29.2.6	max_distance_end_points	62
3.29.2.7	max_flatness_ellipse	62
3.29.2.8	max_length_delete_line	62
3.29.2.9	max_length_ellipse_axisL	62
3.29.2.10	maxErrorofConicFit	62
3.29.2.11	maxErrorofLineFit	62
3.29.2.12	min_distance_ellipse_pairing	62
3.29.2.13	min_filling_ellipse	63
3.29.2.14	min_length_ellipse_axis	63
3.29.2.15	min_length_ellipse_axisL	63
3.29.2.16	min_length_ellipse_axisS	63
3.29.2.17	min_length_hyperbola_data	63
3.29.2.18	min_length_hyperbola_vector	63
3.29.2.19	min_length_line	63
3.29.2.20	min_radian_hyperbola	63
3.29.2.21	minFragment	63
3.29.2.22	overlapRatioCircle	63
3.29.2.23	overlapRatioLine	64
3.30	構造体 ParametersMatch	65
3.30.1	説明	65
3.30.2	構造体	65
3.30.2.1	edge	65
3.30.2.2	interval	65

3.30.2.3	pdist	66
3.30.2.4	search	66
3.30.2.5	tolerance1	66
3.30.2.6	tolerance2	66
3.31	構造体 ParmetersStereo	67
3.31.1	説明	67
3.31.2	構造体	67
3.31.2.1	amax	68
3.31.2.2	amin	68
3.31.2.3	depf	68
3.31.2.4	depn	68
3.31.2.5	ethr	68
3.31.2.6	lmax	68
3.31.2.7	lmin	68
3.31.2.8	ndif	68
3.31.2.9	rdif	68
3.32	構造体 RTVCM_Box	69
3.32.1	説明	69
3.32.2	構造体	69
3.32.2.1	n	69
3.32.2.2	nVertex	70
3.32.2.3	reserved	70
3.32.2.4	Rotate	70
3.32.2.5	Trans	70
3.32.2.6	x	70
3.32.2.7	y	70
3.32.2.8	z	70
3.33	構造体 RTVCM_Circle	71
3.33.1	説明	71
3.33.2	構造体	71
3.33.2.1	center	71
3.33.2.2	n	71
3.33.2.3	ncyliner	72
3.33.2.4	normal	72

3.33.2.5	radius	72
3.33.2.6	reserved	72
3.34	構造体 RTVCM_Cylinder	73
3.34.1	説明	73
3.34.2	構造体	73
3.34.2.1	height	73
3.34.2.2	n	74
3.34.2.3	nCircle	74
3.34.2.4	radius	74
3.34.2.5	reserved	74
3.34.2.6	Rotate	74
3.34.2.7	Trans	74
3.35	構造体 RTVCM_Vertex	75
3.35.1	説明	75
3.35.2	構造体	75
3.35.2.1	angle	75
3.35.2.2	endpoint1	76
3.35.2.3	endpoint2	76
3.35.2.4	n	76
3.35.2.5	nbox	76
3.35.2.6	position	76
3.35.2.7	reserved	76
3.36	構造体 RTVertexCircleModel	77
3.36.1	説明	78
3.36.2	構造体	78
3.36.2.1	box	78
3.36.2.2	circle	78
3.36.2.3	cylinder	78
3.36.2.4	depth	79
3.36.2.5	gravity	79
3.36.2.6	height	79
3.36.2.7	label	79
3.36.2.8	n	79
3.36.2.9	nbox	79

3.36.2.10	ncircle	79
3.36.2.11	ncylinder	79
3.36.2.12	nvertex	79
3.36.2.13	radius	79
3.36.2.14	reserved	80
3.36.2.15	vertex	80
3.36.2.16	width	80
3.37	構造体 StereoCalib	81
3.37.1	説明	81
3.37.2	構造体	81
3.37.2.1	baselineLR	81
3.37.2.2	baselineLV	81
3.37.2.3	baselineRV	82
3.37.2.4	height	82
3.37.2.5	numOfCameras	82
3.37.2.6	width	82
3.38	構造体 StereoConic	83
3.38.1	説明	84
3.38.2	構造体	84
3.38.2.1	center	84
3.38.2.2	circle	84
3.38.2.3	error	84
3.38.2.4	featureL	84
3.38.2.5	featureR	84
3.38.2.6	type	84
3.38.2.7	valid	84
3.38.2.8	vertex	84
3.38.2.9	work	85
3.39	構造体 StereoData	86
3.39.1	説明	86
3.39.2	構造体	86
3.39.2.1	conics	87
3.39.2.2	numOfconics	87
3.40	構造体 TimedCameraImage	88

3.40.1	型定義	88
3.40.1.1	_var_type	89
3.40.2	関数	89
3.40.2.1	operator<<=	89
3.40.2.2	operator>>=	89
3.40.3	構造体	89
3.40.3.1	data	89
3.40.3.2	error_code	89
3.40.3.3	tm	89
3.41	構造体 TimedMultiCameraImage	90
3.41.1	型定義	90
3.41.1.1	_var_type	90
3.41.2	関数	90
3.41.2.1	operator<<=	91
3.41.2.2	operator>>=	91
3.41.3	構造体	91
3.41.3.1	data	91
3.41.3.2	error_code	91
3.41.3.3	tm	91
3.42	構造体 Trace	92
3.42.1	説明	92
3.42.2	構造体	92
3.42.2.1	colrow	92
3.42.2.2	direction	93
3.42.2.3	edge	93
3.42.2.4	label	93
3.42.2.5	peakcr	93
3.42.2.6	search	93
3.42.2.7	weight	93
3.42.2.8	xyz	93
3.43	構造体 Track	94
3.43.1	説明	94
3.43.2	構造体	94
3.43.2.1	nPoint	94

3.43.2.2	offset	94
3.43.2.3	Point	94
3.44	クラス Vec3_copyHelper	95
3.44.1	関数	95
3.44.1.1	alloc	95
3.44.1.2	dup	95
3.44.1.3	free	95
3.45	構造体 Vertex	96
3.45.1	説明	97
3.45.2	構造体	97
3.45.2.1	angle	97
3.45.2.2	direction1	97
3.45.2.3	direction2	97
3.45.2.4	endpoint1	97
3.45.2.5	endpoint2	98
3.45.2.6	label	98
3.45.2.7	n	98
3.45.2.8	numOfTracePoints	98
3.45.2.9	orientation	98
3.45.2.10	position	98
3.45.2.11	projected	98
3.45.2.12	side	98
3.45.2.13	tracepoints	98
3.45.2.14	transformed	98
3.46	構造体 VertexCandidate	99
3.46.1	説明	100
3.46.2	構造体	100
3.46.2.1	angle	100
3.46.2.2	endpoint1	100
3.46.2.3	endpoint2	100
3.46.2.4	len1	100
3.46.2.5	len2	100
3.46.2.6	n1	100
3.46.2.7	n2	100

3.46.2.8	n3	100
3.46.2.9	position	100
3.46.2.10	valid	101
3.46.2.11	vector1	101
3.46.2.12	vector2	101
4	ファイル	103
4.1	calib.cpp	103
4.1.1	説明	104
4.1.2	関数	104
4.1.2.1	backprojectPoint	104
4.1.2.2	distortPosition	104
4.1.2.3	projectPoint	104
4.1.2.4	undistortPosition	104
4.2	calib.h	105
4.2.1	説明	106
4.2.2	関数	106
4.2.2.1	backprojectPoint	106
4.2.2.2	distortPosition	106
4.2.2.3	projectPoint	106
4.2.2.4	undistortPosition	106
4.3	calibUtil.cpp	107
4.3.1	関数	107
4.3.1.1	setCalibFromCameraImage	107
4.4	calibUtil.h	108
4.4.1	説明	108
4.4.2	関数	108
4.4.2.1	setCalibFromCameraImage	109
4.5	circle.cpp	110
4.5.1	説明	110
4.5.2	関数	111
4.5.2.1	EllipseToCircle	111
4.6	circle.h	112
4.6.1	説明	112

4.6.2	関数	112
4.6.2.1	EllipseToCircle	113
4.7	common.h	114
4.7.1	説明	115
4.7.2	マクロ定義	115
4.7.2.1	INVISIBLE	115
4.7.2.2	VISIBLE	115
4.7.2.3	VISION_EPS	115
4.7.3	列挙型	115
4.7.3.1	StereoPairing	115
4.8	conic.cpp	116
4.8.1	説明	116
4.8.2	関数	117
4.8.2.1	addConicSum	117
4.8.2.2	clearConicSum	117
4.8.2.3	distanceConic	117
4.8.2.4	fitConic	117
4.8.2.5	fitConicAny	117
4.8.2.6	getConicProperty	117
4.8.2.7	getConicType	117
4.8.2.8	subConicSum	117
4.9	conic.h	118
4.9.1	説明	119
4.9.2	列挙型	119
4.9.2.1	ConicType	119
4.9.3	関数	119
4.9.3.1	addConicSum	119
4.9.3.2	clearConicSum	120
4.9.3.3	distanceConic	120
4.9.3.4	fitConic	120
4.9.3.5	fitConicAny	120
4.9.3.6	getConicProperty	120
4.9.3.7	getConicType	120
4.9.3.8	subConicSum	120

4.10	debugutil.cpp	121
4.10.1	説明	122
4.10.2	関数	122
4.10.2.1	drawDetectedEllipses	122
4.10.2.2	drawDetectedLines	122
4.10.2.3	drawDetectedVertices	122
4.10.2.4	drawEdgeImage	122
4.10.2.5	drawInputImage	123
4.10.2.6	drawStereoCircles	123
4.10.2.7	drawStereoCorrespondence	123
4.10.2.8	drawStereoVertices	123
4.10.2.9	drawTrackPoints	123
4.10.2.10	printStereoCircles	123
4.10.2.11	printStereoVertices	123
4.11	debugutil.h	124
4.11.1	説明	125
4.11.2	関数	125
4.11.2.1	drawDetectedEllipses	125
4.11.2.2	drawDetectedLines	125
4.11.2.3	drawDetectedVertices	125
4.11.2.4	drawEdgeImage	125
4.11.2.5	drawInputImage	126
4.11.2.6	drawStereoCircles	126
4.11.2.7	drawStereoCorrespondence	126
4.11.2.8	drawStereoVertices	126
4.11.2.9	drawTrackPoints	126
4.11.2.10	printStereoCircles	126
4.11.2.11	printStereoVertices	126
4.12	extractEdge.cpp	127
4.12.1	説明	128
4.12.2	マクロ定義	128
4.12.2.1	Edge	128
4.12.2.2	Gray	128
4.12.3	関数	128

4.12.3.1	<code>extractEdge</code>	128
4.13	<code>extractEdge.h</code>	129
4.13.1	説明	130
4.13.2	マクロ定義	130
4.13.2.1	<code>EE6</code>	130
4.13.2.2	<code>EE7</code>	130
4.13.2.3	<code>EEcandidate</code>	130
4.13.2.4	<code>EEerasedThin</code>	130
4.13.2.5	<code>EEextended</code>	130
4.13.2.6	<code>EEnotEdge</code>	130
4.13.2.7	<code>EEnotSearched</code>	130
4.13.2.8	<code>EEsearchedLarge</code>	130
4.13.2.9	<code>EEsearchedSmall</code>	131
4.13.3	関数	131
4.13.3.1	<code>extractEdge</code>	131
4.14	<code>extractFeature.cpp</code>	132
4.14.1	説明	133
4.14.2	マクロ定義	133
4.14.2.1	<code>Work</code>	133
4.14.3	関数	133
4.14.3.1	<code>destructFeatures</code>	133
4.14.3.2	<code>extractFeatures</code>	133
4.14.3.3	<code>ImageToFeature2D</code>	133
4.15	<code>extractFeature.h</code>	134
4.15.1	説明	135
4.15.2	関数	135
4.15.2.1	<code>destructFeatures</code>	135
4.15.2.2	<code>ImageToFeature2D</code>	135
4.16	<code>imageUtil.cpp</code>	136
4.16.1	説明	137
4.16.2	関数	137
4.16.2.1	<code>convertCameraIntrinsicParameterToCvMat</code>	137
4.16.2.2	<code>convertTimedMultiCameraImageToIplImage</code>	137
4.16.2.3	<code>convertTimedMultiCameraImageToRecogImage</code>	137

4.16.2.4	convertTimedMultiCameraImageToUndistortIplImage	138
4.16.2.5	createUndistortionMap	138
4.16.2.6	createUndistortionMapFromTimedMultiCameraImage	138
4.16.2.7	freeConvertedRecogImage	138
4.16.2.8	freeUndistortIplImage	138
4.17	imageUtil.h	139
4.17.1	説明	140
4.17.2	関数	140
4.17.2.1	convertTimedMultiCameraImageToIplImage	140
4.17.2.2	convertTimedMultiCameraImageToRecogImage	140
4.17.2.3	convertTimedMultiCameraImageToUndistortIplImage	140
4.17.2.4	createUndistortionMapFromTimedMultiCameraImage	141
4.17.2.5	freeConvertedRecogImage	141
4.17.2.6	freeUndistortIplImage	141
4.18	Img.hh	142
4.18.1	マクロ定義	146
4.18.1.1	_core_attr	146
4.18.1.2	_dyn_attr	146
4.18.1.3	USE_core_stub_in_nt_dll_NOT_DEFINED_Img	146
4.18.1.4	USE_dyn_stub_in_nt_dll_NOT_DEFINED_Img	146
4.18.1.5	USE_stub_in_nt_dll_NOT_DEFINED_Img	146
4.18.2	型定義	146
4.18.2.1	CameraCaptureService_out	146
4.18.2.2	CameraCaptureService_ptr	146
4.18.2.3	CameraCaptureService_var	147
4.18.2.4	CameraCaptureServiceRef	147
4.18.2.5	CameraImage_out	147
4.18.2.6	CameraImage_var	147
4.18.2.7	CameraIntrinsicParameter_out	147
4.18.2.8	CameraIntrinsicParameter_var	147
4.18.2.9	ColorFormat_out	147
4.18.2.10	ImageData_out	147
4.18.2.11	ImageData_var	147
4.18.2.12	Mat44	148

4.18.2.13	Mat44_forany	148
4.18.2.14	Mat44_out	148
4.18.2.15	Mat44_slice	148
4.18.2.16	Mat44_var	148
4.18.2.17	MultiCameraImage_out	148
4.18.2.18	MultiCameraImage_var	148
4.18.2.19	TimedCameraImage_out	148
4.18.2.20	TimedCameraImage_var	148
4.18.2.21	TimedMultiCameraImage_out	149
4.18.2.22	TimedMultiCameraImage_var	149
4.18.2.23	Vec3	149
4.18.2.24	Vec3_forany	149
4.18.2.25	Vec3_out	149
4.18.2.26	Vec3_slice	149
4.18.2.27	Vec3_var	149
4.18.3	列挙型	149
4.18.3.1	ColorFormat	149
4.18.4	関数	150
4.18.4.1	Mat44_alloc	150
4.18.4.2	Mat44_copy	150
4.18.4.3	Mat44_dup	150
4.18.4.4	Mat44_free	150
4.18.4.5	operator<<=	150
4.18.4.6	operator<<=	150
4.18.4.7	operator<<=	150
4.18.4.8	operator<<=	150
4.18.4.9	operator<<=	151
4.18.4.10	operator<<=	151
4.18.4.11	operator<<=	151
4.18.4.12	operator<<=	151
4.18.4.13	operator<<=	151
4.18.4.14	operator<<=	151
4.18.4.15	operator<<=	151
4.18.4.16	operator<<=	151

4.18.4.17	operator<<=	151
4.18.4.18	operator<<=	152
4.18.4.19	operator<<=	152
4.18.4.20	operator<<=	152
4.18.4.21	operator<<=	152
4.18.4.22	operator<<=	152
4.18.4.23	operator>>=	152
4.18.4.24	operator>>=	152
4.18.4.25	operator>>=	152
4.18.4.26	operator>>=	152
4.18.4.27	operator>>=	153
4.18.4.28	operator>>=	153
4.18.4.29	operator>>=	153
4.18.4.30	operator>>=	153
4.18.4.31	operator>>=	153
4.18.4.32	operator>>=	153
4.18.4.33	operator>>=	153
4.18.4.34	operator>>=	153
4.18.4.35	operator>>=	153
4.18.4.36	operator>>=	154
4.18.4.37	operator>>=	154
4.18.4.38	operator>>=	154
4.18.4.39	operator>>=	154
4.18.4.40	Vec3_alloc	154
4.18.4.41	Vec3_copy	154
4.18.4.42	Vec3_dup	154
4.18.4.43	Vec3_free	154
4.18.5	変数	154
4.18.5.1	_tc_CameraCaptureService	154
4.18.5.2	_tc_CameraImage	155
4.18.5.3	_tc_CameraIntrinsicParameter	155
4.18.5.4	_tc_ColorFormat	155
4.18.5.5	_tc_ImageData	155
4.18.5.6	_tc_Mat44	155

4.18.5.7	_tc_MultiCameraImage	155
4.18.5.8	_tc_TimedCameraImage	155
4.18.5.9	_tc_TimedMultiCameraImage	155
4.18.5.10	_tc_Vec3	155
4.19	local.h	156
4.20	match3Dfeature.cpp	157
4.20.1	説明	158
4.20.2	関数	158
4.20.2.1	freeFeatures3D	158
4.20.2.2	freeMatch3Dresults	158
4.20.2.3	matchFeatures3d	158
4.21	match3Dfeature.h	159
4.21.1	説明	160
4.21.2	列挙型	161
4.21.2.1	m3df_side	161
4.21.3	関数	161
4.21.3.1	freeFeatures3D	161
4.21.3.2	freeMatch3Dresults	161
4.21.3.3	matchFeatures3d	161
4.21.3.4	matchPairedCircles	161
4.22	modelFileio.cpp	162
4.22.1	関数	162
4.22.1.1	loadModelFile	162
4.23	modelFileio.h	163
4.23.1	説明	163
4.23.2	関数	163
4.23.2.1	loadModelFile	164
4.24	modelListFileIO.cpp	165
4.24.1	関数	165
4.24.1.1	clearModelFileInfo	165
4.24.1.2	loadModelListFile	165
4.25	modelListFileIO.h	166
4.25.1	説明	166
4.25.2	マクロ定義	166

4.25.2.1	MAX_PATH	167
4.25.3	関数	167
4.25.3.1	clearModelFileInfo	167
4.25.3.2	loadModelListFile	167
4.26	modelpoints.cpp	168
4.26.1	説明	169
4.26.2	列挙型	169
4.26.2.1	Azimuth	169
4.26.3	関数	169
4.26.3.1	drawModelPoints	170
4.26.3.2	getPointOnCircle	170
4.26.3.3	getPropertyVector	170
4.26.3.4	makeModelPoints	170
4.26.3.5	traceModelPointsMultiCameras	170
4.27	modelpoints.h	171
4.27.1	説明	172
4.27.2	関数	172
4.27.2.1	drawModelPoints	172
4.27.2.2	getPointOnCircle	172
4.27.2.3	getPropertyVector	172
4.27.2.4	makeModelPoints	172
4.27.2.5	traceModelPointsMultiCameras	172
4.28	pairedcircle.cpp	173
4.28.1	説明	173
4.28.2	関数	174
4.28.2.1	matchPairedCircles	174
4.29	parameters.h	175
4.29.1	説明	176
4.29.2	型定義	176
4.29.2.1	ParametersStereo	176
4.30	quaternion.c	177
4.30.1	関数	177
4.30.1.1	quat_conj	177
4.30.1.2	quat_copy	178

4.30.1.3	<code>quat_fprintf</code>	178
4.30.1.4	<code>quat_irot</code>	178
4.30.1.5	<code>quat_make_from_rvec</code>	178
4.30.1.6	<code>quat_mult</code>	178
4.30.1.7	<code>quat_norm2</code>	178
4.30.1.8	<code>quat_normalize</code>	178
4.30.1.9	<code>quat_q_from_R</code>	178
4.30.1.10	<code>quat_R_from_q</code>	178
4.30.1.11	<code>quat_rot</code>	178
4.31	<code>quaternion.h</code>	179
4.31.1	マクロ定義	180
4.31.1.1	<code>QUAT_EPS</code>	180
4.31.1.2	<code>quat_fprint</code>	180
4.31.1.3	<code>quat_im</code>	180
4.31.1.4	<code>QUAT_INIT_ONE</code>	180
4.31.1.5	<code>QUAT_INIT_ZERO</code>	180
4.31.1.6	<code>quat_print</code>	180
4.31.1.7	<code>quat_printf</code>	180
4.31.1.8	<code>quat_re</code>	181
4.31.2	型定義	181
4.31.2.1	<code>quaternion_t</code>	181
4.31.3	関数	181
4.31.3.1	<code>quat_conj</code>	181
4.31.3.2	<code>quat_copy</code>	181
4.31.3.3	<code>quat_fprintf</code>	181
4.31.3.4	<code>quat_irot</code>	181
4.31.3.5	<code>quat_make_from_rvec</code>	181
4.31.3.6	<code>quat_mult</code>	181
4.31.3.7	<code>quat_norm2</code>	181
4.31.3.8	<code>quat_normalize</code>	182
4.31.3.9	<code>quat_q_from_R</code>	182
4.31.3.10	<code>quat_R_from_q</code>	182
4.31.3.11	<code>quat_rot</code>	182
4.32	<code>recogImage.cpp</code>	183

4.32.1	説明	183
4.32.2	関数	184
4.32.2.1	constructImage	184
4.32.2.2	destructImage	184
4.32.2.3	rgb2grayImage	184
4.32.2.4	undistortImage	184
4.33	recogImage.h	185
4.33.1	説明	186
4.33.2	型定義	186
4.33.2.1	RecogImage	186
4.33.3	関数	186
4.33.3.1	constructImage	186
4.33.3.2	destructImage	186
4.33.3.3	rgb2grayImage	186
4.33.3.4	undistortImage	186
4.34	recogParameter.cpp	187
4.34.1	列挙型	188
4.34.1.1	paramKey	188
4.34.2	関数	188
4.34.2.1	loadDebugParameter	188
4.34.2.2	loadRecogParameter	188
4.34.2.3	setDefaultRecogParameter	189
4.35	recogParameter.h	190
4.35.1	説明	190
4.35.2	関数	190
4.35.2.1	loadDebugParameter	190
4.35.2.2	loadRecogParameter	191
4.35.2.3	setDefaultRecogParameter	191
4.36	recogResult.h	192
4.36.1	説明	192
4.36.2	マクロ定義	192
4.36.2.1	RecogResultElementNum	192
4.36.3	列挙型	192
4.36.3.1	RecogResultElement	192

4.37	rtvcm.cpp	194
4.37.1	説明	195
4.37.2	関数	195
4.37.2.1	convertRTVCMtoFeatures3D	195
4.37.2.2	freeRTVCM	195
4.37.2.3	readRTVCMModel	195
4.37.2.4	reverseCircle	195
4.37.2.5	reverseVertex	195
4.38	rtvcm.h	196
4.38.1	説明	197
4.38.2	型定義	197
4.38.2.1	RTVCM	197
4.38.2.2	RTVCM_Label	197
4.38.3	関数	197
4.38.3.1	convertRTVCMtoFeatures3D	197
4.38.3.2	freeRTVCM	197
4.38.3.3	readRTVCMModel	197
4.38.3.4	reverseCircle	198
4.38.3.5	reverseVertex	198
4.39	score2d.cpp	199
4.39.1	説明	200
4.39.2	マクロ定義	200
4.39.2.1	BOTTOMOFFSET	200
4.39.2.2	COL_TABLE	200
4.39.2.3	EDGE_SEARCH_2SIZE	200
4.39.2.4	EDGE_SEARCH_MAX	200
4.39.2.5	EDGE_SEARCH_SIZE	200
4.39.2.6	ROW_TABLE	200
4.39.3	関数	200
4.39.3.1	compareResultScore	201
4.39.3.2	getResultScore	201
4.39.3.3	isValidPixelPosition	201
4.39.3.4	tracePoint	201
4.40	score2d.h	202

4.40.1	説明	202
4.40.2	関数	202
4.40.2.1	compareResultScore	202
4.40.2.2	getResultScore	202
4.40.2.3	tracePoint	202
4.41	stereo.cpp	203
4.41.1	説明	204
4.41.2	関数	204
4.41.2.1	calculateLR2XYZ	204
4.41.2.2	calculateLR2XYZ2	204
4.41.2.3	calculateSightVector	205
4.41.2.4	freeStereoData	205
4.41.2.5	projectXYZ2LR	205
4.41.2.6	setFeature3D	205
4.41.2.7	setFeature3D_TBLAND	205
4.41.2.8	setFeature3D_TBLOR	205
4.41.2.9	StereoCorrespondence	205
4.42	stereo.h	206
4.42.1	説明	207
4.42.2	関数	208
4.42.2.1	calculateLR2XYZ	208
4.42.2.2	calculateSightVector	208
4.42.2.3	freeStereoData	208
4.42.2.4	projectXYZ2LR	208
4.42.2.5	setFeature3D	208
4.42.2.6	setFeature3D_TBLAND	208
4.42.2.7	setFeature3D_TBLOR	208
4.42.2.8	StereoCorrespondence	208
4.43	vectorutil.cpp	209
4.43.1	説明	210
4.43.2	関数	210
4.43.2.1	addV3	210
4.43.2.2	copyV2	210
4.43.2.3	copyV3	210

4.43.2.4	getAngle2D	210
4.43.2.5	getCrossProductV3	210
4.43.2.6	getDirectionVector	210
4.43.2.7	getDistanceV2	211
4.43.2.8	getDistanceV3	211
4.43.2.9	getInnerProductV3	211
4.43.2.10	getNormV2	211
4.43.2.11	getNormV3	211
4.43.2.12	getOrthogonalDir	211
4.43.2.13	getOrthogonalDir	211
4.43.2.14	inverseM33	211
4.43.2.15	isZero	211
4.43.2.16	mulM33	211
4.43.2.17	mulM33V3	212
4.43.2.18	mulV2S	212
4.43.2.19	mulV3S	212
4.43.2.20	normalizeV2	212
4.43.2.21	normalizeV3	212
4.43.2.22	quaternion_rotation	212
4.43.2.23	subM33	212
4.43.2.24	subV3	212
4.43.2.25	transposeM33	212
4.43.2.26	zeroV3	212
4.44	vectorutil.h	213
4.44.1	説明	214
4.44.2	型定義	214
4.44.2.1	M33	214
4.44.2.2	V2	214
4.44.2.3	V3	214
4.44.3	関数	215
4.44.3.1	addV3	215
4.44.3.2	copyV2	215
4.44.3.3	copyV3	215
4.44.3.4	getAngle2D	215

4.44.3.5	getCrossProductV3	215
4.44.3.6	getDirectionVector	215
4.44.3.7	getDistanceV2	215
4.44.3.8	getDistanceV3	215
4.44.3.9	getInnerProductV3	215
4.44.3.10	getNormV2	216
4.44.3.11	getNormV3	216
4.44.3.12	getOrthogonalDir	216
4.44.3.13	getOrthogonalDir	216
4.44.3.14	inverseM33	216
4.44.3.15	isZero	216
4.44.3.16	mulM33	216
4.44.3.17	mulM33V3	216
4.44.3.18	mulV2S	216
4.44.3.19	mulV3S	216
4.44.3.20	normalizeV2	217
4.44.3.21	normalizeV3	217
4.44.3.22	quaternion_rotation	217
4.44.3.23	subM33	217
4.44.3.24	subV3	217
4.44.3.25	transposeM33	217
4.45	vertex.cpp	218
4.45.1	説明	218
4.45.2	関数	219
4.45.2.1	HyperbolaToVertex	219
4.45.2.2	isValidPixelPosition	219
4.46	vertex.h	220
4.46.1	説明	220
4.46.2	関数	220
4.46.2.1	HyperbolaToVertex	220
4.47	visionErrorCode.h	221
4.47.1	説明	221
4.47.2	列挙型	221
4.47.2.1	VisionErrorCode	221

Chapter 1

データ構造索引

1.1 データ構造

データ構造の説明です。

_impl_CameraCaptureService	5
_objref_CameraCaptureService	7
_pof_CameraCaptureService	9
_recogImage	10
CalibParam (キャリブレーションパラメータ)	11
CameraCaptureService	13
CameraCaptureService_Helper	16
CameraImage	18
CameraIntrinsicParameter	20
CameraParam (カメラパラメータ)	22
Circle (3次元円情報)	25
CircleCandidate (3次元円特徴候補データ)	28
Data_2D	30
DistortionParam (歪みパラメータ)	31
EllipseGroup (楕円重複除去用グループ情報)	33
Feature2D (各2次元特徴)	34
Features2D (2次元特徴情報)	38
Features3D (3次元特徴情報)	40
ImageData	43
Mat44_copyHelper	45
Match3Dresults (全認識結果)	46
MatchResult (各認識結果情報)	48
ModelFileInfo (モデルファイルリスト)	50
ModelFileInfoNode (モデルリストのノード)	52
MultiCameraImage	53
P2D (2次元位置情報)	55
P3D (3次元位置情報)	56
Parameters (全パラメータ)	57
ParametersFeature2D (2次元特徴抽出用パラメータ)	60

ParametersMatch (認識用パラメータ)	65
ParametersStereo (ステレオ対応処理用パラメータ)	67
RTVCM_Box (モデル内の立方体データ)	69
RTVCM_Circle (モデル内の円データ)	71
RTVCM_Cylinder (モデル内の円筒データ)	73
RTVCM_Vertex (モデル内の頂点データ)	75
RTVertexCircleModel (モデルデータ構造体)	77
StereoCalib (ステレオカメラキャリブレーションデータ)	81
StereoConic (二次曲線ステレオ対応データ)	83
StereoData (ステレオ対応データ)	86
TimedCameraImage	88
TimedMultiCameraImage	90
Trace (認識結果評価用サンプリング点列情報)	92
Track (輪郭情報)	94
Vec3_copyHelper	95
Vertex (3次元頂点情報)	96
VertexCandidate (三次元頂点特徴候補データ)	99

Chapter 2

ファイル索引

2.1 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)	103
calib.h (キャリブレーション関連の関数)	105
calibUtil.cpp	107
calibUtil.h (キャリブレーションデータの変換関連)	108
circle.cpp (3次元円特徴生成関連関数)	110
circle.h (3次元円特徴生成関連関数)	112
common.h (各種の共通定義)	114
conic.cpp (二次曲線特徴抽出関連関数)	116
conic.h (二次曲線特徴抽出関連関数)	118
debugutil.cpp (デバッグ用関数)	121
debugutil.h (デバッグ用関数)	124
extractEdge.cpp (エッジ抽出関連関数)	127
extractEdge.h (エッジ抽出関連関数)	129
extractFeature.cpp (2次元特徴抽出関連関数)	132
extractFeature.h (2次元特徴抽出関連関数)	134
imageUtil.cpp (画像入出力関数)	136
imageUtil.h (画像入出力関連)	139
Img.hh	142
local.h	156
match3Dfeature.cpp (3次元特徴による認識関連関数)	157
match3Dfeature.h (3次元特徴による認識関連関数)	159
modelFileio.cpp	162
modelFileio.h (モデルをファイルから読み込む。)	163
modelListFileIO.cpp	165
modelListFileIO.h (モデルファイルの入出力関連)	166
modelpoints.cpp (モデル評価点生成関連関数)	168
modelpoints.h (モデル評価点生成関連関数)	171
pairedcircle.cpp (2円照合関連関数)	173
parameters.h (処理パラメータ設定関連関数)	175

quaternion.c	177
quaternion.h	179
recogImage.cpp (画像入出力関数)	183
recogImage.h (画像入出力関数)	185
recogParameter.cpp	187
recogParameter.h (認識パラメータ設定関連)	190
recogResult.h (認識結果の定義)	192
rtvcm.cpp (モデル入出力関連関数)	194
rtvcm.h (モデル入出力関連関数)	196
score2d.cpp (2次元評価関連関数)	199
score2d.h (2次元評価関連関数)	202
stereo.cpp (ステレオ処理関連関数)	203
stereo.h (ステレオ処理関連関数)	206
vectorutil.cpp (ベクトル処理、行列処理 ユーティリティ関数)	209
vectorutil.h (ベクトル処理、行列処理 ユーティリティ関数)	213
vertex.cpp (3次元頂点特徴生成関連関数)	218
vertex.h (3次元頂点特徴生成関連関数)	220
visionErrorCode.h (返り値の定義)	221

Chapter 3

データ構造

3.1 クラス `_impl_CameraCaptureService`

```
#include <Img.hh>
```

Public メソッド

- virtual `~_impl_CameraCaptureService` ()
- virtual void `take_one_frame` ()=0
- virtual `_CORBA_Boolean _dispatch` (omniCallHandle &)

3.1.1 コンストラクタとデストラクタ

3.1.1.1 `virtual _impl_CameraCaptureService::~~_impl_CameraCaptureService`
() [`virtual`]

3.1.2 関数

3.1.2.1 `virtual _CORBA_Boolean _impl_CameraCaptureService::_dispatch`
(omniCallHandle &) [`virtual`]

3.1.2.2 `virtual void _impl_CameraCaptureService::take_one_frame () [pure virtual]`

このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.2 クラス `_objref_CameraCaptureService`

```
#include <Img.hh>
```

Public メソッド

- void `take_one_frame` ()
- `_objref_CameraCaptureService` ()
- `_objref_CameraCaptureService` (omniIOR *, omniIdentity *)

Protected メソッド

- virtual `~_objref_CameraCaptureService` ()

フレンド

- class `CameraCaptureService`

3.2.1 コンストラクタとデストラクタ

3.2.1.1 `_objref_CameraCaptureService::_objref_CameraCaptureService` ()
[inline]

3.2.1.2 `_objref_CameraCaptureService::_objref_CameraCaptureService`
(omniIOR *, omniIdentity *)

3.2.1.3 virtual `_objref_CameraCaptureService::~~_objref_CameraCaptureService` () [protected, virtual]

3.2.2 関数

3.2.2.1 void _objref_CameraCaptureService::take_one_frame ()

3.2.3 フレンドと関連する関数

3.2.3.1 friend class CameraCaptureService [friend]

このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.3 クラス `_pof_CameraCaptureService`

```
#include <Img.hh>
```

Public メソッド

- [_pof_CameraCaptureService \(\)](#)
- `virtual ~_pof_CameraCaptureService ()`
- `virtual omniObjRef * newObjRef (omniIOR *, omniIdentity *)`
- `virtual _CORBA_Boolean is_a (const char *) const`

3.3.1 コンストラクタとデストラクタ

3.3.1.1 `_pof_CameraCaptureService::_pof_CameraCaptureService ()`
[inline]

3.3.1.2 `virtual _pof_CameraCaptureService::~~_pof_CameraCaptureService ()`
[virtual]

3.3.2 関数

3.3.2.1 `virtual _CORBA_Boolean _pof_CameraCaptureService::is_a (const char *) const` [virtual]

3.3.2.2 `virtual omniObjRef* _pof_CameraCaptureService::newObjRef (omniIOR *, omniIdentity *)` [virtual]

このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.4 構造体 `_recogImage`

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

変数

- `int colsize`
- `int rowsize`
- `int bytePerPixel`
- `unsigned char * pixel`

3.4.1 構造体

3.4.1.1 `int _recogImage::bytePerPixel`

3.4.1.2 `int _recogImage::colsize`

3.4.1.3 `unsigned char* _recogImage::pixel`

3.4.1.4 `int _recogImage::rowsize`

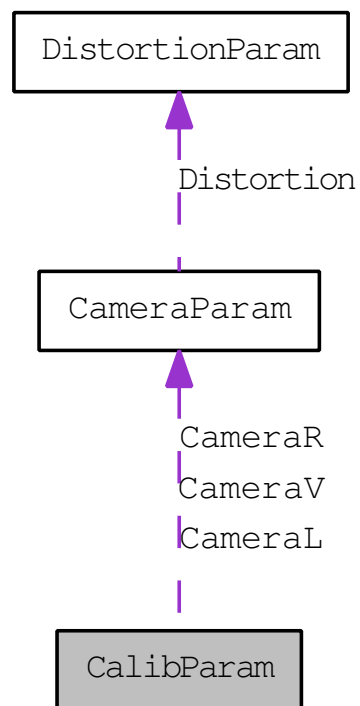
この構造体の説明は次のファイルから生成されました:

- `recogImage.h`

3.5 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

3.5.1 説明

キャリブレーションパラメータ

3.5.2 構造体

3.5.2.1 `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

3.5.2.2 `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

3.5.2.3 `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

3.5.2.4 `int CalibParam::colsize`

画像幅

3.5.2.5 `int CalibParam::numOfCameras`

カメラ数

3.5.2.6 `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](#)

3.6 クラス CameraCaptureService

```
#include <Img.hh>
```

Public 型

- typedef [CameraCaptureService_ptr_ptr_type](#)
- typedef [CameraCaptureService_var_var_type](#)

Public メソッド

- virtual [~CameraCaptureService \(\)](#)
- [inline::Img::CameraCaptureService_ptr_this \(\)](#)

Static Public メソッド

- static [_ptr_type _duplicate \(_ptr_type\)](#)
- static [_ptr_type _narrow \(CORBA::Object_ptr\)](#)
- static [_ptr_type _unchecked_narrow \(CORBA::Object_ptr\)](#)
- static [_ptr_type _nil \(\)](#)
- static void [_marshalObjRef \(_ptr_type, cdrStream &\)](#)
- static [_ptr_type _unmarshalObjRef \(cdrStream &s\)](#)

Static Public 変数

- static [_core_attr const char * _PD_repoId](#)

3.6.1 型定義

3.6.1.1 typedef CameraCaptureService_ptr CameraCaptureService::_ptr_type

3.6.1.2 typedef CameraCaptureService_var CameraCaptureService::_var_type

3.6.2 コンストラクタとデストラクタ

3.6.2.1 `virtual CameraCaptureService::~~CameraCaptureService ()`
`[virtual]`

3.6.3 関数

3.6.3.1 `static _ptr_type CameraCaptureService::_duplicate (_ptr_type)`
`[static]`

3.6.3.2 `static void CameraCaptureService::_marshalObjRef (_ptr_type,`
`cdrStream &) [inline, static]`

3.6.3.3 `static _ptr_type CameraCaptureService::_narrow`
`(CORBA::Object_ptr) [static]`

3.6.3.4 `static _ptr_type CameraCaptureService::_nil () [static]`

3.6.3.5 `inline ::Img::CameraCaptureService_ptr CameraCaptureService::_this`
`() [inline]`

3.6.3.6 `static _ptr_type CameraCaptureService::_unchecked_narrow`
`(CORBA::Object_ptr) [static]`

3.6.3.7 `static _ptr_type CameraCaptureService::_unmarshalObjRef`
`(cdrStream & s) [inline, static]`

3.6.4 構造体

3.6.4.1 `_core_attr const char* CameraCaptureService::_PD_repoId` `[static]`

このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.7 クラス CameraCaptureService_Helper

```
#include <Img.hh>
```

Public 型

- typedef CameraCaptureService_ptr _ptr_type

Static Public メソッド

- static _ptr_type _nil ()
- static _CORBA_Boolean is_nil (_ptr_type)
- static void release (_ptr_type)
- static void duplicate (_ptr_type)
- static void marshalObjRef (_ptr_type, cdrStream &)
- static _ptr_type unmarshalObjRef (cdrStream &)

3.7.1 型定義

3.7.1.1 typedef CameraCaptureService_ptr CameraCaptureService_Helper::_ptr_type

3.7.2 関数

3.7.2.1 static _ptr_type CameraCaptureService_Helper::_nil () [static]

3.7.2.2 static void CameraCaptureService_Helper::duplicate (_ptr_type) [static]

3.7.2.3 static _CORBA_Boolean CameraCaptureService_Helper::is_nil (_ptr_type) [static]

3.7.2.4 `static void CameraCaptureService_Helper::marshalObjRef (_ptr_type,
 cdrStream &) [static]`

3.7.2.5 `static void CameraCaptureService_Helper::release (_ptr_type)
 [static]`

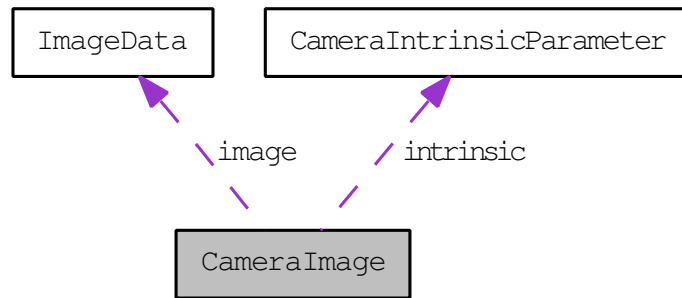
3.7.2.6 `static _ptr_type CameraCaptureService_Helper::unmarshalObjRef
 (cdrStream &) [static]`

このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.8 構造体 CameraImage

#include <Img.hh> CameraImage のコラボレーション図



Public 型

- typedef `_CORBA_ConstrType_Variable_Var< CameraImage > _var_type`

Public メソッド

- void `operator>>=` (cdrStream &) const
- void `operator<<=` (cdrStream &)

変数

- RTC::Time `captured_time`
- `ImageData` `image`
- `CameraIntrinsicParameter` `intrinsic`
- `Mat44` `extrinsic`

3.8.1 型定義

3.8.1.1 `typedef _CORBA_ConstrType_Variable_Var<CameraImage>
CameraImage::_var_type`

3.8.2 関数

3.8.2.1 void CameraImage::operator<<= (cdrStream &)

3.8.2.2 void CameraImage::operator>>= (cdrStream &) const

3.8.3 構造体

3.8.3.1 RTC::Time CameraImage::captured_time

3.8.3.2 Mat44 CameraImage::extrinsic

3.8.3.3 ImageData CameraImage::image

3.8.3.4 CameraIntrinsicParameter CameraImage::intrinsic

この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.9 構造体 CameraIntrinsicParameter

```
#include <Img.hh>
```

Public 型

- typedef _CORBA_ConstrType_Variable_Var< CameraIntrinsicParameter > _var_type
- typedef CORBA::Double _ORL_matrix_element [5]
- typedef CORBA::Double _matrix_element_slice
- typedef _CORBA_Unbounded_Sequence_w_FixSizeElement< CORBA::Double, 8, 8 > _distortion_coefficient_seq

Public メソッド

- void operator>>= (cdrStream &) const
- void operator<<= (cdrStream &)

変数

- CORBA::Double matrix_element [5]
- _distortion_coefficient_seq distortion_coefficient

3.9.1 型定義

3.9.1.1 typedef CORBA::Double CameraIntrinsicParameter::_ORL_matrix_element[5]

3.9.1.2 typedef _CORBA_Unbounded_Sequence_w_FixSizeElement< CORBA::Double, 8, 8 > CameraIntrinsicParameter::_distortion_coefficient_seq

3.9.1.3 `typedef CORBA::Double CameraIntrinsicParameter::_matrix_element_slice`

3.9.1.4 `typedef _CORBA_ConstrType_Variable_Var<CameraIntrinsicParameter> CameraIntrinsicParameter::_var_type`

3.9.2 関数

3.9.2.1 `void CameraIntrinsicParameter::operator<<= (cdrStream &)`

3.9.2.2 `void CameraIntrinsicParameter::operator>>= (cdrStream &) const`

3.9.3 構造体

3.9.3.1 `_distortion_coefficient_seq CameraIntrinsicParameter::distortion_coefficient`

3.9.3.2 `CORBA::Double CameraIntrinsicParameter::matrix_element[5]`

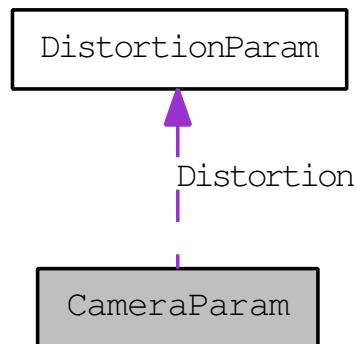
この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.10 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 `cx`
- double `cy`
 レンズの光学的な中心
- double `fx`
- double `fy`
 焦点距離 (ピクセル単位)
- double `Rotation` [3][3]
 回転行列
- double `Translation` [3]
 移動ベクトル
- double `rRotation` [3][3]
 回転行列の逆行列 (転置行列)
- 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

3.10.1 説明

カメラパラメータ

3.10.2 構造体

3.10.2.1 double CameraParam::cx

3.10.2.2 double CameraParam::cy

レンズの光学的な中心

3.10.2.3 DistortionParam CameraParam::Distortion

歪みパラメータ

3.10.2.4 double CameraParam::fx

3.10.2.5 double CameraParam::fy

焦点距離 (ピクセル単位)

3.10.2.6 double CameraParam::intrinsicMatrix[3][3]

内部パラメータ行列

3.10.2.7 double CameraParam::Position[3]

カメラ位置 ($-rR \cdot T$)

3.10.2.8 double CameraParam::Rotation[3][3]

回転行列

3.10.2.9 double CameraParam::rRotation[3][3]

回転行列の逆行列 (転置行列)

3.10.2.10 double CameraParam::Translation[3]

移動ベクトル

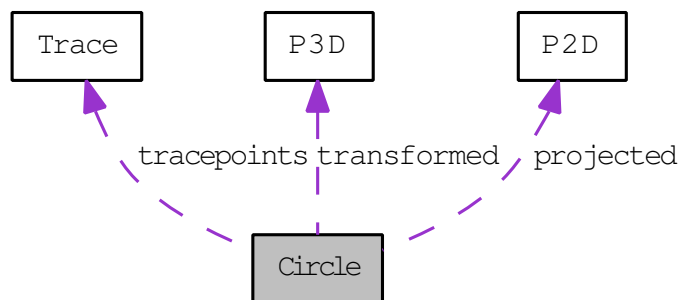
この構造体の説明は次のファイルから生成されました:

- [calib.h](#)

3.11 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`
- int `n`
通し番号
- int `side`
表裏情報
- double `radius`
半径
- double `center` [3]
中心位置
- double `normal` [3]
法線
- double `orientation` [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

3.11.1 説明

3 次元円情報

3.11.2 構造体

3.11.2.1 double Circle::center[3]

中心位置

3.11.2.2 int Circle::label

3.11.2.3 int Circle::n

通し番号

3.11.2.4 double Circle::normal[3]

法線

3.11.2.5 int Circle::numOfTracePoints

認識評価用のサンプリング点列数

3.11.2.6 double Circle::orientation[4][4]

認識用姿勢行列

3.11.2.7 P2D* Circle::projected

2 次元評価時の画像投影 2 次元点列

3.11.2.8 double Circle::radius

半径

3.11.2.9 int Circle::side

表裏情報

3.11.2.10 Trace* Circle::tracepoints

認識評価用のサンプリング点列情報

3.11.2.11 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](#)

3.12 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

3.12.1 説明

三次元円特徴候補データ

3.12.2 構造体

3.12.2.1 double CircleCandidate::center[3]

中心座標

3.12.2.2 double CircleCandidate::normal[3]

法線ベクトル (単位化済)

3.12.2.3 double CircleCandidate::radius

半径

3.12.2.4 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](#)

3.13 構造体 Data_2D

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

変数

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

3.13.1 構造体

3.13.1.1 double Data_2D::col

3.13.1.2 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](#)

3.14 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

3.14.1 説明

歪みパラメータ

3.14.2 構造体

3.14.2.1 double DistortionParam::k1

3.14.2.2 double DistortionParam::k2

半径方向の歪み係数

3.14.2.3 double DistortionParam::k3

半径方向の歪み係数

3.14.2.4 `double DistortionParam::p1`

3.14.2.5 `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](#)

3.15 構造体 `EllipseGroup`

楕円重複除去用グループ情報

```
#include <extractFeature.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

3.15.1 説明

楕円重複除去用グループ情報

3.15.2 構造体

3.15.2.1 `double EllipseGroup::groupCenter[2]`

(work) グループの中心座標

3.15.2.2 `int* EllipseGroup::groupNums`

グループ要素番号

3.15.2.3 `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.h`

3.16 構造体 Feature2D

各 2 次元特徴

```
#include <extractFeature.h>
```

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 1 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

3.16.1 説明

各 2 次元特徴

3.16.2 構造体

3.16.2.1 int Feature2D::all

輪郭全体の点数 (nPoint)

3.16.2.2 double Feature2D::axis[2]

楕円の長半径、短半径

3.16.2.3 double Feature2D::center[2]

楕円中心または、双曲線の漸近線交点

3.16.2.4 double Feature2D::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

3.16.2.5 double Feature2D::direction[2]

直線の方向ベクトル

3.16.2.6 int Feature2D::end

終点番号

3.16.2.7 double Feature2D::endPoint[2]

曲線上の終点

3.16.2.8 double Feature2D::endSPoint[2]

点列の終点の位置

3.16.2.9 double Feature2D::error

当てはめ誤差

3.16.2.10 double Feature2D::ev[2][2]

楕円の回転行列

3.16.2.11 double Feature2D::lineAngle

双曲線の 2 線分のなす角度

3.16.2.12 double Feature2D::lineLength

直線の長さ

3.16.2.13 `double Feature2D::lineLength1`

双曲線の線分 1 の長さ

3.16.2.14 `double Feature2D::lineLength2`

双曲線の線分 2 の長さ

3.16.2.15 `double Feature2D::middleSPoint[2]`

点列の中間の位置

3.16.2.16 `int Feature2D::nPoints`

特徴抽出に使われた点数

3.16.2.17 `int Feature2D::nTrack`

輪郭番号

3.16.2.18 `int Feature2D::start`

始点番号

3.16.2.19 `double Feature2D::startPoint[2]`

曲線上の始点

3.16.2.20 `double Feature2D::startSPoint[2]`

点列の始点の位置

3.16.2.21 `ConicType Feature2D::type`

二次曲線の分類

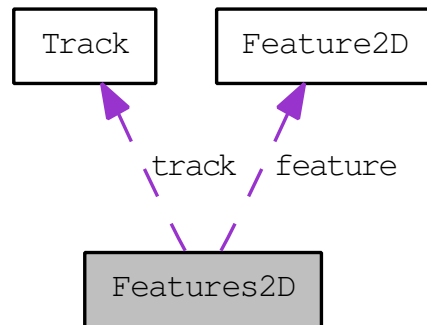
この構造体の説明は次のファイルから生成されました:

- [extractFeature.h](#)

3.17 構造体 Features2D

2次元特徴情報

`#include <extractFeature.h>` Features2D 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 * feature`
2次元特徴情報
- `int nTrack`
輪郭数
- `Track * track`
輪郭情報

3.17.1 説明

2次元特徴情報

3.17.2 構造体

3.17.2.1 Feature2D* Features2D::feature

2次元特徴情報

3.17.2.2 int Features2D::nAlloc

メモリ確保量

3.17.2.3 int Features2D::nFeature

2次元特徴数

3.17.2.4 int Features2D::nTrack

輪郭数

3.17.2.5 Track* Features2D::track

輪郭情報

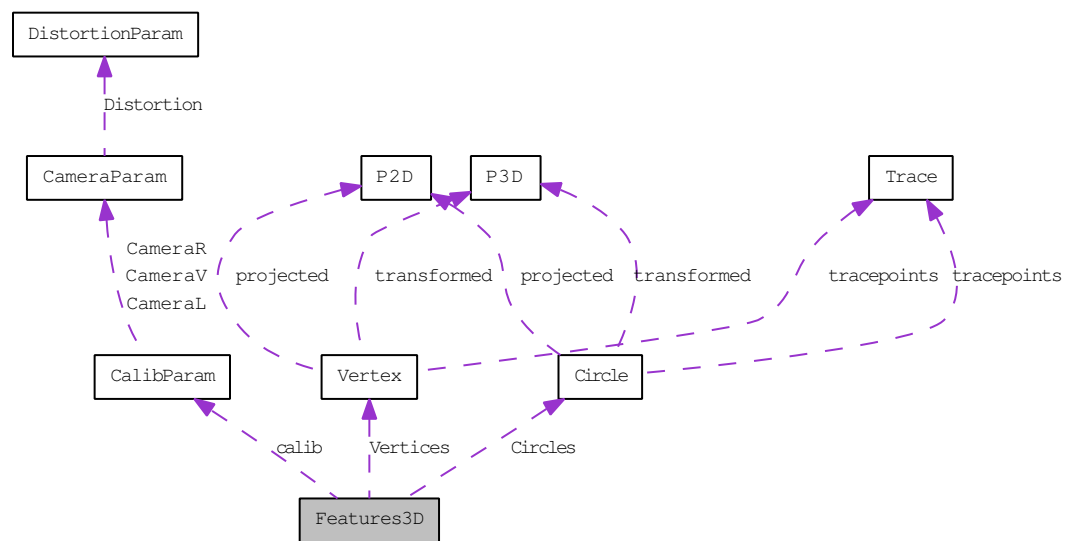
この構造体の説明は次のファイルから生成されました:

- [extractFeature.h](#)

3.18 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
- **int numOfCircles**
3次元円特徴数
- **Circle * Circles**
3次元円特徴
- **uchar * image [3]**
原画像ポインタ

- `uchar * edge [3]`
エッジ画像ポインタ
- `double trace_pdist`
2次元評価時の評価点間隔
- `int trace_search`
2次元評価時の探索範囲
- `int trace_edge`
2次元評価時の有効エッジ強度閾値
- `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

3.18.1 説明

3次元特徴情報

3.18.2 構造体

3.18.2.1 `CalibParam* Features3D::calib`

キャリブレーションデータポインタ

3.18.2.2 `Circle* Features3D::Circles`

3次元円特徴

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

エッジ画像ポインタ

3.18.2.4 `uchar* Features3D::image[3]`

原画像ポインタ

3.18.2.5 int Features3D::numOfCircles

3次元円特徴数

3.18.2.6 int Features3D::numOfVertices

3次元頂点特徴数

3.18.2.7 int Features3D::pointCounts

2次元評価のための全評価点数

3.18.2.8 int Features3D::trace_edge

2次元評価時の有効エッジ強度閾値

3.18.2.9 double Features3D::trace_pdist

2次元評価時の評価点間隔

3.18.2.10 int Features3D::trace_search

2次元評価時の探索範囲

3.18.2.11 double Features3D::traceCounts

2次元評価に用いた評価点数

3.18.2.12 Vertex* Features3D::Vertices

3次元頂点特徴

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](#)

3.19 構造体 ImageData

```
#include <Img.hh>
```

Public 型

- typedef _CORBA_ConstrType_Variable_Var< ImageData > _var_type
- typedef _CORBA_Unbounded_Sequence_Octet _raw_data_seq

Public メソッド

- void operator>>= (cdrStream &) const
- void operator<<= (cdrStream &)

変数

- CORBA::Long width
- CORBA::Long height
- ColorFormat format
- _raw_data_seq raw_data

3.19.1 型定義

3.19.1.1 typedef _CORBA_Unbounded_Sequence_Octet
ImageData::_raw_data_seq

3.19.1.2 typedef _CORBA_ConstrType_Variable_Var<ImageData>
ImageData::_var_type

3.19.2 関数

3.19.2.1 void ImageData::operator<<= (cdrStream &)

3.19.2.2 void ImageData::operator>>= (cdrStream &) const

3.19.3 構造体

3.19.3.1 ColorFormat ImageData::format

3.19.3.2 CORBA::Long ImageData::height

3.19.3.3 _raw_data_seq ImageData::raw_data

3.19.3.4 CORBA::Long ImageData::width

この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.20 クラス Mat44_copyHelper

```
#include <Img.hh>
```

Static Public メソッド

- static [Mat44_slice](#) * [alloc](#) ()
- static [Mat44_slice](#) * [dup](#) (const [Mat44_slice](#) *p)
- static void [free](#) ([Mat44_slice](#) *p)

3.20.1 関数

3.20.1.1 `static Mat44_slice* Mat44_copyHelper::alloc () [inline, static]`

3.20.1.2 `static Mat44_slice* Mat44_copyHelper::dup (const Mat44_slice * p) [inline, static]`

3.20.1.3 `static void Mat44_copyHelper::free (Mat44_slice * p) [inline, static]`

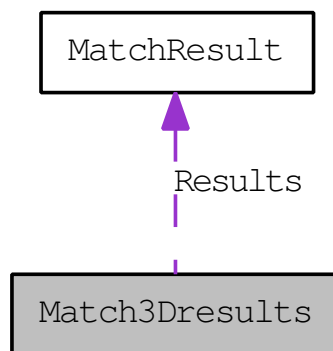
このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.21 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

3.21.1 説明

全認識結果

3.21.2 構造体

3.21.2.1 int Match3Dresults::error

認識エラーフラグ

3.21.2.2 int Match3Dresults::numOfResults

認識結果数

3.21.2.3 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](#)

3.22 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

3.22.1 説明

各認識結果情報

3.22.2 構造体

3.22.2.1 double **MatchResult::mat**[4][4]

変換行列

3.22.2.2 int MatchResult::model[2]

モデル特徴番号

3.22.2.3 int MatchResult::n

通し番号

3.22.2.4 int MatchResult::scene[2]

シーン特徴番号

3.22.2.5 double MatchResult::score

2次元評価値

3.22.2.6 int MatchResult::type

特徴タイプ 0:頂点、1:単円、2:2円

3.22.2.7 double MatchResult::vec[7]

変換行列の7次元のベクトル（位置 + 回転）表現

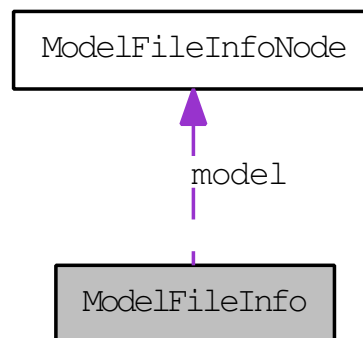
この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

3.23 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

3.23.1 説明

モデルファイルリスト

3.23.2 構造体

3.23.2.1 **ModelFileInfoNode* ModelFileInfo::model**

モデルリスト

3.23.2.2 `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](#)

3.24 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

3.24.1 説明

モデルリストのノード

3.24.2 構造体

3.24.2.1 `int ModelFileInfoNode::id`

モデル ID

3.24.2.2 `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](#)

3.25 構造体 MultiCameraImage

```
#include <Img.hh>
```

Public 型

- typedef _CORBA_ConstrType_Variable_Var< MultiCameraImage > _var_type
- typedef _CORBA_Unbounded_Sequence< CameraImage > _image_seq_seq

Public メソッド

- void operator>>= (cdrStream &) const
- void operator<<= (cdrStream &)

変数

- _image_seq_seq image_seq
- CORBA::Long camera_set_id

3.25.1 型定義

3.25.1.1 typedef _CORBA_Unbounded_Sequence< CameraImage >
MultiCameraImage::_image_seq_seq

3.25.1.2 typedef _CORBA_ConstrType_Variable_Var<MultiCameraImage>
MultiCameraImage::_var_type

3.25.2 関数

3.25.2.1 void MultiCameraImage::operator<<= (cdrStream &)

3.25.2.2 `void MultiCameraImage::operator>>= (cdrStream &) const`

3.25.3 構造体

3.25.3.1 `CORBA::Long MultiCameraImage::camera_set_id`

3.25.3.2 `_image_seq_seq MultiCameraImage::image_seq`

この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.26 構造体 P2D

2次元位置情報

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

変数

- double [colrow](#) [2]
2次元座標

3.26.1 説明

2次元位置情報

3.26.2 構造体

3.26.2.1 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](#)

3.27 構造体 P3D

3次元位置情報

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

変数

- `double xyz[3]`
3次元座標

3.27.1 説明

3次元位置情報

3.27.2 構造体

3.27.2.1 `double P3D::xyz[3]`

3次元座標

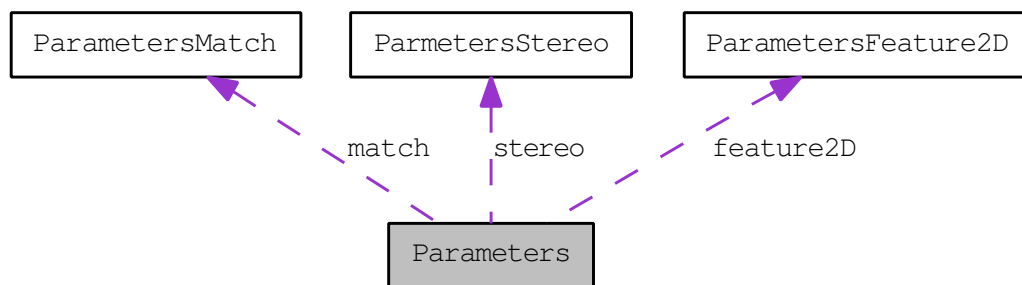
この構造体の説明は次のファイルから生成されました:

- [match3Dfeature.h](#)

3.28 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`
- `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

3.28.1 説明

全パラメータ

3.28.2 構造体

3.28.2.1 `int Parameters::colsize`

画像サイズ横（画素）

3.28.2.2 `int Parameters::dbgdisp`

デバッグ画像表示

3.28.2.3 `int Parameters::dbgimag`

デバッグ画像生成

3.28.2.4 `int Parameters::dbgtext`

デバッグテキスト生成

3.28.2.5 `ParametersFeature2D Parameters::feature2D`

3.28.2.6 `int Parameters::imgsize`

画像サイズ総画素数

3.28.2.7 `ParametersMatch Parameters::match`

3.28.2.8 `int Parameters::outputCandNum`

出力候補数

3.28.2.9 `StereoPairing Parameters::pairing`

3.28.2.10 int Parameters::rowsize

画像サイズ縦（画素）

3.28.2.11 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](#)

3.29 構造体 ParametersFeature2D

2次元特徴抽出用パラメータ

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

変数

- int `id`
ID 番号.
- 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 `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 ~ 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 ~ 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 ~ 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)

3.29.1 説明

2次元特徴抽出用パラメータ

3.29.2 構造体

3.29.2.1 int ParametersFeature2D::edgeDetectFunction

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

3.29.2.2 double ParametersFeature2D::edgeStrength

検出するエッジの最低微分強度

3.29.2.3 int ParametersFeature2D::id

ID 番号.

3.29.2.4 double ParametersFeature2D::max_distance_ellipse_grouping

中心距離判定閾値 (画素)

3.29.2.5 double ParametersFeature2D::max_distance_ellipse_pairing

中心距離判定閾値 (画素)

3.29.2.6 double ParametersFeature2D::max_distance_end_points

端点間距離の閾値 (画素)

3.29.2.7 double ParametersFeature2D::max_flatness_ellipse

楕円の偏平率 (長軸/短軸)

3.29.2.8 double ParametersFeature2D::max_length_delete_line

削除する直線の最大の長さ (画素)

3.29.2.9 double ParametersFeature2D::max_length_ellipse_axisL

楕円の軸長の閾値 (画素)

3.29.2.10 double ParametersFeature2D::maxErrorofConicFit

二次曲線をあてはめる時の最大誤差 (画素)

3.29.2.11 double ParametersFeature2D::maxErrorofLineFit

直線をあてはめる時の最大誤差 (画素)

3.29.2.12 double ParametersFeature2D::min_distance_ellipse_pairing

中心距離判定閾値 (画素)

3.29.2.13 double ParametersFeature2D::min_filling_ellipse

楕円の充填率の閾値 (0.0 ~ 1.0)

3.29.2.14 double ParametersFeature2D::min_length_ellipse_axis

楕円の軸長の閾値 (画素)

3.29.2.15 double ParametersFeature2D::min_length_ellipse_axisL

楕円の軸長の閾値 (画素)

3.29.2.16 double ParametersFeature2D::min_length_ellipse_axisS

楕円の軸長の閾値 (画素)

3.29.2.17 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)

3.29.2.18 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)

3.29.2.19 double ParametersFeature2D::min_length_line

直線の長さの閾値 (画素)

3.29.2.20 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)

3.29.2.21 int ParametersFeature2D::minFragment

検出するエッジの最低外周長 (画素)

3.29.2.22 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 ~ 1.0)

3.29.2.23 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 ~ 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](#)

3.30 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)
- double [interval](#)
評価時サンプル点間隔 (画素)
- int [search](#)
評価時エッジ探索範囲 (画素)
- int [edge](#)
評価時エッジ強度閾値

3.30.1 説明

認識用パラメータ

3.30.2 構造体

3.30.2.1 int ParametersMatch::edge

評価時エッジ強度閾値

3.30.2.2 double ParametersMatch::interval

評価時サンプル点間隔 (画素)

3.30.2.3 double ParametersMatch::pdist

モデルサンプル点間隔 (mm)

3.30.2.4 int ParametersMatch::search

評価時エッジ探索範囲 (画素)

3.30.2.5 double ParametersMatch::tolerance1

頂点の角度差の許容割合 (%)

3.30.2.6 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](#)

3.31 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)
- double **lmin**
頂点 線分最小値 (*mm*)
- double **lmax**
頂点 線分最大値 (*mm*)

3.31.1 説明

ステレオ対応処理用パラメータ

3.31.2 構造体

3.31.2.1 double ParmetersStereo::amax

頂点 角度最大値 (度)

3.31.2.2 double ParmetersStereo::amin

頂点 角度最小値 (度)

3.31.2.3 double ParmetersStereo::depf

円中心・頂点位置奥行終了 (mm)

3.31.2.4 double ParmetersStereo::depn

円中心・頂点位置奥行開始 (mm)

3.31.2.5 double ParmetersStereo::ethr

対応誤差閾値 (mm)

3.31.2.6 double ParmetersStereo::lmax

頂点 線分最大値 (mm)

3.31.2.7 double ParmetersStereo::lmin

頂点 線分最小値 (mm)

3.31.2.8 double ParmetersStereo::ndif

左右法線角度許容差 (度)

3.31.2.9 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](#)

3.32 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

3.32.1 説明

モデル内の立方体データ

3.32.2 構造体

3.32.2.1 int RTVCM_Box::n

通し番号

3.32.2.2 int RTVCM_Box::nVertex[24]

頂点通し番号列

3.32.2.3 void* RTVCM_Box::reserved

拡張

3.32.2.4 double RTVCM_Box::Rotate[3][3]

基準位置からの回転

3.32.2.5 double RTVCM_Box::Trans[3]

基準位置からの移動

3.32.2.6 double RTVCM_Box::x**3.32.2.7 double RTVCM_Box::y****3.32.2.8 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](#)

3.33 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

3.33.1 説明

モデル内の円データ

3.33.2 構造体

3.33.2.1 double RTVCM_Circle::center[3]

中心の 3 次元位置

3.33.2.2 int RTVCM_Circle::n

通し番号

3.33.2.3 int RTVCM_Circle::ncyliner

属する円筒の通し番号

3.33.2.4 double RTVCM_Circle::normal[3]

3次元法線方向

3.33.2.5 double RTVCM_Circle::radius

半径

3.33.2.6 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](#)

3.34 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

3.34.1 説明

モデル内の円筒データ

3.34.2 構造体

3.34.2.1 double RTVCM_Cylinder::height

高さ

3.34.2.2 int RTVCM_Cylinder::n

通し番号

3.34.2.3 int* RTVCM_Cylinder::nCircle[2]

構成円の通し番号配列

3.34.2.4 double RTVCM_Cylinder::radius

半径

3.34.2.5 void* RTVCM_Cylinder::reserved

拡張

3.34.2.6 double RTVCM_Cylinder::Rotate[3][3]

基準位置からの回転

3.34.2.7 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](#)

3.35 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

3.35.1 説明

モデル内の頂点データ

3.35.2 構造体

3.35.2.1 double RTVCM_Vertex::angle

2 直線のなす角度

3.35.2.2 double RTVCM_Vertex::endpoint1[3]

3次元端点 1

3.35.2.3 double RTVCM_Vertex::endpoint2[3]

3次元端点 2

3.35.2.4 int RTVCM_Vertex::n

通し番号

3.35.2.5 int RTVCM_Vertex::nbox

属する立方体の通し番号

3.35.2.6 double RTVCM_Vertex::position[3]

3次元位置

3.35.2.7 void* RTVCM_Vertex::reserved

拡張

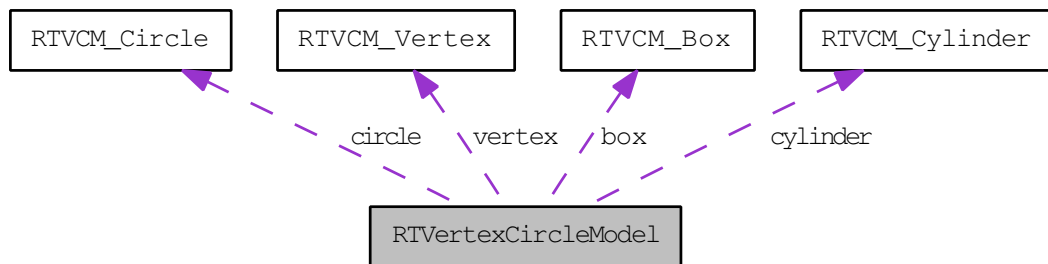
この構造体の説明は次のファイルから生成されました:

- [rtvcm.h](#)

3.36 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

3.36.1 説明

モデルデータ構造体

3.36.2 構造体

3.36.2.1 `RTVCM_Box* RTVertexCircleModel::box`

直方体列 (表示用)

3.36.2.2 `RTVCM_Circle* RTVertexCircleModel::circle`

円列

3.36.2.3 `RTVCM_Cylinder* RTVertexCircleModel::cylinder`

円筒列 (表示用)

3.36.2.4 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)

3.36.2.5 double RTVertexCircleModel::gravity[3]

重心

3.36.2.6 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)

3.36.2.7 RTVCM_Label RTVertexCircleModel::label

属性

3.36.2.8 int RTVertexCircleModel::n

通し番号

3.36.2.9 int RTVertexCircleModel::nbox

直方体数

3.36.2.10 int RTVertexCircleModel::ncircle

円数

3.36.2.11 int RTVertexCircleModel::ncylinder

円筒数

3.36.2.12 int RTVertexCircleModel::nvertex

頂点数

3.36.2.13 double RTVertexCircleModel::radius

属性が円柱の場合の半径

3.36.2.14 void* RTVertexCircleModel::reserved

拡張

3.36.2.15 RTVCM_Vertex* RTVertexCircleModel::vertex

頂点列

3.36.2.16 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](#)

3.37 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.

3.37.1 説明

ステレオカメラキャリブレーションデータ

3.37.2 構造体

3.37.2.1 double StereoCalib::baselineLR

LR 間ベースライン長.

3.37.2.2 double StereoCalib::baselineLV

LV 間ベースライン長.

3.37.2.3 double StereoCalib::baselineRV

RV 間ベースライン長.

3.37.2.4 int StereoCalib::height

画像高さ

3.37.2.5 int StereoCalib::numOfCameras

ステレオセットのカメラ数

3.37.2.6 int StereoCalib::width

画像幅

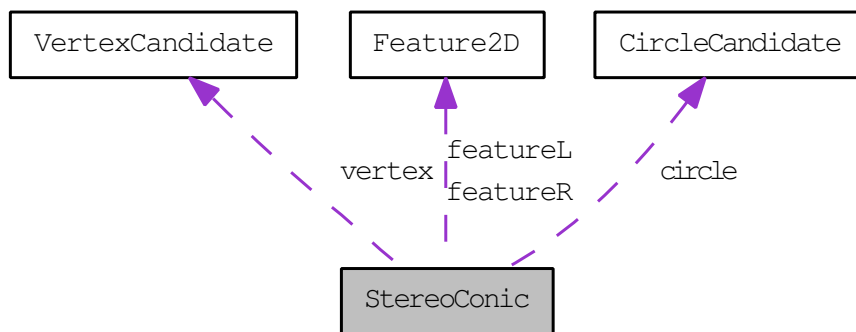
この構造体の説明は次のファイルから生成されました:

- [stereo.h](#)

3.38 構造体 StereoConic

二次曲線ステレオ対応データ

#include <stereo.h> StereoConic のコラボレーション図



変数

- [ConicType type](#)
二次曲線のタイプ
- [Feature2D * featureL](#)
左画像の2次元特徴
- [Feature2D * featureR](#)
右画像の2次元特徴
- [int valid](#)
有効無効フラグ
- [double error](#)
ステレオ対応誤差
- [double center \[3\]](#)
復元円中心、復元頂点の3次元座標
- [union {](#)
 - [VertexCandidate vertex](#)
頂点として処理した場合に保存する頂点情報
 - [CircleCandidate circle](#)
円として処理した場合に保存する円情報
- [} work](#)

3.38.1 説明

二次曲線ステレオ対応データ

3.38.2 構造体

3.38.2.1 `double StereoConic::center[3]`

復元円中心、復元頂点の 3 次元座標

3.38.2.2 `CircleCandidate StereoConic::circle`

円として処理した場合に保存する円情報

3.38.2.3 `double StereoConic::error`

ステレオ対応誤差

3.38.2.4 `Feature2D* StereoConic::featureL`

左画像の 2 次元特徴

3.38.2.5 `Feature2D* StereoConic::featureR`

右画像の 2 次元特徴

3.38.2.6 `ConicType StereoConic::type`

二次曲線のタイプ

3.38.2.7 `int StereoConic::valid`

有効無効フラグ

3.38.2.8 `VertexCandidate StereoConic::vertex`

頂点として処理した場合に保存する頂点情報

3.38.2.9 union { ... } StereoConic::work

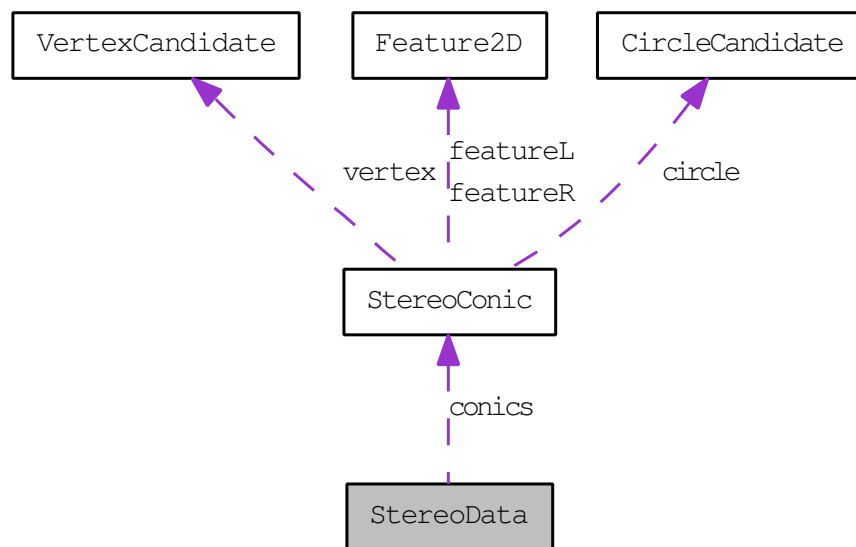
この構造体の説明は次のファイルから生成されました:

- [stereo.h](#)

3.39 構造体 StereoData

ステレオ対応データ

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



変数

- `int numOfconics`
2 次曲線ステレオ対応個数
- `StereoConic * conics`
2 次曲線ステレオ対応データ

3.39.1 説明

ステレオ対応データ

3.39.2 構造体

3.39.2.1 StereoConic* StereoData::conics

2 次曲線ステレオ対応データ

3.39.2.2 int StereoData::numOfconics

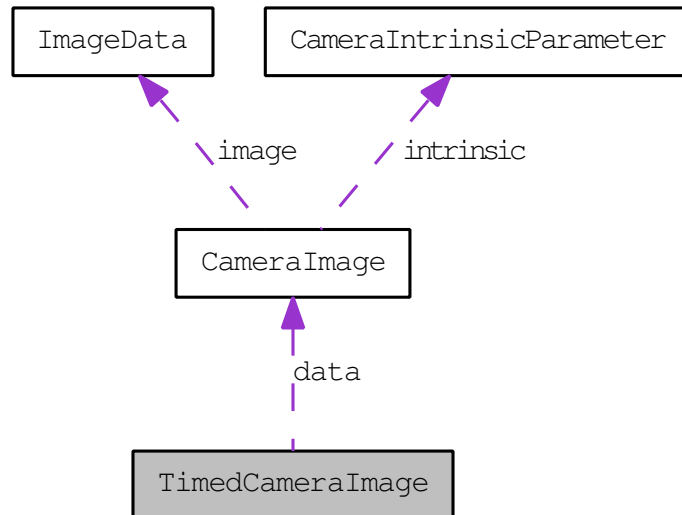
2 次曲線ステレオ対応個数

この構造体の説明は次のファイルから生成されました:

- [stereo.h](#)

3.40 構造体 TimedCameraImage

#include <Img.hh>TimedCameraImage のコラボレーション図



Public 型

- typedef _CORBA_ConstrType_Variable_Var< [TimedCameraImage](#) >
_var_type

Public メソッド

- void [operator>>=](#) (cdrStream &) const
- void [operator<<=](#) (cdrStream &)

変数

- RTC::Time [tm](#)
- [CameraImage](#) data
- CORBA::Long [error_code](#)

3.40.1 型定義

3.40.1.1 `typedef _CORBA_ConstrType_Variable_Var<TimedCameraImage>
TimedCameraImage::_var_type`

3.40.2 関数

3.40.2.1 `void TimedCameraImage::operator<<= (cdrStream &)`

3.40.2.2 `void TimedCameraImage::operator>>= (cdrStream &) const`

3.40.3 構造体

3.40.3.1 `CameraImage TimedCameraImage::data`

3.40.3.2 `CORBA::Long TimedCameraImage::error_code`

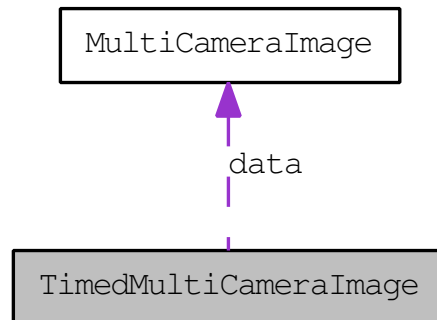
3.40.3.3 `RTC::Time TimedCameraImage::tm`

この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.41 構造体 TimedMultiCameraImage

#include <Img.hh>TimedMultiCameraImage のコラボレーション図



Public 型

- typedef _CORBA_ConstrType_Variable_Var< [TimedMultiCameraImage](#) > [_var_type](#)

Public メソッド

- void [operator>>=](#) (cdrStream &) const
- void [operator<<=](#) (cdrStream &)

変数

- RTC::Time [tm](#)
- [MultiCameraImage](#) [data](#)
- CORBA::Long [error_code](#)

3.41.1 型定義

3.41.1.1 typedef _CORBA_ConstrType_Variable_Var<[TimedMultiCameraImage](#)> [TimedMultiCameraImage::_var_type](#)

3.41.2 関数

3.41.2.1 void TimedMultiCameraImage::operator<<= (cdrStream &)

3.41.2.2 void TimedMultiCameraImage::operator>>= (cdrStream &) const

3.41.3 構造体

3.41.3.1 MultiCameraImage TimedMultiCameraImage::data

3.41.3.2 CORBA::Long TimedMultiCameraImage::error_code

3.41.3.3 RTC::Time TimedMultiCameraImage::tm

この構造体の説明は次のファイルから生成されました:

- [Img.hh](#)

3.42 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）

3.42.1 説明

認識結果評価用サンプリング点列情報

3.42.2 構造体

3.42.2.1 double Trace::colrow[2]

画像投影位置情報（画素）

3.42.2.2 `int Trace::direction`

対応エッジ探索方向

3.42.2.3 `int Trace::edge`

対応エッジの強度

3.42.2.4 `int Trace::label`

ラベル：可視情報 (VISIBLE/INVISIBLE)

3.42.2.5 `double Trace::peakcr[2]`

対応エッジ点の位置（画素）

3.42.2.6 `int Trace::search`

対応エッジの距離（画素）

3.42.2.7 `double Trace::weight`**3.42.2.8 `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](#)

3.43 構造体 Track

輪郭情報

```
#include <extractFeature.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

3.43.1 説明

輪郭情報

3.43.2 構造体

3.43.2.1 int Track::nPoint

点数

3.43.2.2 double Track::offset[2]

楕円係数計算時のオフセット

3.43.2.3 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.h](#)

3.44 クラス Vec3_copyHelper

```
#include <Img.hh>
```

Static Public メソッド

- static [Vec3_slice](#) * [alloc](#) ()
- static [Vec3_slice](#) * [dup](#) (const [Vec3_slice](#) *p)
- static void [free](#) ([Vec3_slice](#) *p)

3.44.1 関数

3.44.1.1 static [Vec3_slice](#)* [Vec3_copyHelper::alloc](#) () [[inline](#), [static](#)]

3.44.1.2 static [Vec3_slice](#)* [Vec3_copyHelper::dup](#) (const [Vec3_slice](#) * *p*)
[[inline](#), [static](#)]

3.44.1.3 static void [Vec3_copyHelper::free](#) ([Vec3_slice](#) * *p*) [[inline](#),
[static](#)]

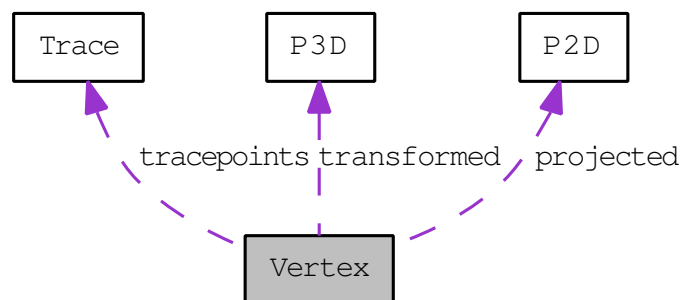
このクラスの説明は次のファイルから生成されました:

- [Img.hh](#)

3.45 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：可視情報 (*VISIBLE/INVISIBLE*)
- int **n**
通し番号
- int **side**
表裏情報
- double **position** [3]
頂点の 3 次元位置 (*mm*)
- 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 [orientation](#) [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)

3.45.1 説明

3 次元頂点情報

3.45.2 構造体

3.45.2.1 double Vertex::angle

頂点角度 (ラジアン)

3.45.2.2 double Vertex::direction1[3]

辺の方向

3.45.2.3 double Vertex::direction2[3]

辺の方向

3.45.2.4 double Vertex::endpoint1[3]

辺の端点 (*mm*)

3.45.2.5 double Vertex::endpoint2[3]

辺の端点 (mm)

3.45.2.6 int Vertex::label

ラベル : 可視情報 (VISIBLE/INVISIBLE)

3.45.2.7 int Vertex::n

通し番号

3.45.2.8 int Vertex::numOfTracePoints

認識評価用のサンプリング点列数

3.45.2.9 double Vertex::orientation[4][4]

認識用姿勢行列

3.45.2.10 double Vertex::position[3]

頂点の 3 次元位置 (mm)

3.45.2.11 P2D* Vertex::projected

2 次元評価時の画像投影 2 次元点列 (画素)

3.45.2.12 int Vertex::side

表裏情報

3.45.2.13 Trace* Vertex::tracepoints

認識評価用のサンプリング点列情報

3.45.2.14 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](#)

3.46 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

3.46.1 説明

三次元頂点特徴候補データ

3.46.2 構造体

3.46.2.1 double VertexCandidate::angle

頂点を成す線分の角度

3.46.2.2 double VertexCandidate::endpoint1[3]

端点 1 座標

3.46.2.3 double VertexCandidate::endpoint2[3]

端点 2 座標

3.46.2.4 double VertexCandidate::len1

線分 1 の長さ

3.46.2.5 double VertexCandidate::len2

線分 2 の長さ

3.46.2.6 int VertexCandidate::n1

for debug

3.46.2.7 int VertexCandidate::n2

for debug

3.46.2.8 int VertexCandidate::n3

for debug

3.46.2.9 double VertexCandidate::position[3]

頂点座標

3.46.2.10 int VertexCandidate::valid

有効・無効フラグ

3.46.2.11 double VertexCandidate::vector1[3]

端点 1 向き単位ベクトル

3.46.2.12 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](#)

Chapter 4

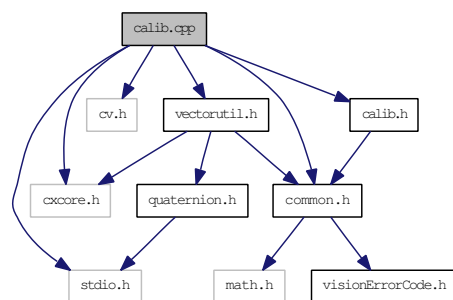
ファイル

4.1 calib.cpp

キャリブレーション関連の関数

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

4.1.1 説明

キャリブレーション関連の関数

日付:

\$Date:: 2011-06-30 19:26:38 +0900 #

4.1.2 関数

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

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

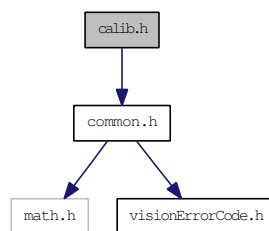
4.1.2.3 void `projectPoint` (`Data_2D` * *iPos2D*, `Data_2D` *icPos2D*, `CameraParam` * *cameraParam*)

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

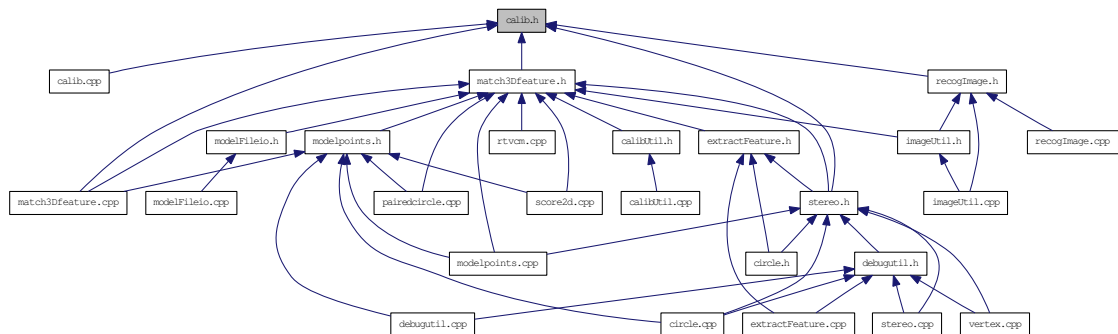
4.2 calib.h

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)

4.2.1 説明

キャリブレーション関連の関数

日付:

\$Date:: 2011-06-30 19:26:38 +0900 #

4.2.2 関数

4.2.2.1 void backprojectPoint ([Data_2D](#) * *icPos*, [Data_2D](#) *iPos*, [CameraParam](#) * *cameraParam*)

4.2.2.2 void distortPosition ([Data_2D](#) * *iPos2D*, [Data_2D](#) *icPos2D*, [CameraParam](#) * *cameraParam*)

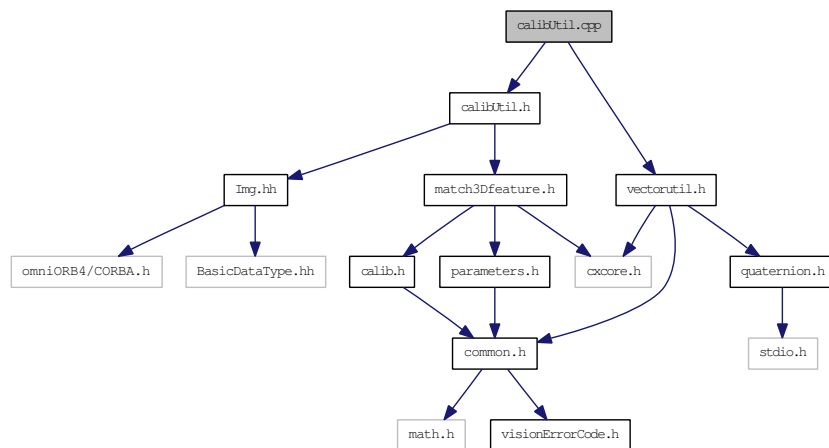
4.2.2.3 void projectPoint ([Data_2D](#) * *iPos2D*, [Data_2D](#) *icPos2D*, [CameraParam](#) * *cameraParam*)

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

4.3 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)

4.3.1 関数

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

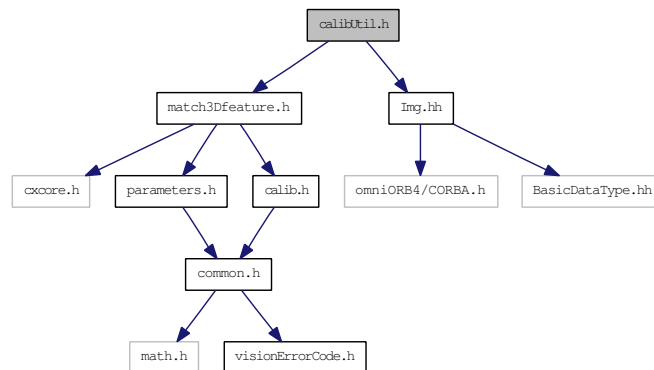
[CameraImage](#) 内のキャリブレーションデータを、Calib 構造体にセットする。

4.4 calibUtil.h

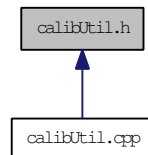
キャリブレーションデータの変換関連 `#include "match3Dfeature.h"`

`#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)

4.4.1 説明

キャリブレーションデータの変換関連

4.4.2 関数

4.4.2.1 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。

4.5 circle.cpp

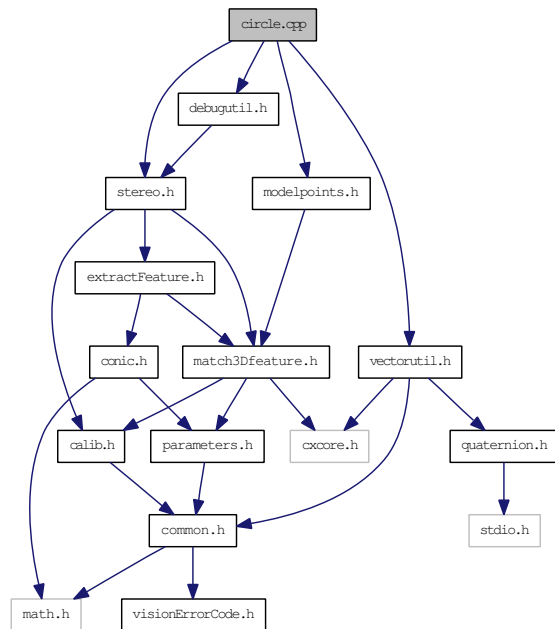
3次元円特徴生成関連関数 `#include "stereo.h"`

`#include "vectorutil.h"`

`#include "debugutil.h"`

`#include "modelpoints.h"`

circle.cpp のインクルード依存関係図



関数

- void [EllipseToCircle](#) ([StereoPairing](#) pairing, [CalibParam](#) calib, [StereoData](#) &stereo, unsigned char *edgeL, unsigned char *edgeR, [Parameters](#) parameters)

4.5.1 説明

3次元円特徴生成関連関数

日付:

\$Date:: 2011-08-04 17:04:46 +0900 # \$

4.5.2 関数

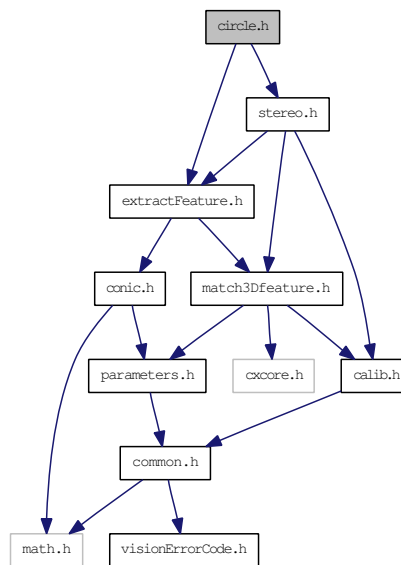
4.5.2.1 void EllipseToCircle (StereoPairing *pairing*, CalibParam *calib*, StereoData & *stereo*, unsigned char * *edgeL*, unsigned char * *edgeR*, Parameters *parameters*)

4.6 circle.h

3次元円特徴生成関連関数 `#include "extractFeature.h"`

`#include "stereo.h"`

circle.h のインクルード依存関係図



関数

- void [EllipseToCircle](#) ([StereoPairing](#) pairing, [CalibParam](#) calib, [StereoData](#) &stereo, unsigned char *edgeL, unsigned char *edgeR, [Parameters](#) parameters)

4.6.1 説明

3次元円特徴生成関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.6.2 関数

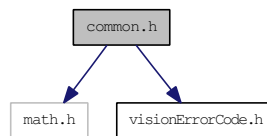
4.6.2.1 void EllipseToCircle (StereoPairing *pairing*, CalibParam *calib*, StereoData & *stereo*, unsigned char * *edgeL*, unsigned char * *edgeR*, Parameters *parameters*)

4.7 common.h

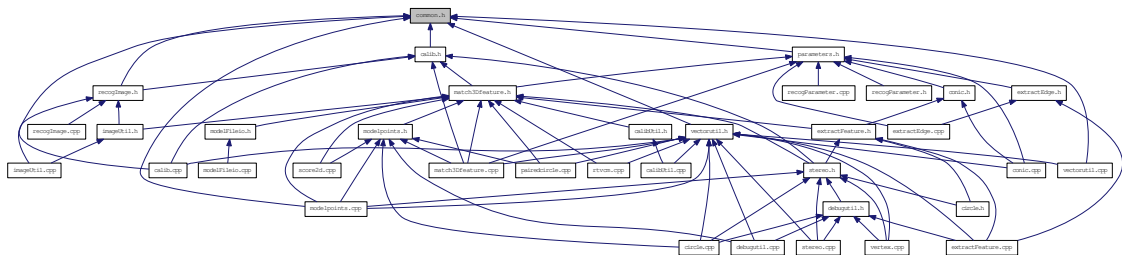
各種の共通定義 `#include <math.h>`

`#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

列挙型

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

4.7.1 説明

各種の共通定義

日付:

\$Date:: 2011-07-04 15:54:51 +0900 #

4.7.2 マクロ定義

4.7.2.1 #define INVISIBLE 0

4.7.2.2 #define VISIBLE 1

4.7.2.3 #define VISION_EPS 1.0e-10

4.7.3 列挙型

4.7.3.1 enum StereoPairing

列挙型の値:

DBL_LR
DBL_LV
DBL_RV
TBL_OR
TBL_AND

4.8 conic.cpp

二次曲線特徴抽出関連関数 `#include <math.h>`

`#include <string.h>`

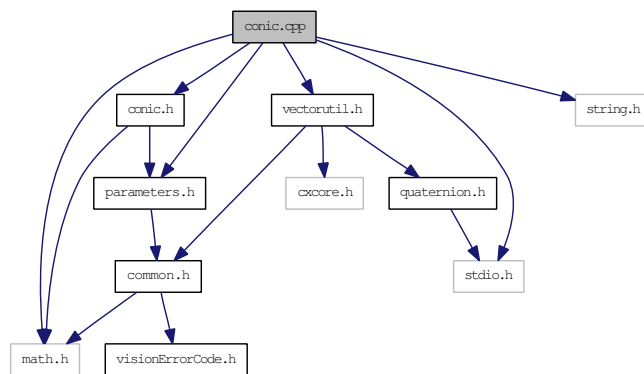
`#include <stdio.h>`

`#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)

4.8.1 説明

二次曲線特徴抽出関連関数

日付:

\$Date:: 2011-07-07 15:41:46 +0900 #

4.8.2 関数

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

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

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

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

4.8.2.5 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*)

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

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

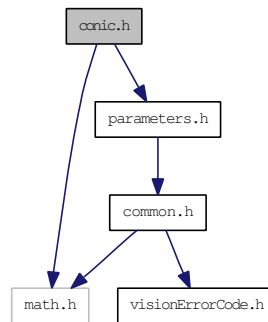
4.8.2.8 void subConicSum (double *sum*[5][5], int * *point*, double * *offset*)

4.9 conic.h

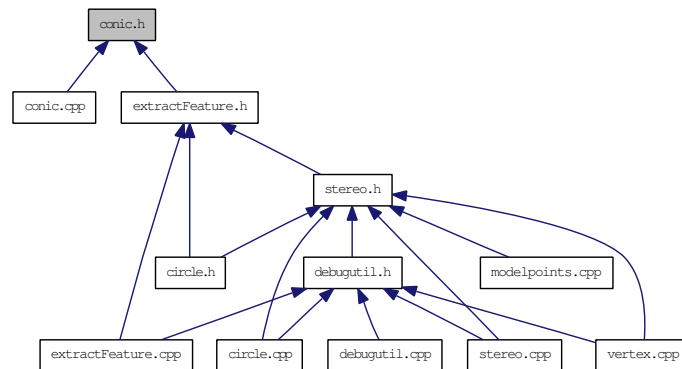
二次曲線特徴抽出関連関数 `#include <math.h>`

`#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)

4.9.1 説明

二次曲線特徴抽出関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.9.2 列挙型

4.9.2.1 enum ConicType

列挙型の値:

ConicType_Unknown

ConicType_Line

ConicType_Ellipse

ConicType_Hyperbola

ConicType_Parabola

4.9.3 関数

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

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

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

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

4.9.3.5 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*)

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

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

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

4.10 debugutil.cpp

デバッグ用関数 `#include <stdio.h>`

`#include <highgui.h>`

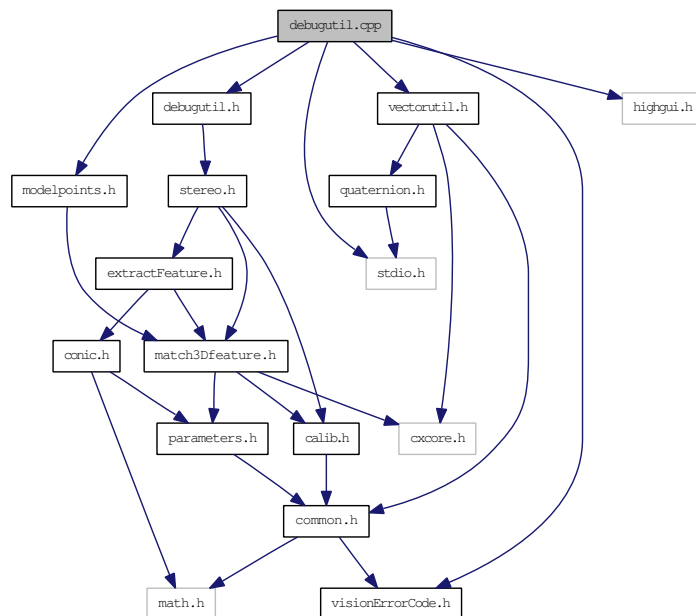
`#include "debugutil.h"`

`#include "vectorutil.h"`

`#include "visionErrorCode.h"`

`#include "modelpoints.h"`

debugutil.cpp のインクルード依存関係図



関数

- int `drawInputImage` (const uchar *src, const `Parameters` ¶meters)
- int `drawEdgeImage` (const uchar *edge, const `Parameters` ¶meters)
- int `drawDetectedLines` (const uchar *edge, const `Features2D` *lineFeatures, const `Parameters` ¶meters)
- int `drawDetectedVertices` (const `Features2D` *features, const `Parameters` ¶meters)
- int `drawTrackPoints` (const `Features2D` *features, const `Parameters` ¶meters)

- int [drawDetectedEllipses](#) (const uchar *edge, const [Features2D](#) *features, const [Parameters](#) ¶meters)
- int [drawStereoCorrespondence](#) (const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters)
- int [drawStereoVertices](#) (const uchar *edge, const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters, const [CameraParam](#) *cameraParam)
- int [drawStereoCircles](#) (const uchar *edge, const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters, const [CameraParam](#) *cameraParam)
- int [printStereoVertices](#) (const [StereoData](#) &stereo, int pairing)
- int [printStereoCircles](#) (const [StereoData](#) &stereo, int pairing)

4.10.1 説明

デバッグ用関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.10.2 関数

4.10.2.1 int [drawDetectedEllipses](#) (const uchar * *edge*, const [Features2D](#) * *features*, const [Parameters](#) & *parameters*)

4.10.2.2 int [drawDetectedLines](#) (const uchar * *edge*, const [Features2D](#) * *lineFeatures*, const [Parameters](#) & *parameters*)

4.10.2.3 int [drawDetectedVertices](#) (const [Features2D](#) * *features*, const [Parameters](#) & *parameters*)

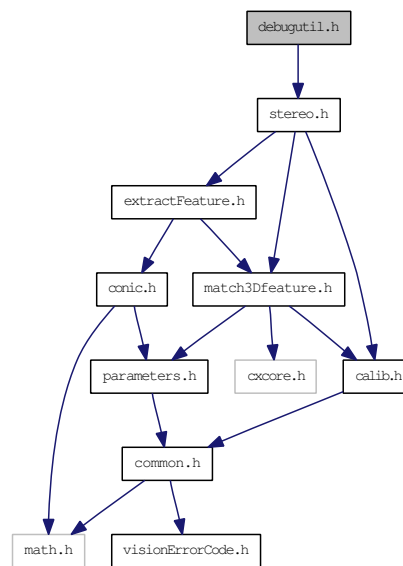
4.10.2.4 int [drawEdgeImage](#) (const uchar * *edge*, const [Parameters](#) & *parameters*)

- 4.10.2.5 **int drawInputImage** (const uchar * *src*, const Parameters & *parameters*)
- 4.10.2.6 **int drawStereoCircles** (const uchar * *edge*, const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*, const CameraParam * *cameraParam*)
- 4.10.2.7 **int drawStereoCorrespondence** (const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*)
- 4.10.2.8 **int drawStereoVertices** (const uchar * *edge*, const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*, const CameraParam * *cameraParam*)
- 4.10.2.9 **int drawTrackPoints** (const Features2D * *features*, const Parameters & *parameters*)
- 4.10.2.10 **int printStereoCircles** (const StereoData & *stereo*, int *pairing*)
- 4.10.2.11 **int printStereoVertices** (const StereoData & *stereo*, int *pairing*)

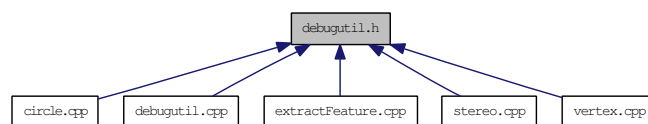
4.11 debugutil.h

デバッグ用関数 `#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。



関数

- `int drawInputImage (const uchar *src, const Parameters ¶meters)`
- `int drawEdgeImage (const uchar *edge, const Parameters ¶meters)`
- `int drawDetectedLines (const uchar *edge, const Features2D *lineFeatures, const Parameters ¶meters)`
- `int drawDetectedVertices (const Features2D *features, const Parameters ¶meters)`
- `int drawDetectedEllipses (const uchar *edge, const Features2D *features, const Parameters ¶meters)`

- int [drawTrackPoints](#) (const [Features2D](#) *features, const [Parameters](#) ¶meters)
- int [drawStereoCorrespondence](#) (const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters)
- int [drawStereoVertices](#) (const uchar *edge, const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters, const [CameraParam](#) *cameraParam)
- int [drawStereoCircles](#) (const uchar *edge, const [StereoData](#) &stereo, int pairing, const [Parameters](#) ¶meters, const [CameraParam](#) *cameraParam)
- int [printStereoVertices](#) (const [StereoData](#) &stereo, int pairing)
- int [printStereoCircles](#) (const [StereoData](#) &stereo, int pairing)

4.11.1 説明

デバッグ用関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #\$

4.11.2 関数

4.11.2.1 int [drawDetectedEllipses](#) (const uchar * *edge*, const [Features2D](#) * *features*, const [Parameters](#) & *parameters*)

4.11.2.2 int [drawDetectedLines](#) (const uchar * *edge*, const [Features2D](#) * *lineFeatures*, const [Parameters](#) & *parameters*)

4.11.2.3 int [drawDetectedVertices](#) (const [Features2D](#) * *features*, const [Parameters](#) & *parameters*)

4.11.2.4 int [drawEdgeImage](#) (const uchar * *edge*, const [Parameters](#) & *parameters*)

- 4.11.2.5 **int drawInputImage** (const uchar * *src*, const Parameters & *parameters*)

- 4.11.2.6 **int drawStereoCircles** (const uchar * *edge*, const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*, const CameraParam * *cameraParam*)

- 4.11.2.7 **int drawStereoCorrespondence** (const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*)

- 4.11.2.8 **int drawStereoVertices** (const uchar * *edge*, const StereoData & *stereo*, int *pairing*, const Parameters & *parameters*, const CameraParam * *cameraParam*)

- 4.11.2.9 **int drawTrackPoints** (const Features2D * *features*, const Parameters & *parameters*)

- 4.11.2.10 **int printStereoCircles** (const StereoData & *stereo*, int *pairing*)

- 4.11.2.11 **int printStereoVertices** (const StereoData & *stereo*, int *pairing*)

4.12 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>`

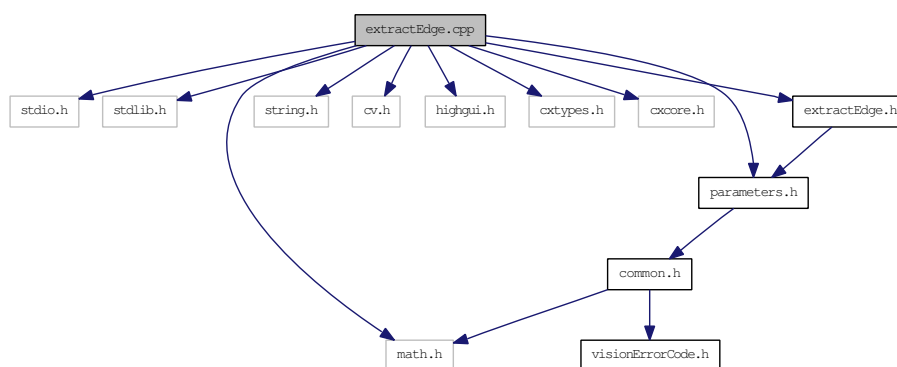
`#include <cxtypes.h>`

`#include <cxcore.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)`

4.12.1 説明

エッジ抽出関連関数

日付:

\$Date:: 2011-08-01 12:44:57 +0900 #

4.12.2 マクロ定義

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

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

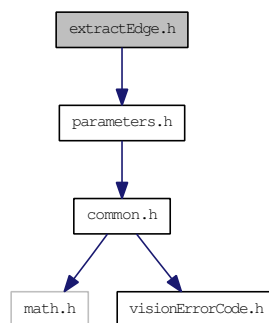
4.12.3 関数

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

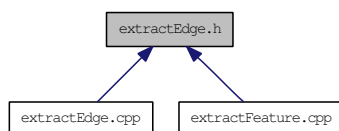
4.13 extractEdge.h

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)

4.13.1 説明

エッジ抽出関連関数

日付:

\$Date:: 2011-08-01 12:44:57 +0900 #

4.13.2 マクロ定義

4.13.2.1 #define EE6 (6)

4.13.2.2 #define EE7 (7)

4.13.2.3 #define EEcandidate (2)

4.13.2.4 #define EEerasedThin (1)

4.13.2.5 #define EEextended (5)

4.13.2.6 #define EEnotEdge (0)

4.13.2.7 #define EEnotSearched (2)

4.13.2.8 #define EEsearchedLarge (4)

4.13.2.9 #define EEsearchedSmall (3)

4.13.3 関数

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

4.14 extractFeature.cpp

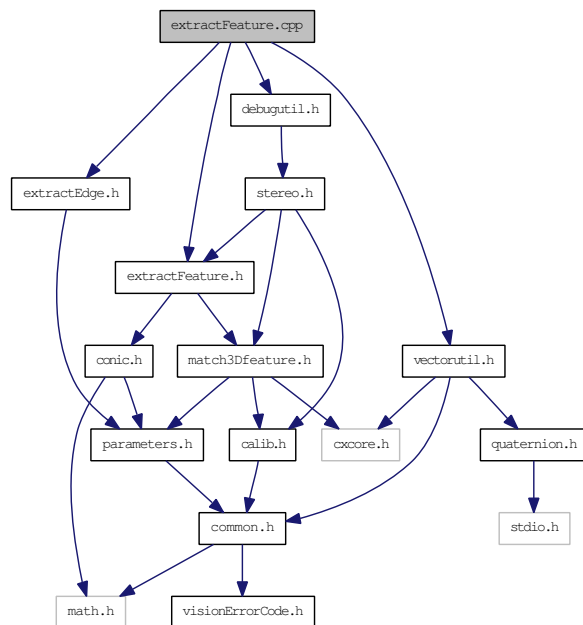
2次元特徴抽出関連関数 `#include "extractEdge.h"`

`#include "extractFeature.h"`

`#include "vectorutil.h"`

`#include "debugutil.h"`

extractFeature.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)])`

関数

- `void destructFeatures (Features2D *features)`
2次元特徴情報のメモリ解放
- `Features2D * extractFeatures (unsigned char *edge, Parameters parameters, Features3D model)`
- `Features2D * ImageToFeature2D (unsigned char *src, unsigned char *edge, Parameters parameters, Features3D model)`

ステレオ画像の一枚から二次元特徴の抽出

4.14.1 説明

2次元特徴抽出関連関数

日付:

\$Date:: 2011-08-01 12:44:57 +0900 #

4.14.2 マクロ定義

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

4.14.3 関数

4.14.3.1 `void destructFeatures (Features2D * features)`

2次元特徴情報のメモリ解放

4.14.3.2 `Features2D* extractFeatures (unsigned char * edge, Parameters parameters, Features3D model)`

4.14.3.3 `Features2D* ImageToFeature2D (unsigned char * src, unsigned char * edge, Parameters parameters, Features3D model)`

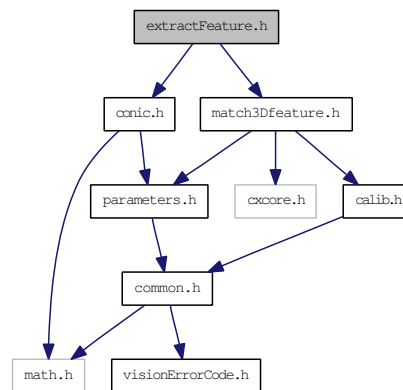
ステレオ画像の一枚から二次元特徴の抽出

4.15 extractFeature.h

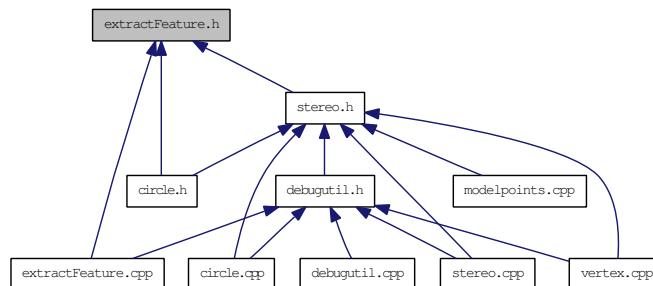
2次元特徴抽出関連関数 `#include "conic.h"`

`#include "match3Dfeature.h"`

extractFeature.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 [Feature2D](#)
各2次元特徴
- struct [Track](#)
輪郭情報
- struct [Features2D](#)

2次元特徴情報

- struct [EllipseGroup](#)
楕円重複除去用グループ情報

関数

- void [destructFeatures](#) ([Features2D](#) *features)
2次元特徴情報のメモリ解放
- [Features2D](#) * [ImageToFeature2D](#) (unsigned char *src, unsigned char *edge, [Parameters](#) parameters, [Features3D](#) model)
ステレオ画像の一枚から二次元特徴の抽出

4.15.1 説明

2次元特徴抽出関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #

4.15.2 関数

4.15.2.1 void destructFeatures ([Features2D](#) **features*)

2次元特徴情報のメモリ解放

4.15.2.2 [Features2D](#)* [ImageToFeature2D](#) (unsigned char * *src*, unsigned char * *edge*, [Parameters](#) *parameters*, [Features3D](#) *model*)

ステレオ画像の一枚から二次元特徴の抽出

4.16 imageUtil.cpp

画像入出力関数 `#include <stdio.h>`

`#include <stdlib.h>`

`#include <math.h>`

`#include <string.h>`

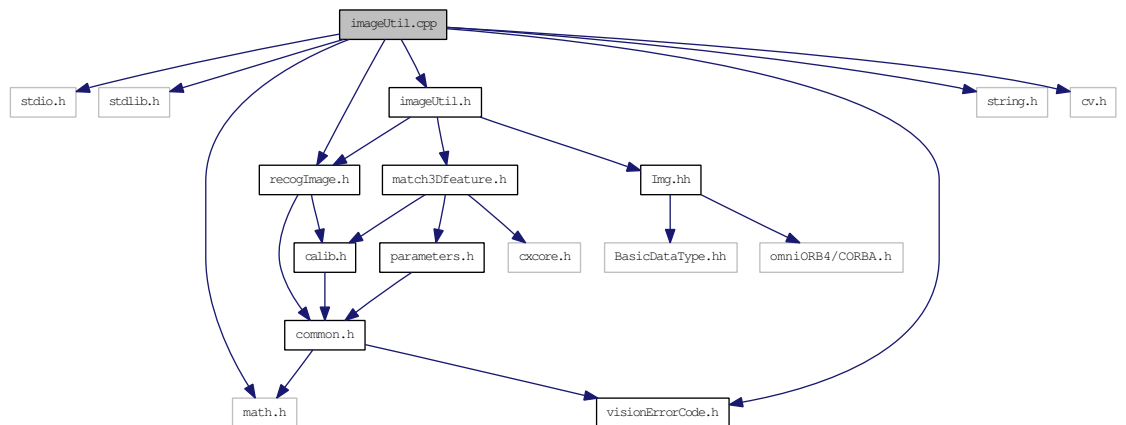
`#include <cv.h>`

`#include "recogImage.h"`

`#include "imageUtil.h"`

`#include "visionErrorCode.h"`

imageUtil.cpp のインクルード依存関係図



関数

- int [convertCameraIntrinsicParameterToCvMat](#) (const `Img::CameraIntrinsicParameter &intrinsic`, `CvMat **intrinsic_matrix`, `CvMat **distortion_coeffs`)

CameraIntrinsicParameter 構造体を *OpenCV* の *CvMat* 構造体形式に変換する。.

- int [createUndistortionMap](#) (int width, int height, const `Img::CameraIntrinsicParameter &intrinsic`, `IplImage **mapx`, `IplImage **mapy`)

CameraIntrinsicParameter 構造体から、*OpenCV* の歪み補正マップを作成する。.

- `int createUndistortionMapFromTimedMultiCameraImage` (const `Img::TimedMultiCameraImage &frame`, `IplImage ***mapx`, `IplImage ***mapy`)
`TimedMultiCameraImage` 構造体のそれぞれの画像の歪み補正マップを作成する。.
- `IplImage ** convertTimedMultiCameraImageToUndistortIplImage` (const `Img::TimedMultiCameraImage &frame`, `IplImage ***mapx`, `IplImage ***mapy`)
- `void freeUndistortIplImage` (`IplImage **resultImage`, `IplImage **undistortMapX`, `IplImage **undistortMapY`, `int imageNum`)
- `IplImage ** convertTimedMultiCameraImageToIplImage` (const `Img::TimedMultiCameraImage &frame`)
- `RecogImage ** convertTimedMultiCameraImageToRecogImage` (const `Img::TimedMultiCameraImage &frame`)
- `void freeConvertedRecogImage` (`RecogImage **recogImage`, `int imageNum`)

4.16.1 説明

画像入出力関数

日付:

\$Date:: 2011-08-03 08:35:23 +0900 # \$

4.16.2 関数

- 4.16.2.1** `int convertCameraIntrinsicParameterToCvMat` (const `Img::CameraIntrinsicParameter &intrinsic`, `CvMat **intrinsic_matrix`, `CvMat **distortion_coeffs`)

`CameraIntrinsicParameter` 構造体を OpenCV の `CvMat` 構造体形式に変換する。.

- 4.16.2.2** `IplImage** convertTimedMultiCameraImageToIplImage` (const `Img::TimedMultiCameraImage &frame`)

`TimedMultiCameraImage` 画像データを OpenCV の `IplImage` 構造体形式に変換する。`ImageData::raw_data` は、TopLeft、RGB 順、行単位での padding なし。画像 1 枚ごとに個別の `IplImage` 構造体を作成する。画像は channel 数 3 で確保する。

- 4.16.2.3** `RecogImage** convertTimedMultiCameraImageToRecogImage` (const `Img::TimedMultiCameraImage &frame`)

`TimedMultiCameraImage` 画像データを、`RecogImage` 構造体形式に変換する。カラー画像はグレイ画像に変換する。

4.16.2.4 `IplImage** convertTimedMultiCameraImageToUndistortIplImage` (const `Img::TimedMultiCameraImage & frame`, `IplImage *** mapx`, `IplImage *** mapy`)

`TimedMultiCameraImage` 画像データを、歪みを補正して OpenCV の `IplImage` 構造体形式に変換する。`ImageData::raw_data` は、TopLeft、RGB 順、行単位での padding なし。画像 1 枚ごとに個別の `IplImage` 構造体を作成する。画像は channel 数 3 で確保する。

4.16.2.5 `int createUndistortionMap` (int *width*, int *height*, const `Img::CameraIntrinsicParameter & intrinsic`, `IplImage ** mapx`, `IplImage ** mapy`)

`CameraIntrinsicParameter` 構造体から、OpenCV の歪み補正マップを作成する。.

4.16.2.6 `int createUndistortionMapFromTimedMultiCameraImage` (const `Img::TimedMultiCameraImage & frame`, `IplImage *** mapx`, `IplImage *** mapy`)

`TimedMultiCameraImage` 構造体のそれぞれの画像の歪み補正マップを作成する。.

4.16.2.7 `void freeConvertedRecogImage` (`RecogImage ** recogImage`, int *imageNum*)

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

4.16.2.8 `void freeUndistortIplImage` (`IplImage ** resultImage`, `IplImage **` *undistortMapX*, `IplImage ** undistortMapY`, int *imageNum*)

`convertTimedMultiCameraImageToUndistortIplImage` によって 確保されたメモリを開放する

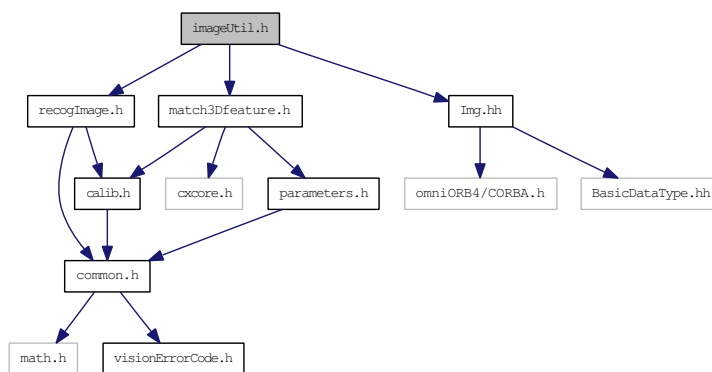
4.17 imageUtil.h

画像入出力関連 `#include "recogImage.h"`

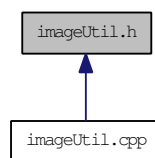
`#include "match3Dfeature.h"`

`#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

- `int createUndistortionMapFromTimedMultiCameraImage` (const `Img::TimedMultiCameraImage &frame`, `IplImage ***mapx`, `IplImage ***mapy`)
`TimedMultiCameraImage` 構造体のそれぞれの画像の歪み補正マップを作成する。.
- `IplImage ** convertTimedMultiCameraImageToUndistortIplImage` (const `Img::TimedMultiCameraImage &frame`, `IplImage ***mapx`, `IplImage ***mapy`)
- `void freeUndistortIplImage` (`IplImage **resultImage`, `IplImage **undistortMapX`, `IplImage **undistortMapY`, `int imageNum`)

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

4.17.1 説明

画像入出力関連

日付:

\$Date:: 2011-07-26 18:34:36 +0900 # \$

4.17.2 関数

4.17.2.1 `IplImage** convertTimedMultiCameraImageToIplImage` (const `Img::TimedMultiCameraImage & frame`)

`TimedMultiCameraImage` 画像データを、OpenCV の `IplImage` 構造体形式に変換する。画像 1 枚ごとに個別の `IplImage` 構造体を作成する。画像は、channel 数 3 で確保する。

`TimedMultiCameraImage` 画像データを OpenCV の `IplImage` 構造体形式に変換する。`ImageData::raw_data` は、TopLeft、RGB 順、行単位での padding なし。画像 1 枚ごとに個別の `IplImage` 構造体を作成する。画像は channel 数 3 で確保する。

4.17.2.2 `RecogImage** convertTimedMultiCameraImageToRecogImage` (const `Img::TimedMultiCameraImage & frame`)

`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。カラー画像はグレイ画像に変換する。

4.17.2.3 `IplImage** convertTimedMultiCameraImageToUndistortIplImage` (const `Img::TimedMultiCameraImage & frame`, `IplImage *** mapx`, `IplImage *** mapy`)

`TimedMultiCameraImage` 画像データを、歪み補正をして OpenCV の `IplImage` 構造体形式に変換する。画像 1 枚ごとに個別の `IplImage` 構造体を作成する。画像は、channel 数 3 で確保する。

`TimedMultiCameraImage` 画像データを、歪みを補正して OpenCV の `IplImage` 構造体形式に変換する。`ImageData::raw_data` は、TopLeft、RGB 順、行単位での

padding なし。画像 1 枚ごとに個別の IplImage 構造体を作成する。画像は channel 数 3 で確保する。

4.17.2.4 `int createUndistortionMapFromTimedMultiCameraImage (const
Img::TimedMultiCameraImage & frame, IplImage *** mapx,
IplImage *** mapy)`

[TimedMultiCameraImage](#) 構造体のそれぞれの画像の歪み補正マップを作成する。.

4.17.2.5 `void freeConvertedRecogImage (RecogImage ** recogImage, int
imageNum)`

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

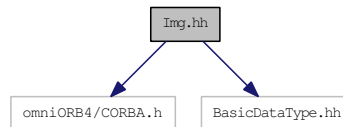
4.17.2.6 `void freeUndistortIplImage (IplImage ** resultImage, IplImage **
undistortMapX, IplImage ** undistortMapY, int imageNum)`

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

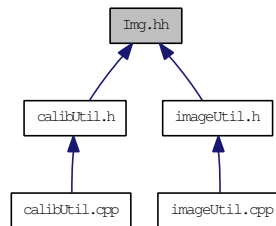
4.18 Img.hh

```
#include <omniORB4/CORBA.h>
#include <BasicDataType.hh>
```

Img.hh 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

- class [Vec3_copyHelper](#)
- class [Mat44_copyHelper](#)
- struct [ImageData](#)
- struct [CameraIntrinsicParameter](#)
- struct [CameraImage](#)
- struct [TimedCameraImage](#)
- struct [MultiCameraImage](#)
- struct [TimedMultiCameraImage](#)
- class [CameraCaptureService_Helper](#)
- class [CameraCaptureService](#)
- class [_objref_CameraCaptureService](#)
- class [_pof_CameraCaptureService](#)
- class [_impl_CameraCaptureService](#)
- class [CameraCaptureService](#)

マクロ定義

- #define USE_stub_in_nt_dll_NOT_DEFINED_Img
- #define USE_core_stub_in_nt_dll_NOT_DEFINED_Img
- #define USE_dyn_stub_in_nt_dll_NOT_DEFINED_Img
- #define _core_attr
- #define _dyn_attr

型定義

- typedef CORBA::Double Vec3 [3]
- typedef CORBA::Double Vec3_slice
- typedef _CORBA_Array_Fix_Var< Vec3_copyHelper, Vec3_slice > Vec3_var
- typedef _CORBA_Array_Fix_Forany< Vec3_copyHelper, Vec3_slice > Vec3_forany
- typedef Vec3_slice * Vec3_out
- typedef CORBA::Double Mat44 [4][4]
- typedef CORBA::Double Mat44_slice [4]
- typedef _CORBA_Array_Fix_Var< Mat44_copyHelper, Mat44_slice > Mat44_var
- typedef _CORBA_Array_Fix_Forany< Mat44_copyHelper, Mat44_slice > Mat44_forany
- typedef Mat44_slice * Mat44_out
- typedef ColorFormat & ColorFormat_out
- typedef ImageData::_var_type ImageData_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< ImageData, ImageData_var > ImageData_out
- typedef CameraIntrinsicParameter::_var_type CameraIntrinsicParameter_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< CameraIntrinsicParameter, CameraIntrinsicParameter_var > CameraIntrinsicParameter_out
- typedef CameraImage::_var_type CameraImage_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< CameraImage, CameraImage_var > CameraImage_out
- typedef TimedCameraImage::_var_type TimedCameraImage_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< TimedCameraImage, TimedCameraImage_var > TimedCameraImage_out
- typedef MultiCameraImage::_var_type MultiCameraImage_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< MultiCameraImage, MultiCameraImage_var > MultiCameraImage_out
- typedef TimedMultiCameraImage::_var_type TimedMultiCameraImage_var
- typedef _CORBA_ConstrType_Variable_OUT_arg< TimedMultiCameraImage, TimedMultiCameraImage_var > TimedMultiCameraImage_out
- typedef _objref_CameraCaptureService * CameraCaptureService_ptr
- typedef CameraCaptureService_ptr CameraCaptureServiceRef

- typedef `_CORBA_ObjRef_Var< _objref_CameraCaptureService, CameraCaptureService_Helper > CameraCaptureService_var`
- typedef `_CORBA_ObjRef_OUT_arg< _objref_CameraCaptureService, CameraCaptureService_Helper > CameraCaptureService_out`

列挙型

- enum `ColorFormat { CF_UNKNOWN, CF_GRAY, CF_RGB }`

関数

- `_CORBA_MODULE_INLINE Vec3_slice * Vec3_alloc ()`
- `_CORBA_MODULE_INLINE Vec3_slice * Vec3_dup (const Vec3_slice *_s)`
- `_CORBA_MODULE_INLINE void Vec3_copy (Vec3_slice *_to, const Vec3_slice *_from)`
- `_CORBA_MODULE_INLINE void Vec3_free (Vec3_slice *_s)`
- `_CORBA_MODULE_INLINE Mat44_slice * Mat44_alloc ()`
- `_CORBA_MODULE_INLINE Mat44_slice * Mat44_dup (const Mat44_slice *_s)`
- `_CORBA_MODULE_INLINE void Mat44_copy (Mat44_slice *_to, const Mat44_slice *_from)`
- `_CORBA_MODULE_INLINE void Mat44_free (Mat44_slice *_s)`
- `_CORBA_MODULE_END _CORBA_MODULE_OBV_Img _CORBA_MODULE_BEG _CORBA_MODULE_END void operator<=<= (CORBA::Any &_a, const Img::Vec3_forany &_s)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, Img::Vec3_forany &_s)`
- `void operator<=<= (CORBA::Any &_a, const Img::Mat44_forany &_s)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, Img::Mat44_forany &_s)`
- `void operator>>= (Img::ColorFormat _e, cdrStream &s)`
- `void operator<=<= (Img::ColorFormat &_e, cdrStream &s)`
- `void operator<=<= (CORBA::Any &_a, Img::ColorFormat _s)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, Img::ColorFormat &_s)`
- `void operator<=<= (CORBA::Any &_a, const Img::ImageData &_s)`
- `void operator<=<= (CORBA::Any &_a, Img::ImageData *_sp)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, Img::ImageData *&_sp)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, const Img::ImageData *&_sp)`
- `void operator<=<= (CORBA::Any &_a, const Img::CameraIntrinsicParameter &_s)`
- `void operator<=<= (CORBA::Any &_a, Img::CameraIntrinsicParameter *_sp)`
- `CORBA::Boolean operator>>= (const CORBA::Any &_a, Img::CameraIntrinsicParameter *&_sp)`

- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, const Img::CameraIntrinsicParameter *&_sp)
- void [operator<<=](#) (CORBA::Any &_a, const Img::CameraImage &_s)
- void [operator<<=](#) (CORBA::Any &_a, Img::CameraImage *_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, Img::CameraImage *&_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, const Img::CameraImage *&_sp)
- void [operator<<=](#) (CORBA::Any &_a, const Img::TimedCameraImage &_s)
- void [operator<<=](#) (CORBA::Any &_a, Img::TimedCameraImage *_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, Img::TimedCameraImage *&_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, const Img::TimedCameraImage *&_sp)
- void [operator<<=](#) (CORBA::Any &_a, const Img::MultiCameraImage &_s)
- void [operator<<=](#) (CORBA::Any &_a, Img::MultiCameraImage *_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, Img::MultiCameraImage *&_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, const Img::MultiCameraImage *&_sp)
- void [operator<<=](#) (CORBA::Any &_a, const Img::TimedMultiCameraImage &_s)
- void [operator<<=](#) (CORBA::Any &_a, Img::TimedMultiCameraImage *_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, Img::TimedMultiCameraImage *&_sp)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, const Img::TimedMultiCameraImage *&_sp)
- void [operator<<=](#) (CORBA::Any &_a, [Img::CameraCaptureService_ptr](#) _s)
- void [operator<<=](#) (CORBA::Any &_a, [Img::CameraCaptureService_ptr](#) *_s)
- CORBA::Boolean [operator>>=](#) (const CORBA::Any &_a, [Img::CameraCaptureService_ptr](#) &_s)

変数

- [_CORBA_MODULE](#) [Img](#) [_CORBA_MODULE_BEG](#) [_CORBA_MODULE_VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_Vec3](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_Mat44](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_ColorFormat](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_ImageData](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_CameraIntrinsicParameter](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_CameraImage](#)
- [_CORBA_MODULE](#) [VAR](#) [_dyn_attr](#) const CORBA::TypeCode_ptr [_tc_TimedCameraImage](#)

- `_CORBA_MODULE_VAR` `_dyn_attr` `const` `CORBA::TypeCode_ptr`
`_tc_MultiCameraImage`
- `_CORBA_MODULE_VAR` `_dyn_attr` `const` `CORBA::TypeCode_ptr`
`_tc_TimedMultiCameraImage`
- `_CORBA_MODULE_VAR` `_dyn_attr` `const` `CORBA::TypeCode_ptr`
`_tc_CameraCaptureService`

4.18.1 マクロ定義

4.18.1.1 `#define _core_attr`

4.18.1.2 `#define _dyn_attr`

4.18.1.3 `#define USE_core_stub_in_nt_dll_NOT_DEFINED_Img`

4.18.1.4 `#define USE_dyn_stub_in_nt_dll_NOT_DEFINED_Img`

4.18.1.5 `#define USE_stub_in_nt_dll_NOT_DEFINED_Img`

4.18.2 型定義

4.18.2.1 `typedef _CORBA_ObjRef_OUT_arg<_objref_ CameraCaptureService, CameraCaptureService_Helper > CameraCaptureService_out`

4.18.2.2 `typedef _objref_CameraCaptureService* CameraCaptureService_ptr`

4.18.2.3 `typedef _CORBA_ObjRef_Var<_objref_CameraCaptureService,
CameraCaptureService_Helper> CameraCaptureService_var`

4.18.2.4 `typedef CameraCaptureService_ptr CameraCaptureServiceRef`

4.18.2.5 `typedef _CORBA_ConstrType_Variable_OUT_arg<
CameraImage, CameraImage_var > CameraImage_out`

4.18.2.6 `typedef CameraImage::_var_type CameraImage_var`

4.18.2.7 `typedef _CORBA_ConstrType_Variable_OUT_arg<
CameraIntrinsicParameter, CameraIntrinsicParameter_var >
CameraIntrinsicParameter_out`

4.18.2.8 `typedef CameraIntrinsicParameter::_var_type
CameraIntrinsicParameter_var`

4.18.2.9 `typedef ColorFormat& ColorFormat_out`

4.18.2.10 `typedef _CORBA_ConstrType_Variable_OUT_arg<
ImageData, ImageData_var > ImageData_out`

4.18.2.11 `typedef ImageData::_var_type ImageData_var`

4.18.2.12 `typedef CORBA::Double Mat44[4][4]`

4.18.2.13 `typedef _CORBA_Array_Fix_Forany<Mat44_copyHelper,Mat44_slice> Mat44_forany`

4.18.2.14 `typedef Mat44_slice* Mat44_out`

4.18.2.15 `typedef CORBA::Double Mat44_slice[4]`

4.18.2.16 `typedef _CORBA_Array_Fix_Var<Mat44_copyHelper,Mat44_slice> Mat44_var`

4.18.2.17 `typedef _CORBA_ConstrType_Variable_OUT_arg<MultiCameraImage,MultiCameraImage_var > MultiCameraImage_out`

4.18.2.18 `typedef MultiCameraImage::_var_type MultiCameraImage_var`

4.18.2.19 `typedef _CORBA_ConstrType_Variable_OUT_arg<TimedCameraImage,TimedCameraImage_var > TimedCameraImage_out`

4.18.2.20 `typedef TimedCameraImage::_var_type TimedCameraImage_var`

4.18.2.21 `typedef _CORBA_ConstrType_Variable_OUT_arg<
TimedMultiCameraImage, TimedMultiCameraImage_var >
TimedMultiCameraImage_out`

4.18.2.22 `typedef TimedMultiCameraImage::_var_type
TimedMultiCameraImage_var`

4.18.2.23 `typedef CORBA::Double Vec3[3]`

4.18.2.24 `typedef _CORBA_Array_Fix_Forany<Vec3_copyHelper, Vec3_slice>
Vec3_forany`

4.18.2.25 `typedef Vec3_slice* Vec3_out`

4.18.2.26 `typedef CORBA::Double Vec3_slice`

4.18.2.27 `typedef _CORBA_Array_Fix_Var<Vec3_copyHelper, Vec3_slice>
Vec3_var`

4.18.3 列挙型

4.18.3.1 enum ColorFormat

列挙型の値:

CF_UNKNOWN

CF_GRAY

CF_RGB

4.18.4 関数

4.18.4.1 `_CORBA_MODULE_INLINE Mat44_slice* Mat44_alloc ()`

4.18.4.2 `_CORBA_MODULE_INLINE void Mat44_copy (Mat44_slice * _to,
const Mat44_slice * _from)`

4.18.4.3 `_CORBA_MODULE_INLINE Mat44_slice* Mat44_dup (const
Mat44_slice * _s)`

4.18.4.4 `_CORBA_MODULE_INLINE void Mat44_free (Mat44_slice * _s)`

4.18.4.5 `void operator<<= (CORBA::Any & _a,
Img::CameraCaptureService_ptr * _s)`

4.18.4.6 `void operator<<= (CORBA::Any & _a,
Img::CameraCaptureService_ptr _s)`

4.18.4.7 `void operator<<= (CORBA::Any & _a,
Img::TimedMultiCameraImage * _sp)`

4.18.4.8 `void operator<<= (CORBA::Any & _a, const
Img::TimedMultiCameraImage & _s)`

- 4.18.4.9** void operator<<= (CORBA::Any & _a, Img::MultiCameraImage *
_sp)
- 4.18.4.10** void operator<<= (CORBA::Any & _a, const
Img::MultiCameraImage & _s)
- 4.18.4.11** void operator<<= (CORBA::Any & _a, Img::TimedCameraImage *
_sp)
- 4.18.4.12** void operator<<= (CORBA::Any & _a, const
Img::TimedCameraImage & _s)
- 4.18.4.13** void operator<<= (CORBA::Any & _a, Img::CameraImage * _sp)
- 4.18.4.14** void operator<<= (CORBA::Any & _a, const Img::CameraImage &
_s)
- 4.18.4.15** void operator<<= (CORBA::Any & _a,
Img::CameraIntrinsicParameter * _sp)
- 4.18.4.16** void operator<<= (CORBA::Any & _a, const
Img::CameraIntrinsicParameter & _s)
- 4.18.4.17** void operator<<= (CORBA::Any & _a, Img::ImageData * _sp)

4.18.4.18 void operator<<= (CORBA::Any & _a, const Img::ImageData & _s)

4.18.4.19 void operator<<= (CORBA::Any & _a, Img::ColorFormat _s)

4.18.4.20 void operator<<= (Img::ColorFormat & _e, cdrStream & s)
[inline]

4.18.4.21 void operator<<= (CORBA::Any & _a, const Img::Mat44_forany & _s)

4.18.4.22 _CORBA_MODULE_END _CORBA_MODULE_OBV_Img
_CORBA_MODULE_BEG _CORBA_MODULE_END void
operator<<= (CORBA::Any & _a, const Img::Vec3_forany & _s)

4.18.4.23 CORBA::Boolean operator>>= (const CORBA::Any & _a,
Img::CameraCaptureService_ptr & _s)

4.18.4.24 CORBA::Boolean operator>>= (const CORBA::Any & _a, const
Img::TimedMultiCameraImage *& _sp)

4.18.4.25 CORBA::Boolean operator>>= (const CORBA::Any & _a,
Img::TimedMultiCameraImage *& _sp)

4.18.4.26 CORBA::Boolean operator>>= (const CORBA::Any & _a, const
Img::MultiCameraImage *& _sp)

- 4.18.4.27 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*,
Img::MultiCameraImage *& *_sp*)
- 4.18.4.28 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*, const
Img::TimedCameraImage *& *_sp*)
- 4.18.4.29 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*,
Img::TimedCameraImage *& *_sp*)
- 4.18.4.30 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*, const
Img::CameraImage *& *_sp*)
- 4.18.4.31 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*,
Img::CameraImage *& *_sp*)
- 4.18.4.32 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*, const
Img::CameraIntrinsicParameter *& *_sp*)
- 4.18.4.33 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*,
Img::CameraIntrinsicParameter *& *_sp*)
- 4.18.4.34 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*, const
Img::ImageData *& *_sp*)
- 4.18.4.35 **CORBA::Boolean operator>>=** (const CORBA::Any & *_a*,
Img::ImageData *& *_sp*)

4.18.4.36 `CORBA::Boolean operator>>= (const CORBA::Any & _a,
Img::ColorFormat & _s)`

4.18.4.37 `void operator>>= (Img::ColorFormat _e, cdrStream & s)
[inline]`

4.18.4.38 `CORBA::Boolean operator>>= (const CORBA::Any & _a,
Img::Mat44_forany & _s)`

4.18.4.39 `CORBA::Boolean operator>>= (const CORBA::Any & _a,
Img::Vec3_forany & _s)`

4.18.4.40 `_CORBA_MODULE_INLINE Vec3_slice* Vec3_alloc ()`

4.18.4.41 `_CORBA_MODULE_INLINE void Vec3_copy (Vec3_slice * _to,
const Vec3_slice * _from)`

4.18.4.42 `_CORBA_MODULE_INLINE Vec3_slice* Vec3_dup (const
Vec3_slice * _s)`

4.18.4.43 `_CORBA_MODULE_INLINE void Vec3_free (Vec3_slice * _s)`

4.18.5 変数

4.18.5.1 `_CORBA_MODULE_VAR _dyn_attr const CORBA::TypeCode_ptr
_tc_CameraCaptureService`

4.18.5.2 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_CameraImage`

4.18.5.3 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_CameraIntrinsicParameter`

4.18.5.4 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_ColorFormat`

4.18.5.5 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_ImageData`

4.18.5.6 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_Mat44`

4.18.5.7 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_MultiCameraImage`

4.18.5.8 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_TimedCameraImage`

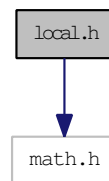
4.18.5.9 `_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_TimedMultiCameraImage`

4.18.5.10 `_CORBA_MODULE Img _CORBA_MODULE_BEG
_CORBA_MODULE_VAR_dyn_attr const CORBA::TypeCode_ptr
_tc_Vec3`

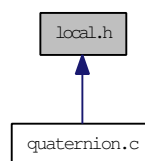
4.19 local.h

```
#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。

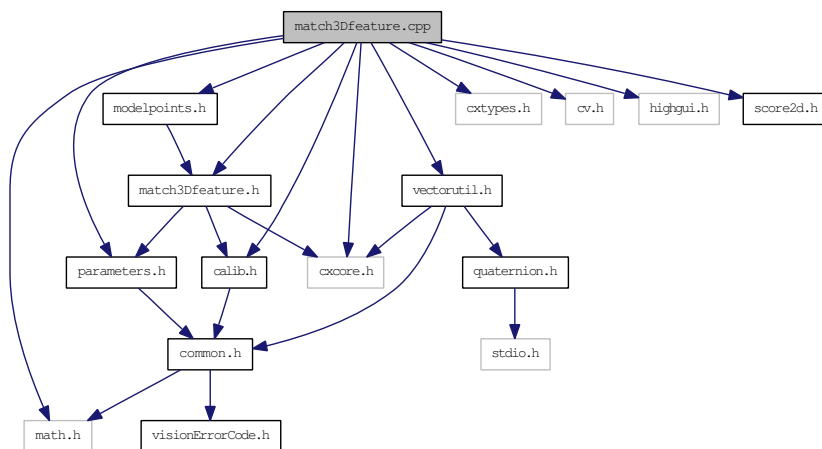


4.20 match3Dfeature.cpp

3次元特徴による認識関連関数

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

- [Match3Dresults](#) [matchFeatures3d](#) ([Features3D](#) &scene, [Features3D](#) &model, unsigned char *edgeL, unsigned char *edgeR, unsigned char *edgeV, [Parameters](#) ¶meters)

4.20.1 説明

3次元特徴による認識関連関数

日付:

\$Date:: 2011-08-08 17:46:34 +0900 # \$

4.20.2 関数

4.20.2.1 void freeFeatures3D (Features3D * *feature*)

3次元特徴データのメモリ解放

4.20.2.2 void freeMatch3Dresults (Match3Dresults * *holder*)

認識結果データのメモリ解除

4.20.2.3 Match3Dresults matchFeatures3d (Features3D & *scene*, Features3D & *model*, unsigned char * *edgeL*, unsigned char * *edgeR*, unsigned char * *edgeV*, 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

```
#include <cxcore.h>
#include "parameters.h"
#include "calib.h"
```

```
graph TD; match3Dfeature.h --> cxcore.h; match3Dfeature.h --> parameters.h; match3Dfeature.h --> calib.h; parameters.h --> common.h; calib.h --> common.h; common.h --> math.h; common.h --> visionErrorCode.h;
```

- 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 [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

列挙型

- enum [m3df_side](#) { [M3DF_FRONT](#) = 0, [M3DF_BACK](#) = 1 }

表裏を表す定数

関数

- void [freeMatch3Dresults](#) ([Match3Dresults](#) *holder)

認識結果データのメモリ解除

- void [freeFeatures3D](#) ([Features3D](#) *feature)

3次元特徴データのメモリ解放

- [Match3Dresults](#) [matchFeatures3d](#) ([Features3D](#) &scene, [Features3D](#) &model, unsigned char *edgeL, unsigned char *edgeR, unsigned char *edgeV, [Parameters](#) ¶meters)

- [Match3Dresults](#) [matchPairedCircles](#) ([Features3D](#) &scene, [Features3D](#) &model, double tolerance, [StereoPairing](#) &pairing)

4.21.1 説明

3次元特徴による認識関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #

4.21.2 列挙型

4.21.2.1 enum m3df_side

表裏を表す定数

列挙型の値:

M3DF_FRONT

M3DF_BACK

4.21.3 関数

4.21.3.1 void freeFeatures3D (Features3D * *feature*)

3次元特徴データのメモリ解放

4.21.3.2 void freeMatch3Dresults (Match3Dresults * *holder*)

認識結果データのメモリ解除

4.21.3.3 Match3Dresults matchFeatures3d (Features3D & *scene*, Features3D & *model*, unsigned char * *edgeL*, unsigned char * *edgeR*, unsigned char * *edgeV*, 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

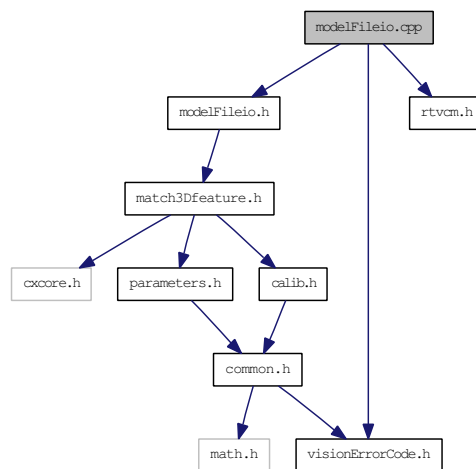
4.21.3.4 Match3Dresults matchPairedCircles (Features3D & *scene*, Features3D & *model*, double *tolerance*, StereoPairing & *pairing*)

2円を使った照合 戻り値: 認識結果

4.22 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。

4.22.1 関数

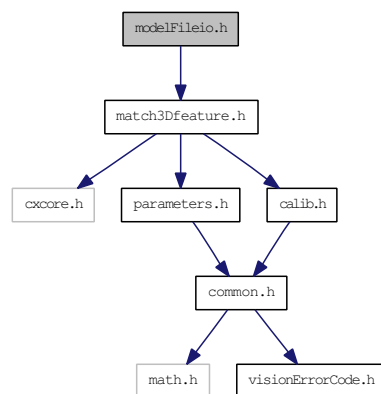
4.22.1.1 int loadModelFile (char * path, Features3D & model)

モデルデータをファイルから読み込む。

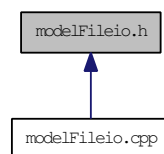
4.23 modelFileio.h

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。

4.23.1 説明

モデルをファイルから読み込む。

4.23.2 関数

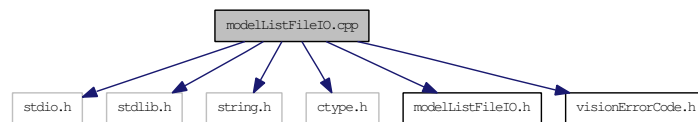
4.23.2.1 int loadModelFile (char * *path*, Features3D & *model*)

モデルデータをファイルから読み込む。

4.24 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。

4.24.1 関数

4.24.1.1 void clearModelFileInfo (ModelFileInfo * mfInfo)

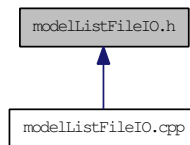
モデルリストをクリアする。

4.24.1.2 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。

4.25 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。

4.25.1 説明

モデルファイルの入出力関連

4.25.2 マクロ定義

4.25.2.1 #define MAX_PATH 256

4.25.3 関数

4.25.3.1 void clearModelFileInfo (ModelFileInfo * *mfInfo*)

モデルリストをクリアする。

4.25.3.2 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。

4.26 modelpoints.cpp

モデル評価点生成関連関数 `#include <math.h>`

`#include <cv.h>`

`#include <highgui.h>`

`#include "common.h"`

`#include "quaternion.h"`

`#include "stereo.h"`

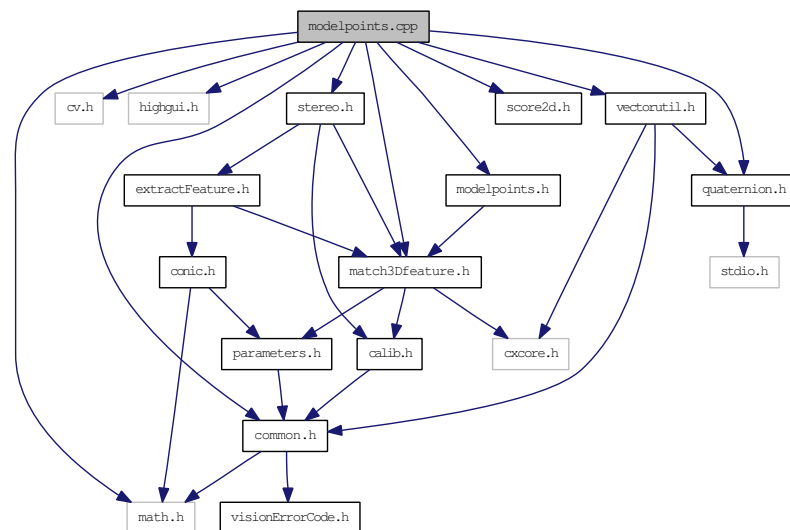
`#include "match3Dfeature.h"`

`#include "score2d.h"`

`#include "vectorutil.h"`

`#include "modelpoints.h"`

modelpoints.cpp のインクルード依存関係図



列挙型

- enum `Azimuth` {
`AZIMUTH_S = 0, AZIMUTH_SE = 1, AZIMUTH_E = 2, AZIMUTH_NE = 3,`
`AZIMUTH_N = 4, AZIMUTH_NW = 5, AZIMUTH_W = 6, AZIMUTH_SW`
`= 7,`
`AZIMUTH_NONE = -1 }`

関数

- int `getPointOnCircle` (double normal[3], double radius, double point[3])
円の始点座標計算
- 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 `traceModelPointsMultiCameras` (`Features3D` *model, `StereoPairing` &pairing, double matrix[4][4])

4.26.1 説明

モデル評価点生成関連関数

日付:

\$Date:: 2011-08-08 17:46:34 +0900 # \$

4.26.2 列挙型

4.26.2.1 enum Azimuth

列挙型の値:

AZIMUTH_S
AZIMUTH_SE
AZIMUTH_E
AZIMUTH_NE
AZIMUTH_N
AZIMUTH_NW
AZIMUTH_W
AZIMUTH_SW
AZIMUTH_NONE

4.26.3 関数

4.26.3.1 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）

4.26.3.2 int getPointOnCircle (double *normal*[3], double *radius*, double *point*[3])

円の始点座標計算

4.26.3.3 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

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

モデルの評価点の生成 戻り値：総評価点数

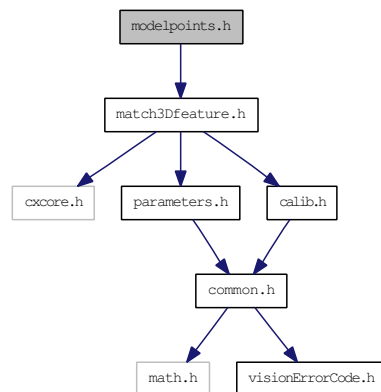
4.26.3.5 double traceModelPointsMultiCameras (Features3D * *model*, StereoPairing & *pairing*, double *matrix*[4][4])

使用した全画像を用いた 2 次元評価値計算 戻り値：2 次元評価値

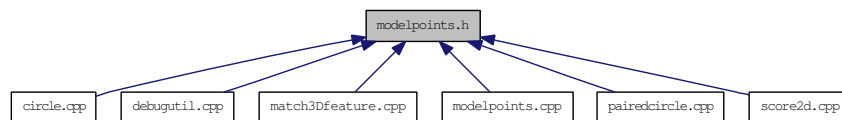
4.27 modelpoints.h

モデル評価点生成関連関数 `#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。



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 [traceModelPointsMultiCameras](#) ([Features3D](#) *model, [StereoPairing](#) &pairing, double matrix[4][4])
- int [getPointOnCircle](#) (double normal[3], double radius, double point[3])
円の始点座標計算

4.27.1 説明

モデル評価点生成関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #

4.27.2 関数

4.27.2.1 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）

4.27.2.2 int getPointOnCircle (double *normal*[3], double *radius*, double *point*[3])

円の始点座標計算

4.27.2.3 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

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

モデルの評価点の生成 戻り値：総評価点数

4.27.2.5 double traceModelPointsMultiCameras (Features3D * *model*, StereoPairing & *pairing*, double *matrix*[4][4])

使用した全画像を用いた 2 次元評価値計算 戻り値：2 次元評価値

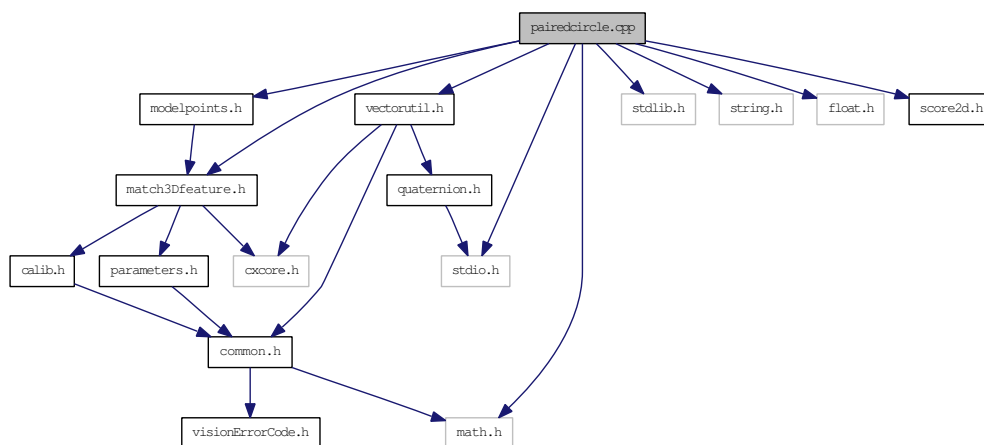
4.28 pairedcircle.cpp

```

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, double tolerance, [StereoPairing](#) &pairing)

4.28.1 説明

2 円照合関連関数

日付:

\$Date:: 2011-07-06 08:47:29 +0900 #\$

4.28.2 関数

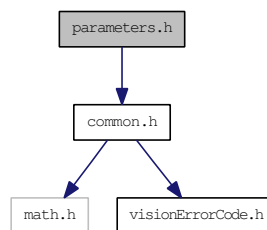
4.28.2.1 Match3Dresults matchPairedCircles (Features3D & *scene*, Features3D & *model*, *double tolerance*, StereoPairing & *pairing*)

2 円を使った照合 戻り値：認識結果

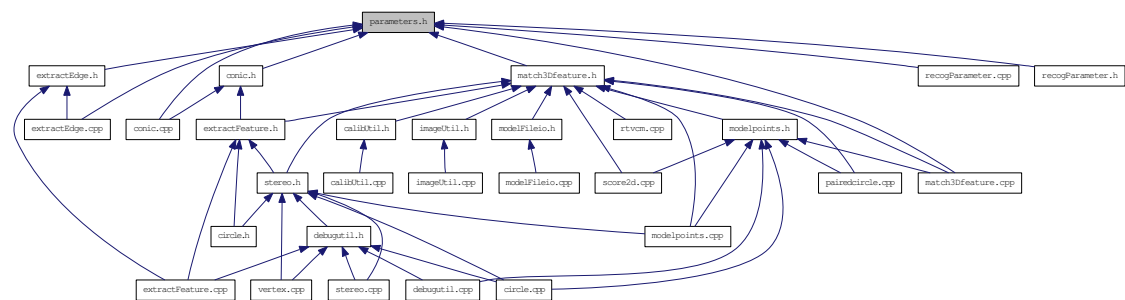
4.29 parameters.h

処理パラメータ設定関連関数 `#include "common.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

4.29.1 説明

処理パラメータ設定関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #

4.29.2 型定義

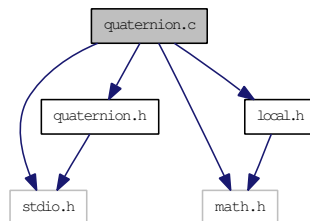
4.29.2.1 typedef struct [ParametersStereo](#) [ParametersStereo](#)

ステレオ対応処理用パラメータ

4.30 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)

4.30.1 関数

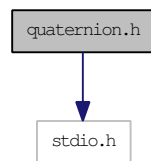
4.30.1.1 void [quat_conj](#) ([quaternion_t](#) result, const [quaternion_t](#) q)

- 4.30.1.2** void quat_copy (quaternion_t *dst*, const quaternion_t *src*)
- 4.30.1.3** void quat_fprintf (FILE **fp*, const char **fmt*, const char *sep*, const quaternion_t *q*)
- 4.30.1.4** void quat_irot (double *result*[3], const quaternion_t *q*, const double *x*[3])
- 4.30.1.5** void quat_make_from_rvec (quaternion_t *result*, const double *theta*, const double *x*, const double *y*, const double *z*)
- 4.30.1.6** void quat_mult (quaternion_t *result*, const quaternion_t *q1*, const quaternion_t *q2*)
- 4.30.1.7** double quat_norm2 (const quaternion_t *q*)
- 4.30.1.8** double quat_normalize (quaternion_t *q*)
- 4.30.1.9** void quat_q_from_R (quaternion_t *q*, const double **R*, const int *ldim*)
- 4.30.1.10** void quat_R_from_q (double **R*, const int *ldim*, const quaternion_t *q*)
- 4.30.1.11** void quat_rot (double *result*[3], const quaternion_t *q*, const double *x*[3])

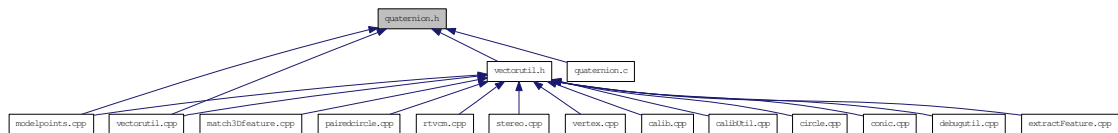
4.31 quaternion.h

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

quaternion.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 QUAT_EPS 1.0e-15
- #define QUAT_INIT_ZERO {0.0, 0.0, 0.0, 0.0}
- #define QUAT_INIT_ONE {0.0, 0.0, 0.0, 1.0}
- #define quat_re(q) (q)[3]
- #define quat_im(q, i) (q)[(i)]
- #define quat_printf(fmt, sep, q) quat_fprintf(stdout, (fmt), (sep), (q))
- #define quat_fprint(fp, q) quat_fprintf((fp), "% 10.3g", ' ', (q))
- #define quat_print(q) quat_fprintf(stdout, "% 10.3g", ' ', (q))

型定義

- typedef double quaternion_t [4]

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_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)

4.31.1 マクロ定義

4.31.1.1 `#define QUAT_EPS 1.0e-15`

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

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

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

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

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

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

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

4.31.2 型定義

4.31.2.1 `typedef double quaternion_t[4]`

4.31.3 関数

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

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

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

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

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

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

4.31.3.7 `double quat_norm2 (const quaternion_t q)`

4.31.3.8 `double quat_normalize (quaternion_t q)`

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

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

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

4.32 recogImage.cpp

画像入出力関数#include <stdlib.h>

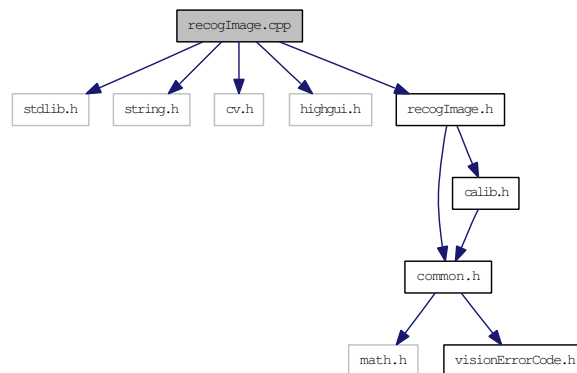
```
#include <string.h>
```

```
#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
- 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, RecogImage *dst, CameraParam *cp)

4.32.1 説明

画像入出力関数

日付:

\$Date:: 2011-08-02 09:15:13 +0900 #

4.32.2 関数

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

画像メモリの確保と初期化

4.32.2.2 **void** **destructImage** (**RecogImage** * *image*)

画像メモリの解放

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

RGB 画像から Grey 画像への変換.

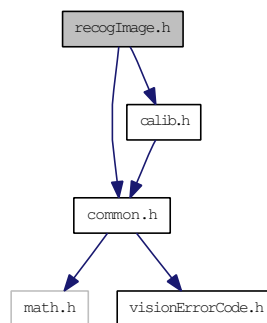
4.32.2.4 **void** **undistortImage** (**const** **RecogImage** * *src*, **RecogImage** * *dst*, **CameraParam** * *cp*)

4.33 recogImage.h

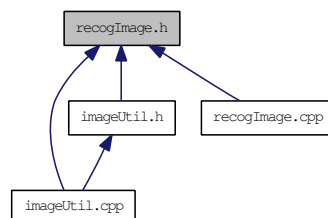
画像入出力関数 `#include "common.h"`

`#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

- 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, `RecogImage` *dst, `CameraParam` *cp)

4.33.1 説明

画像入出力関数

日付:

\$Date:: 2011-06-30 18:09:57 +0900 #

4.33.2 型定義

4.33.2.1 typedef struct _recogImage RecogImage

4.33.3 関数

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

画像メモリの確保と初期化

4.33.3.2 void `destructImage` (`RecogImage` * *image*)

画像メモリの解放

4.33.3.3 void `rgb2grayImage` (`RecogImage` * *target*, `RecogImage` * *source*)

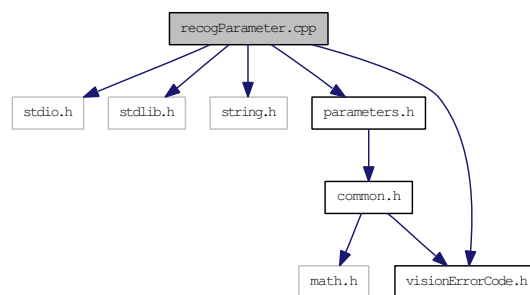
RGB 画像から Grey 画像への変換.

4.33.3.4 void `undistortImage` (const `RecogImage` * *src*, `RecogImage` * *dst*, `CameraParam` * *cp*)

4.34 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, eMaxErrorOfConicFit, eOverlapRatioLine,
 - eOverlapRatioCircle,
 - eMinLengthLine2D, eHDMMax, eDepN, eDepF,
 - eAMin, eAMax, eLMin, eLMax,
 - eStereoError, eMatchEdge, 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)

4.34.1 列挙型

4.34.1.1 enum paramKey

列挙型の値:

eStereoPair
eOutputCandNum
eEdgeDetectFunction
eEdgeStrength
eMaxErrorOfLineFit
eMaxErrorOfConicFit
eOverlapRatioLine
eOverlapRatioCircle
eMinLengthLine2D
eHDMax
eDepN
eDepF
eAMin
eAMax
eLMin
eLMax
eStereoError
eMatchEdge
eParamSentinel

4.34.2 関数

4.34.2.1 int loadDebugParameter (int *text*, int *image*, int *display*, Parameters & *param*)

4.34.2.2 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。

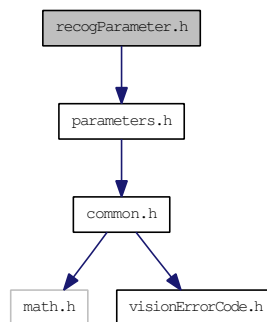
4.34.2.3 void setDefaultRecogParameter (Parameters & *param*)

[Parameters](#) 構造体にデフォルト値をセットする。 .

4.35 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)

4.35.1 説明

認識パラメータ設定関連

4.35.2 関数

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

4.35.2.2 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。

4.35.2.3 void setDefaultRecogParameter (Parameters & *param*)

Parameters 構造体にデフォルト値をセットする。.

4.36 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` }

4.36.1 説明

認識結果の定義

4.36.2 マクロ定義

4.36.2.1 #define `RecogResultElementNum` 20

4.36.3 列挙型

4.36.3.1 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

4.37 rtvcm.cpp

モデル入出力関連関数 `#include <iostream>`

`#include <fstream>`

`#include <string>`

`#include <math.h>`

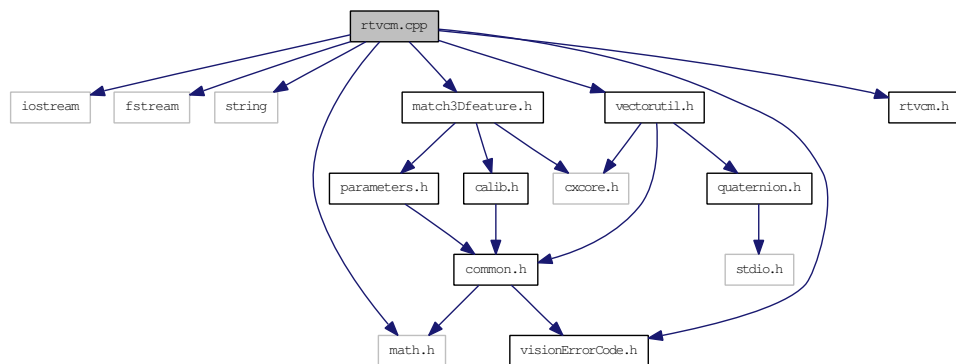
`#include "match3Dfeature.h"`

`#include "vectorutil.h"`

`#include "rtvcm.h"`

`#include "visionErrorCode.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
- 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

4.37.1 説明

モデル入出力関連関数

日付:

\$Date:: 2011-07-22 17:40:47 +0900 #

4.37.2 関数

4.37.2.1 `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

4.37.2.2 `void freeRTVCM (RTVCM & rtvcm)`

モデルデータのメモリ解放

4.37.2.3 `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

4.37.2.4 `void reverseCircle (Circle src, Circle & dst)`

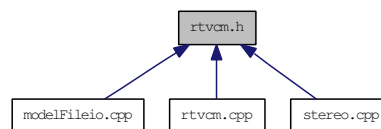
3 次元円データの裏データ作成

4.37.2.5 `void reverseVertex (Vertex src, Vertex & dst)`

3 次元頂点データの裏データ作成

4.38 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

4.38.1 説明

モデル入出力関連関数

日付:

\$Date:: 2011-07-01 14:38:35 +0900 #

4.38.2 型定義

4.38.2.1 typedef struct RTVertexCircleModel RTVCM

モデルデータ構造体

4.38.2.2 typedef int RTVCM_Label

4.38.3 関数

4.38.3.1 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

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

モデルデータのメモリ解放

4.38.3.3 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

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

3次元円データの裏データ作成

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

3次元頂点データの裏データ作成

4.39 score2d.cpp

2次元評価関連関数 `#include <stdio.h>`

`#include <assert.h>`

`#include <stdlib.h>`

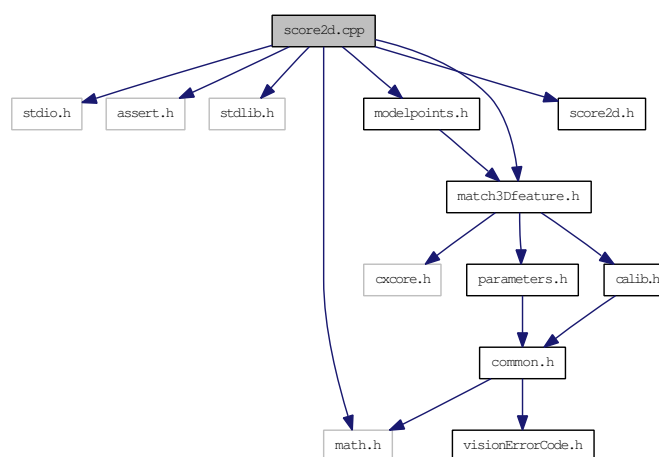
`#include <math.h>`

`#include "modelpoints.h"`

`#include "match3Dfeature.h"`

`#include "score2d.h"`

score2d.cpp のインクルード依存関係図



マクロ定義

- `#define COL_TABLE {1, 1, 0, -1, -1, -1, 0, 1}`
- `#define ROW_TABLE {0, -1, -1, -1, 0, 1, 1, 1}`
- `#define BOTTOMOFFSET 1`
- `#define EDGE_SEARCH_MAX 36`
- `#define EDGE_SEARCH_SIZE EDGE_SEARCH_MAX`
- `#define EDGE_SEARCH_2SIZE (((EDGE_SEARCH_SIZE)*2)+1+BOTTOMOFFSET)`

関数

- `int isValidPixelPosition (int col, int row, Features3D *finfo)`

- int `compareResultScore` (const void *c1, const void *c2)
- int `tracePoint` (`Features3D` *finfo, `Trace` *data, int p_search, int p_edge, int p_camera)
- void `getResultScore` (`MatchResult` *results, int numOfResults, `Features3D` *model, `StereoPairing` &pairing, double weight)

4.39.1 説明

2次元評価関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 #

4.39.2 マクロ定義

4.39.2.1 `#define BOTTOMOFFSET 1`

4.39.2.2 `#define COL_TABLE {1, 1, 0, -1, -1, -1, 0, 1}`

4.39.2.3 `#define EDGE_SEARCH_2SIZE (((EDGE_SEARCH_SIZE)*2)+1+BOTTOMOFFSET)`

4.39.2.4 `#define EDGE_SEARCH_MAX 36`

4.39.2.5 `#define EDGE_SEARCH_SIZE EDGE_SEARCH_MAX`

4.39.2.6 `#define ROW_TABLE {0, -1, -1, -1, 0, 1, 1, 1}`

4.39.3 関数

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

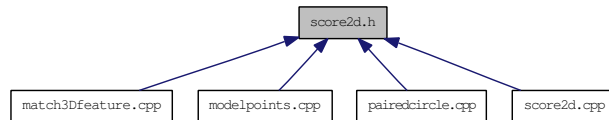
4.39.3.2 `void getResultScore (MatchResult * results, int numOfResults,
Features3D * model, StereoPairing & pairing, double weight)`

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

4.39.3.4 `int tracePoint (Features3D * finfo, Trace * data, int p_search, int
p_edge, int p_camera)`

4.40 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 [tracePoint](#) ([Features3D](#) *finfo, [Trace](#) *data, int p_search, int p_edge, int p_camera)
- int [compareResultScore](#) (const void *c1, const void *c2)
- void [getResultScore](#) ([MatchResult](#) *results, int numOfResults, [Features3D](#) *model, [StereoPairing](#) &pairing, double weight)

4.40.1 説明

2次元評価関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.40.2 関数

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

4.40.2.2 void [getResultScore](#) ([MatchResult](#) * *results*, int *numOfResults*, [Features3D](#) * *model*, [StereoPairing](#) & *pairing*, double *weight*)

4.40.2.3 int [tracePoint](#) ([Features3D](#) * *finfo*, [Trace](#) * *data*, int *p_search*, int *p_edge*, int *p_camera*)

4.41 stereo.cpp

ステレオ処理関連関数 `#include "stereo.h"`

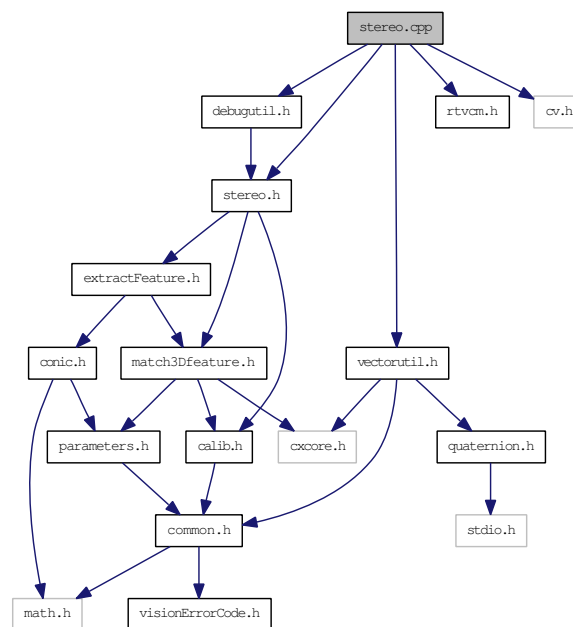
`#include "vectorutil.h"`

`#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

- void `calculateSightVector` (double *SightVector, `Data_2D` icPos, `CameraParam` *cameraParam)
歪み補正点座標 (X' , Y') より視線ベクトルを計算する
- double `calculateLR2XYZ` (double position3D[3], `Data_2D` posL, `Data_2D` posR, `CameraParam` *camParamL, `CameraParam` *camParamR)
- double `calculateLR2XYZ2` (double position3D[3], `Data_2D` posL, `Data_2D` posR, `CameraParam` *camParamL, `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

- void `freeStereoData` (`StereoData` *stereo)
ステレオ対応データのメモリ解放
- `StereoData StereoCorrespondence` (`StereoPairing` pairing, `CalibParam` calib, `Features2D` *left, `Features2D` *right, `Parameters` parameters)
- bool `setFeature3D` (`StereoData` &stereo, `Features3D` &feature)
ステレオ処理結果を3次元特徴構造体へセットする
- bool `setFeature3D_TBLOR` (`StereoData` &stereoLR, `StereoData` &stereoLV, `StereoData` &stereoRV, `Features3D` &feature)
ステレオ処理結果を3次元特徴構造体へセットする：3眼OR処理
- bool `setFeature3D_TBLAND` (`StereoData` &stereoLR, `StereoData` &stereoLV, `Features3D` &feature)
ステレオ処理結果を3次元特徴構造体へセットする：3眼AND処理

4.41.1 説明

ステレオ処理関連関数

日付:

\$Date:: 2011-08-02 11:42:18 +0900 # \$

4.41.2 関数

4.41.2.1 `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

4.41.2.2 `double calculateLR2XYZ2` (`double position3D`[3], `Data_2D posL`, `Data_2D posR`, `CameraParam` * `camParamL`, `CameraParam` * `camParamR`)

4.41.2.3 void calculateSightVector (double * *SightVector*, Data_2D *icPos*, CameraParam * *cameraParam*)

歪み補正点座標 (X', Y') より視線ベクトルを計算する

4.41.2.4 void freeStereoData (StereoData * *stereo*)

ステレオ対応データのメモリ解放

4.41.2.5 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

4.41.2.6 bool setFeature3D (StereoData & *stereo*, Features3D & *feature*)

ステレオ処理結果を3次元特徴構造体へセットする

4.41.2.7 bool setFeature3D_TBLAND (StereoData & *stereoLR*, StereoData & *stereoLV*, Features3D & *feature*)

ステレオ処理結果を3次元特徴構造体へセットする：3眼AND処理

4.41.2.8 bool setFeature3D_TBLOR (StereoData & *stereoLR*, StereoData & *stereoLV*, StereoData & *stereoRV*, Features3D & *feature*)

ステレオ処理結果を3次元特徴構造体へセットする：3眼OR処理

4.41.2.9 StereoData StereoCorrespondence (StereoPairing *pairing*, CalibParam *calib*, Features2D * *left*, Features2D * *right*, Parameters *parameters*)

ステレオ対応データの作成 戻り値：ステレオ対応データ

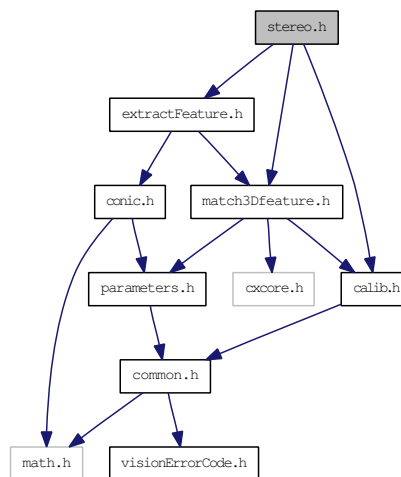
4.42 stereo.h

ステレオ処理関連関数 `#include "calib.h"`

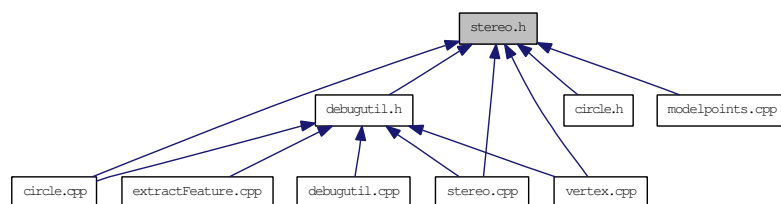
`#include "extractFeature.h"`

`#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
- struct [StereoConic](#)
二次曲線ステレオ対応データ
- struct [StereoData](#)
ステレオ対応データ

関数

- void [calculateSightVector](#) (double *SightVector, [Data_2D](#) icPos, [CameraParam](#) *cameraParam)
歪み補正点座標 (X' , Y') より視線ベクトルを計算する
- double [calculateLR2XYZ](#) (double position3D[3], [Data_2D](#) posL, [Data_2D](#) posR, [CameraParam](#) *camParamL, [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
- void [freeStereoData](#) ([StereoData](#) *stereo)
ステレオ対応データのメモリ解放
- [StereoData](#) [StereoCorrespondence](#) ([StereoPairing](#) pairing, [CalibParam](#) calib, [Features2D](#) *left, [Features2D](#) *right, [Parameters](#) parameters)
- bool [setFeature3D](#) ([StereoData](#) &stereo, [Features3D](#) &feature)
ステレオ処理結果を3次元特徴構造体へセットする
- bool [setFeature3D_TBLOR](#) ([StereoData](#) &stereoLR, [StereoData](#) &stereoLV, [StereoData](#) &stereoRV, [Features3D](#) &feature)
ステレオ処理結果を3次元特徴構造体へセットする：3眼OR処理
- bool [setFeature3D_TBLAND](#) ([StereoData](#) &stereoLR, [StereoData](#) &stereoLV, [Features3D](#) &feature)
ステレオ処理結果を3次元特徴構造体へセットする：3眼AND処理

4.42.1 説明

ステレオ処理関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.42.2 関数

4.42.2.1 `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

4.42.2.2 `void calculateSightVector (double * SightVector, Data_2D icPos, CameraParam * cameraParam)`

歪み補正点座標 (X', Y') より視線ベクトルを計算する

4.42.2.3 `void freeStereoData (StereoData * stereo)`

ステレオ対応データのメモリ解放

4.42.2.4 `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

4.42.2.5 `bool setFeature3D (StereoData & stereo, Features3D & feature)`

ステレオ処理結果を 3 次元特徴構造体へセットする

4.42.2.6 `bool setFeature3D_TBLAND (StereoData & stereoLR, StereoData & stereoLV, Features3D & feature)`

ステレオ処理結果を 3 次元特徴構造体へセットする：3 眼 AND 処理

4.42.2.7 `bool setFeature3D_TBLOR (StereoData & stereoLR, StereoData & stereoLV, StereoData & stereoRV, Features3D & feature)`

ステレオ処理結果を 3 次元特徴構造体へセットする：3 眼 OR 処理

4.42.2.8 `StereoData StereoCorrespondence (StereoPairing pairing, CalibParam calib, Features2D * left, Features2D * right, Parameters parameters)`

ステレオ対応データの作成 戻り値：ステレオ対応データ

4.43 vectorutil.cpp

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

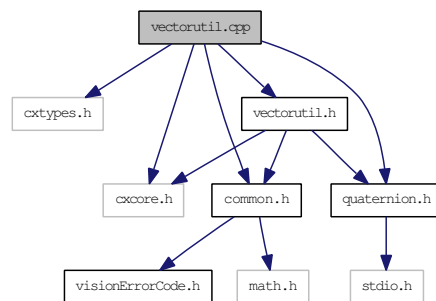
```
#include <cxcore.h>
```

```
#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])
- `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])

4.43.1 説明

ベクトル処理、行列処理 ユーティリティ関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.43.2 関数

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

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

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

4.43.2.4 double `getAngle2D` (double *vec1*[2], double *vec2*[2])

4.43.2.5 void `getCrossProductV3` (`V3` in1, `V3` in2, `V3` out)

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

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

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

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

4.43.2.10 double getNormV2 (V2 *in*)

4.43.2.11 double getNormV3 (V3 *in*)

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

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

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

4.43.2.15 int isZero (double *value*)

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

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

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

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

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

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

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

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

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

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

4.43.2.26 void zeroV3 (*V3 out*)

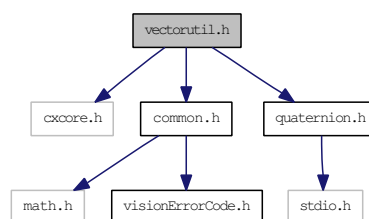
4.44 vectorutil.h

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

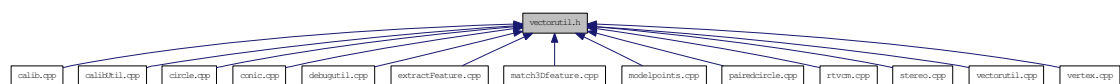
```
#include "common.h"
```

```
#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])
- 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])

4.44.1 説明

ベクトル処理、行列処理 ユーティリティ関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.44.2 型定義

4.44.2.1 `typedef double M33[3][3]`

3x3 matrix

4.44.2.2 `typedef double V2[2]`

2 dimensional vector

4.44.2.3 `typedef double V3[3]`

3 dimensional vector

4.44.3 関数

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

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

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

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

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

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

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

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

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

4.44.3.10 **double** getNormV2 (*V2 in*)

4.44.3.11 **double** getNormV3 (*V3 in*)

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

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

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

4.44.3.15 **int** isZero (*double value*)

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

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

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

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

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

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

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

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

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

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

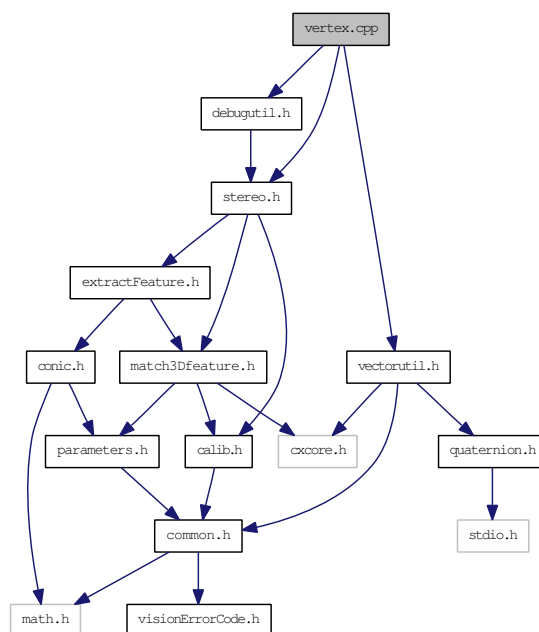
4.45 vertex.cpp

3次元頂点特徴生成関連関数 `#include "stereo.h"`

`#include "vectorutil.h"`

`#include "debugutil.h"`

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 HyperbolaToVertex` (`StereoPairing` pairing, `CalibParam` calib, `StereoData` &stereo, `Features2D` *left, `Features2D` *right, unsigned char *edgeL, unsigned char *edgeR, `Parameters` parameters)

二次元双曲線データから三次元頂点データを生成

4.45.1 説明

3次元頂点特徴生成関連関数

日付:

\$Date:: 2011-08-08 15:03:50 +0900 #

4.45.2 関数

4.45.2.1 void HyperbolaToVertex (StereoPairing *pairing*, CalibParam *calib*, StereoData & *stereo*, Features2D * *left*, Features2D * *right*, unsigned char * *edgeL*, unsigned char * *edgeR*, Parameters *parameters*)

二次元双曲線データから三次元頂点データを生成

4.45.2.2 int isValidPixelPosition (double *col*, double *row*, const Parameters & *parameters*) [inline]

4.46 vertex.h

3次元頂点特徴生成関連関数

関数

- void [HyperbolaToVertex](#) ([StereoPairing](#) pairing, [CalibParam](#) calib, [StereoData](#) &stereo, [Features2D](#) *left, [Features2D](#) *right, unsigned char *edgeL, unsigned char *edgeR, [Parameters](#) parameters)
二次元双曲線データから三次元頂点データを生成

4.46.1 説明

3次元頂点特徴生成関連関数

日付:

\$Date:: 2011-06-23 14:52:42 +0900 # \$

4.46.2 関数

4.46.2.1 void [HyperbolaToVertex](#) ([StereoPairing](#) *pairing*, [CalibParam](#) *calib*, [StereoData](#) & *stereo*, [Features2D](#) * *left*, [Features2D](#) * *right*, unsigned char * *edgeL*, unsigned char * *edgeR*, [Parameters](#) *parameters*)

二次元双曲線データから三次元頂点データを生成

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