

RAW-ROAM

Generated by Doxygen 1.9.3

1 RAW-ROAM: Really Adverse Weather-Radar Odometry and Mapping (Python reimplementaion of RadarSLAM)	1
1.1 Results (Odometry)	1
1.2 Running Code	1
1.2.1 Radar Sequences	1
1.3 Documentation	1
1.4 Relevant Papers	1
2 Namespace Index	3
2.1 Packages	3
3 Hierarchical Index	5
3.1 Class Hierarchy	5
4 Class Index	7
4.1 Class List	7
5 Namespace Documentation	9
5.1 ANMS Namespace Reference	9
5.1.1 Function Documentation	9
5.1.1.1 ssc()	9
5.2 Coord Namespace Reference	9
5.3 FMT Namespace Reference	10
5.3.1 Function Documentation	10
5.3.1.1 getRotationUsingFMT()	10
5.3.1.2 getTranslationUsingPhaseCorrelation()	11
5.3.1.3 plotCartPolar()	11
5.3.1.4 plotCartPolarWithRotation()	11
5.3.1.5 rotateImg()	11
5.3.2 Variable Documentation	12
5.3.2.1 currImgCart	12
5.3.2.2 currImgPolar	12
5.3.2.3 dataPath	12
5.3.2.4 endSeqInd	12
5.3.2.5 exist_ok	12
5.3.2.6 FMT_DOWNSAMPLE_FACTOR	12
5.3.2.7 FMT_RANGE_CLIP_M	13
5.3.2.8 imgPathArr	13
5.3.2.9 imgSavePath	13
5.3.2.10 imgSavePathInd	13
5.3.2.11 prevImgCart	13
5.3.2.12 prevImgPolar	13
5.3.2.13 REMOVE_OLD_RESULTS	13
5.3.2.14 response	14

5.3.2.15 rotRad	14
5.3.2.16 scale	14
5.3.2.17 sequenceName	14
5.3.2.18 sequenceSize	14
5.3.2.19 startSeqInd	14
5.3.2.20 stepSize	14
5.3.2.21 timestampPath	14
5.4 genFakeData Namespace Reference	15
5.4.1 Function Documentation	15
5.4.1.1 addNoise()	15
5.4.1.2 convertPolarPointsToCartesian()	15
5.4.1.3 createOutliers()	16
5.4.1.4 distort()	16
5.4.1.5 generateFakeCorrespondences()	16
5.4.1.6 generateFakeCorrespondencesPolar()	17
5.4.1.7 generateFakeFeatures()	17
5.4.1.8 generateFakeFeaturesPolar()	17
5.4.1.9 generateTranslationVector()	17
5.4.1.10 plotFakeFeatures()	17
5.4.1.11 transformCoords()	18
5.5 getFeatures Namespace Reference	18
5.5.1 Function Documentation	19
5.5.1.1 adaptiveNMS()	19
5.5.1.2 appendNewFeatures()	19
5.5.1.3 calculateFeatureLossThreshold()	19
5.5.1.4 getBlobsFromCart()	19
5.5.1.5 getFeatures()	20
5.5.2 Variable Documentation	20
5.5.2.1 blobIndices	20
5.5.2.2 blobSigma	20
5.5.2.3 blobX	20
5.5.2.4 blobY	20
5.5.2.5 color	21
5.5.2.6 coord	21
5.5.2.7 dataPath	21
5.5.2.8 datasetName	21
5.5.2.9 DEFAULT_FEATURE_PARAMS	21
5.5.2.10 end	21
5.5.2.11 exist_ok	21
5.5.2.12 flush	22
5.5.2.13 imgCart	22
5.5.2.14 imgCartBGR	22

5.5.2.15 imgPolar	22
5.5.2.16 N_FEATURES_BEFORE_RETRACK	22
5.5.2.17 nImgs	22
5.5.2.18 nIndices	22
5.5.2.19 nIndicesANMS	22
5.5.2.20 PERCENT_FEATURE_LOSS_THRESHOLD	23
5.5.2.21 s_blobIndices	23
5.5.2.22 streamArr	23
5.5.2.23 timestampPath	23
5.5.2.24 toSaveImgPath	23
5.5.2.25 toSavePath	23
5.6 getPointCloud Namespace Reference	23
5.6.1 Function Documentation	24
5.6.1.1 getPointCloudPolarInd()	24
5.6.2 Variable Documentation	24
5.6.2.1 c	24
5.6.2.2 dataPath	24
5.6.2.3 datasetName	25
5.6.2.4 featureAzim	25
5.6.2.5 featureImgCart	25
5.6.2.6 featurePolarImage	25
5.6.2.7 featurePolarIndices	25
5.6.2.8 featureRange	25
5.6.2.9 imgCart	25
5.6.2.10 imgCartRGB	25
5.6.2.11 imgPolar	26
5.6.2.12 nImgs	26
5.6.2.13 streamArr	26
5.6.2.14 timestampPath	26
5.7 getTransformKLT Namespace Reference	26
5.7.1 Function Documentation	27
5.7.1.1 calculateTransform()	28
5.7.1.2 calculateTransformDth()	28
5.7.1.3 calculateTransformDxDth()	28
5.7.1.4 calculateTransformSVD()	28
5.7.1.5 estimateTransformUsingDelats()	29
5.7.1.6 getTrackedPointsKLT()	29
5.7.1.7 visualize_transform()	29
5.7.2 Variable Documentation	30
5.7.2.1 alpha	30
5.7.2.2 bad_new	30
5.7.2.3 bad_old	30

5.7.2.4 blobCoord	30
5.7.2.5 corrStatus	30
5.7.2.6 currImg	30
5.7.2.7 currTimestamp	30
5.7.2.8 dataPath	31
5.7.2.9 datasetName	31
5.7.2.10 dth	31
5.7.2.11 dx	31
5.7.2.12 ERR_THRESHOLD	31
5.7.2.13 est_deltas	31
5.7.2.14 estTraj	31
5.7.2.15 exist_ok	31
5.7.2.16 extraLabel	32
5.7.2.17 good_new	32
5.7.2.18 good_old	32
5.7.2.19 gt_deltas	32
5.7.2.20 gtTraj	32
5.7.2.21 gtTrajPath	32
5.7.2.22 h	32
5.7.2.23 imgPathArr	32
5.7.2.24 imgSavePath	33
5.7.2.25 initPose	33
5.7.2.26 initTimestamp	33
5.7.2.27 LK_PARAMS	33
5.7.2.28 N_FEATURES_BEFORE_RETRACK	33
5.7.2.29 nBadFeatures	33
5.7.2.30 nFeatures	34
5.7.2.31 nGoodFeatures	34
5.7.2.32 nImgs	34
5.7.2.33 PLOT_BAD_FEATURES	34
5.7.2.34 prev_good_old	34
5.7.2.35 prevImg	34
5.7.2.36 R	34
5.7.2.37 REMOVE_OLD_RESULTS	34
5.7.2.38 saveFeaturePath	35
5.7.2.39 savePath	35
5.7.2.40 show	35
5.7.2.41 start	35
5.7.2.42 startImgInd	35
5.7.2.43 timestamp	35
5.7.2.44 timestampPath	35
5.7.2.45 toSaveImgPath	36

5.7.2.46 toSaveTrajPath	36
5.7.2.47 trajSavePath	36
5.7.2.48 transformed_pts	36
5.8 Mapping Namespace Reference	36
5.8.1 Variable Documentation	36
5.8.1.1 RADAR_CART_CENTER	37
5.8.1.2 ROT_THRESHOLD	37
5.8.1.3 TRANS_THRESHOLD	37
5.8.1.4 TRANS_THRESHOLD_SQ	37
5.9 motionDistortion Namespace Reference	37
5.9.1 Variable Documentation	37
5.9.1.1 RADAR_SCAN_FREQUENCY	37
5.9.1.2 VERBOSE	38
5.10 outlierRejection Namespace Reference	38
5.10.1 Function Documentation	38
5.10.1.1 rejectOutliers()	39
5.10.2 Variable Documentation	39
5.10.2.1 alpha	39
5.10.2.2 DIST_THRESHOLD_M	39
5.10.2.3 DIST_THRESHOLD_PX	39
5.10.2.4 DISTSQ_THRESHOLD_PX	39
5.10.2.5 FORCE_OUTLIERS	40
5.10.2.6 max_translation_m	40
5.10.2.7 n_outliers	40
5.10.2.8 n_points	40
5.10.2.9 new_coord	40
5.10.2.10 new_coord_perfect	40
5.10.2.11 noiseToAdd	40
5.10.2.12 outlier_ind	40
5.10.2.13 prev_coord	41
5.10.2.14 pruned_new_coord	41
5.10.2.15 pruned_prev_coord	41
5.10.2.16 show	41
5.10.2.17 theta_deg	41
5.10.2.18 theta_max_deg	41
5.10.2.19 title_append	41
5.10.2.20 trans_vec	41
5.11 parseData Namespace Reference	42
5.11.1 Function Documentation	42
5.11.1.1 convertCartesianImageToPolar()	42
5.11.1.2 convertPolarImageToCartesian()	43
5.11.1.3 convertPolarImgToLogPolar()	43

5.11.1.4 drawCVPoint()	43
5.11.1.5 extractDataFromRadarImage()	43
5.11.1.6 getCartImageFromImgPaths()	44
5.11.1.7 getDataFromImgPathsByIndex()	44
5.11.1.8 getPolarImageFromImgPaths()	44
5.11.1.9 getRadarImgPaths()	45
5.11.1.10 getRadarStreamPolar()	45
5.11.2 Variable Documentation	45
5.11.2.1 c	45
5.11.2.2 dataPath	45
5.11.2.3 datasetName	45
5.11.2.4 DOWNSAMPLE_FACTOR	46
5.11.2.5 imgCart	46
5.11.2.6 imgPolar	46
5.11.2.7 MAX_RANGE_CLIP_DEFAULT	46
5.11.2.8 nImgs	46
5.11.2.9 RANGE_RESOLUTION_CART_M	46
5.11.2.10 RANGE_RESOLUTION_M	46
5.11.2.11 streamArr	46
5.11.2.12 timestampPath	47
5.12 PoseGraphLib Namespace Reference	47
5.13 RawROAMSystem Namespace Reference	47
5.13.1 Variable Documentation	47
5.13.1.1 datasetName	47
5.13.1.2 endSeqInd	47
5.13.1.3 imgSavePath	48
5.13.1.4 paramFlags	48
5.13.1.5 RADAR_CART_CENTER	48
5.13.1.6 REMOVE_OLD_RESULTS	48
5.13.1.7 startSeqInd	48
5.13.1.8 system	48
5.13.1.9 trajSavePath	49
5.13.1.10 wantToPlot	49
5.14 testMotionDistortion Namespace Reference	49
5.14.1 Variable Documentation	50
5.14.1.1 A	50
5.14.1.2 A_inv	50
5.14.1.3 alpha	50
5.14.1.4 clear	50
5.14.1.5 cov_p	50
5.14.1.6 cov_v	50
5.14.1.7 currentFrame	51

5.14.1.8 distorted	51
5.14.1.9 False	51
5.14.1.10 final_undistorted	51
5.14.1.11 frequency	51
5.14.1.12 groundTruth	51
5.14.1.13 h	51
5.14.1.14 h_fit	51
5.14.1.15 MDS	52
5.14.1.16 N	52
5.14.1.17 n_points	52
5.14.1.18 noiseToAdd	52
5.14.1.19 noisy	52
5.14.1.20 outlier_ind	52
5.14.1.21 outlier_rate	52
5.14.1.22 p_jt	52
5.14.1.23 p_w	53
5.14.1.24 params	53
5.14.1.25 period	53
5.14.1.26 plotDisplace	53
5.14.1.27 R_fit	53
5.14.1.28 show	53
5.14.1.29 solution	53
5.14.1.30 srcCoord2	53
5.14.1.31 srcCoord3	54
5.14.1.32 T_wj	54
5.14.1.33 T_wj0	54
5.14.1.34 theta_deg	54
5.14.1.35 theta_fit	54
5.14.1.36 title_append	54
5.14.1.37 transform	54
5.14.1.38 undistorted	55
5.14.1.39 useOld	55
5.14.1.40 v_j0	55
5.14.1.41 velocity	55
5.15 testTransform Namespace Reference	55
5.15.1 Variable Documentation	56
5.15.1.1 A	56
5.15.1.2 A_inv	56
5.15.1.3 alpha	56
5.15.1.4 clear	56
5.15.1.5 False	56
5.15.1.6 h	56

5.15.1.7 h_fit	57
5.15.1.8 N	57
5.15.1.9 n_points	57
5.15.1.10 noiseToAdd	57
5.15.1.11 noisy	57
5.15.1.12 outlier_ind	57
5.15.1.13 outlier_rate	57
5.15.1.14 R_fit	57
5.15.1.15 show	58
5.15.1.16 srcCoord	58
5.15.1.17 srcCoord2	58
5.15.1.18 targetCoord	58
5.15.1.19 theta_deg	58
5.15.1.20 theta_fit	58
5.15.1.21 title_append	58
5.15.1.22 useOld	58
5.16 Tracker Namespace Reference	59
5.17 trajectoryPlotting Namespace Reference	59
5.17.1 Function Documentation	59
5.17.1.1 computePosesRMSE()	59
5.17.1.2 getGroundTruthTrajectory()	60
5.17.1.3 getGroundTruthTrajectoryGPS()	60
5.17.1.4 plotGtAndEstTrajectory()	60
5.17.2 Variable Documentation	60
5.17.2.1 block	60
5.17.2.2 datasetName	61
5.17.2.3 estPoses	61
5.17.2.4 estTraj	61
5.17.2.5 gtPath	61
5.17.2.6 gtTraj	61
5.17.2.7 keyframe_timestamps	61
5.17.2.8 noise	61
5.17.2.9 timestampPath	62
5.18 utils Namespace Reference	62
5.18.1 Function Documentation	62
5.18.1.1 convertPoseToTransform()	62
5.18.1.2 convertRandHtoDeltas()	62
5.18.1.3 convertTransformToPose()	63
5.18.1.4 f_arr()	63
5.18.1.5 flatten()	63
5.18.1.6 getRotationMatrix()	63
5.18.1.7 homogenize()	63

5.18.1.8 invert_transform()	63
5.18.1.9 normalize_angles()	64
5.18.1.10 plt_full_extent()	64
5.18.1.11 plt_savefig_by_axis()	64
5.18.1.12 quiver()	64
5.18.1.13 radarImgPathToTimestamp()	64
5.18.1.14 tic()	65
5.18.1.15 toc()	65
6 Class Documentation	67
6.1 BundleAdjustment Class Reference	67
6.1.1 Constructor & Destructor Documentation	67
6.1.1.1 __init__()	67
6.1.2 Member Function Documentation	67
6.1.2.1 add_edge()	67
6.1.2.2 add_point()	68
6.1.2.3 add_pose()	68
6.1.2.4 get_point()	68
6.1.2.5 get_pose()	68
6.1.2.6 optimize()	68
6.2 CartCoord Class Reference	69
6.2.1 Detailed Description	69
6.2.2 Constructor & Destructor Documentation	69
6.2.2.1 __init__()	69
6.2.3 Member Function Documentation	69
6.2.3.1 __str__()	70
6.2.3.2 add()	70
6.2.3.3 addCoord()	70
6.2.3.4 asTuple()	70
6.2.3.5 getAngle()	70
6.2.3.6 getDistance()	70
6.2.3.7 getX()	71
6.2.3.8 getY()	71
6.2.3.9 scale()	71
6.2.3.10 scaleX()	71
6.2.3.11 scaleY()	71
6.2.3.12 sub()	71
6.2.4 Member Data Documentation	71
6.2.4.1 x	72
6.2.4.2 y	72
6.3 Keyframe Class Reference	72
6.3.1 Constructor & Destructor Documentation	72

6.3.1.1 <code>__init__()</code>	72
6.3.2 Member Function Documentation	73
6.3.2.1 <code>convertFeaturesLocalToGlobal()</code>	73
6.3.2.2 <code>copyFromOtherKeyframe()</code>	73
6.3.2.3 <code>getPrunedFeaturesGlobalPosition()</code>	73
6.3.2.4 <code>pruneFeaturePoints()</code>	73
6.3.2.5 <code>updateInfo()</code>	74
6.3.3 Member Data Documentation	74
6.3.3.1 <code>featurePointsLocal</code>	74
6.3.3.2 <code>featurePointsLocalUndistorted</code>	74
6.3.3.3 <code>pointCloud</code>	74
6.3.3.4 <code>pose</code>	74
6.3.3.5 <code>prunedFeaturePoints</code>	74
6.3.3.6 <code>prunedUndistortedLocals</code>	75
6.3.3.7 <code>radarPolarImg</code>	75
6.3.3.8 <code>velocity</code>	75
6.4 Map Class Reference	75
6.4.1 Constructor & Destructor Documentation	75
6.4.1.1 <code>__init__()</code>	75
6.4.2 Member Function Documentation	76
6.4.2.1 <code>addKeyframe()</code>	76
6.4.2.2 <code>isGoodKeyframe()</code>	76
6.4.2.3 <code>plot()</code>	76
6.4.2.4 <code>updateInternalTraj()</code>	76
6.4.3 Member Data Documentation	77
6.4.3.1 <code>estTraj</code>	77
6.4.3.2 <code>filePaths</code>	77
6.4.3.3 <code>imgPathArr</code>	77
6.4.3.4 <code>keyframes</code>	77
6.4.3.5 <code>mapPoints</code>	77
6.4.3.6 <code>sequenceName</code>	77
6.4.3.7 <code>sequenceSize</code>	77
6.5 MotionDistortionSolver Class Reference	78
6.5.1 Constructor & Destructor Documentation	78
6.5.1.1 <code>__init__()</code> [1/2]	78
6.5.1.2 <code>__init__()</code> [2/2]	79
6.5.2 Member Function Documentation	79
6.5.2.1 <code>compute_time_deltas()</code>	79
6.5.2.2 <code>error()</code>	79
6.5.2.3 <code>error_vector()</code>	79
6.5.2.4 <code>expected_observed_pts()</code>	80
6.5.2.5 <code>infer_velocity()</code>	80

6.5.2.6 jacobian()	80
6.5.2.7 jacobian_vector()	80
6.5.2.8 optimize()	80
6.5.2.9 optimize_library()	81
6.5.2.10 undistort()	81
6.5.2.11 update_problem()	81
6.5.3 Member Data Documentation	81
6.5.3.1 debug	81
6.5.3.2 dT	81
6.5.3.3 info_vector	82
6.5.3.4 p_jt	82
6.5.3.5 p_w	82
6.5.3.6 sigma_p	82
6.5.3.7 sigma_v	82
6.5.3.8 T_wj0	82
6.5.3.9 T_wj0_inv	82
6.5.3.10 T_wj_initial	82
6.5.3.11 total_scan_time	83
6.5.3.12 v_j_initial	83
6.6 PolarCoord Class Reference	83
6.6.1 Detailed Description	83
6.6.2 Constructor & Destructor Documentation	83
6.6.2.1 __init__()	83
6.6.3 Member Function Documentation	84
6.6.3.1 __str__()	84
6.6.3.2 asTuple()	84
6.6.3.3 getR()	84
6.6.3.4 getTheta()	84
6.6.3.5 scaleR()	84
6.6.3.6 toCart()	84
6.6.4 Member Data Documentation	85
6.6.4.1 r	85
6.6.4.2 theta	85
6.7 PoseGraphOptimization Class Reference	85
6.7.1 Constructor & Destructor Documentation	85
6.7.1.1 __init__()	85
6.7.2 Member Function Documentation	85
6.7.2.1 add_edge()	85
6.7.2.2 add_vertex()	86
6.7.2.3 get_pose()	86
6.7.2.4 optimize()	86
6.8 RawROAMSystem Class Reference	86

6.8.1 Constructor & Destructor Documentation	87
6.8.1.1 <code>__init__()</code>	87
6.8.2 Member Function Documentation	87
6.8.2.1 <code>plot()</code>	87
6.8.2.2 <code>plotTraj()</code>	88
6.8.2.3 <code>run()</code>	88
6.8.2.4 <code>updateTrajectory()</code>	88
6.8.2.5 <code>updateTrajectoryAbsolute()</code>	89
6.8.2.6 <code>updateTrajFromTracker()</code>	89
6.8.3 Member Data Documentation	89
6.8.3.1 <code>estTraj</code>	89
6.8.3.2 <code>fig</code>	89
6.8.3.3 <code>filePaths</code>	89
6.8.3.4 <code>gtTraj</code>	89
6.8.3.5 <code>hasGroundTruth</code>	90
6.8.3.6 <code>imgPathArr</code>	90
6.8.3.7 <code>map</code>	90
6.8.3.8 <code>paramFlags</code>	90
6.8.3.9 <code>sequenceName</code>	90
6.8.3.10 <code>sequenceSize</code>	90
6.8.3.11 <code>tracker</code>	90
6.9 Tracker Class Reference	91
6.9.1 Constructor & Destructor Documentation	91
6.9.1.1 <code>__init__()</code>	91
6.9.2 Member Function Documentation	91
6.9.2.1 <code>getTransform()</code>	91
6.9.2.2 <code>initTraj()</code>	92
6.9.2.3 <code>plot()</code>	92
6.9.2.4 <code>track()</code>	92
6.9.3 Member Data Documentation	92
6.9.3.1 <code>estTraj</code>	93
6.9.3.2 <code>filePaths</code>	93
6.9.3.3 <code>gtTraj</code>	93
6.9.3.4 <code>imgPathArr</code>	93
6.9.3.5 <code>paramFlags</code>	93
6.9.3.6 <code>sequenceName</code>	93
6.9.3.7 <code>sequenceSize</code>	93
6.10 Trajectory Class Reference	94
6.10.1 Constructor & Destructor Documentation	94
6.10.1.1 <code>__init__()</code>	94
6.10.2 Member Function Documentation	94
6.10.2.1 <code>appendAbsoluteTransform()</code>	94

6.10.2.2 <code>appendRelativeDeltas()</code>	95
6.10.2.3 <code>appendRelativeTransform()</code>	95
6.10.2.4 <code>getGroundTruthDeltasAtTime()</code>	95
6.10.2.5 <code>getPoseAtTimes()</code>	95
6.10.2.6 <code>plot()</code>	96
6.10.3 Member Data Documentation	96
6.10.3.1 <code>pose_transform</code>	96
6.10.3.2 <code>poses</code>	96
6.10.3.3 <code>timestamps</code>	96
Index	97

Chapter 1

RAW-ROAM: Really Adverse Weather-Radar Odometry and Mapping (Python reimplementa-tion of RadarSLAM)

A Python reimplementa-tion of odometry and mapping component of RadarSLAM by Hong et al. [1,2].

Final Paper: [RAW-ROAM: An Open-Source Implementation of Adverse Weather RadarSLAM](#)

1.1 Results (Odometry)

With Motion Compensation

Without Motion Compensation:

1.2 Running Code

Requires Python. Tested on Python ≥ 3.9 .

```
pip install -r requirements.txt
python3 RawROAMSystem.py <DATASET_NAME> [START_FRAME_IND [END_FRAME_IND]]
```

1.2.1 Radar Sequences

Radar sequences can be obtained from [Oxford Radar RobotCar Dataset](#) and should be placed in the `./data` folder. The folder organization listed as `full_seq_1` is an example of how the directory looks like, and is taken from the [10-11-46-21 sequence](#).

1.3 Documentation

See the [docs](#)

1.4 Relevant Papers

1. [RadarSLAM \(2021\)](#)
2. [RadarSLAM \(2020\)](#)
3. [PhaRaO](#)

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

ANMS	9
Coord	9
FMT	10
genFakeData	15
getFeatures	18
getPointCloud	23
getTransformKLT	26
Mapping	36
motionDistortion	37
outlierRejection	38
parseData	42
PoseGraphLib	47
RawROAMSystem	47
testMotionDistortion	49
testTransform	55
Tracker	59
trajectoryPlotting	59
utils	62

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

Keyframe	72
Map	75
MotionDistortionSolver	78
object	
CartCoord	69
PolarCoord	83
RawROAMSystem	86
SparseOptimizer	
BundleAdjustment	67
PoseGraphOptimization	85
Tracker	91
Trajectory	94

Chapter 4

Class Index

4.1 Class List

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

BundleAdjustment	67
CartCoord	69
Keyframe	72
Map	75
MotionDistortionSolver	78
PolarCoord	83
PoseGraphOptimization	85
RawROAMSystem	86
Tracker	91
Trajectory	94

Chapter 5

Namespace Documentation

5.1 ANMS Namespace Reference

Functions

- def [ssc](#) (keypoints, num_ret_points, tolerance, cols, rows)

5.1.1 Function Documentation

5.1.1.1 [ssc\(\)](#)

```
def ANMS.ssc (  
    keypoints,  
    num_ret_points,  
    tolerance,  
    cols,  
    rows )
```

5.2 Coord Namespace Reference

Classes

- class [CartCoord](#)
- class [PolarCoord](#)

5.3 FMT Namespace Reference

Functions

- tuple[tuple[float, float], float] [getTranslationUsingPhaseCorrelation](#) (np.ndarray srcImg, np.ndarray targetImg)
- tuple[float, float, float] [getRotationUsingFMT](#) (np.ndarray srcPolarImg, np.ndarray targetPolarImg, int downsampleFactor=[FMT_DOWNSAMPLE_FACTOR](#), maxRangeClipM=[FMT_RANGE_CLIP_M](#))
- def [rotatelmg](#) (image, angle_degrees)
- def [plotCartPolar](#) (prevImgPolar, currImgPolar, prevImgCart, currImgCart)
- def [plotCartPolarWithRotation](#) (prevImgCart, currImgCart, rotRad)

Variables

- int [FMT_DOWNSAMPLE_FACTOR](#) = 10
- float [FMT_RANGE_CLIP_M](#) = 87.5
- int [sequenceName](#) = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- [dataPath](#) = os.path.join("data", sequenceName, "radar")
- [timestampPath](#) = os.path.join("data", sequenceName, "radar.timestamps")
- int [startSeqInd](#) = int(sys.argv[2]) if len(sys.argv) > 2 else 0
- int [endSeqInd](#) = int(sys.argv[3]) if len(sys.argv) > 3 else -1
- [imgPathArr](#) = getRadarImgPaths([dataPath](#), [timestampPath](#))
- [sequenceSize](#) = len([imgPathArr](#))
- [prevImgPolar](#) = getPolarImageFromImgPaths([imgPathArr](#), [startSeqInd](#))
- [prevImgCart](#)
- [imgSavePath](#)
- [exist_ok](#)
- int [stepSize](#) = 1
- [currImgPolar](#) = getPolarImageFromImgPaths([imgPathArr](#), [seqInd](#))
- [currImgCart](#)
- [rotRad](#)
- [scale](#)
- [response](#)
- [imgSavePathInd](#) = os.path.join([imgSavePath](#), f"{seqInd:04d}_5.jpg")
- bool [REMOVE_OLD_RESULTS](#) = False

5.3.1 Function Documentation

5.3.1.1 getRotationUsingFMT()

```
tuple[float, float, float] FMT.getRotationUsingFMT (
    np.ndarray srcPolarImg,
    np.ndarray targetPolarImg,
    int downsampleFactor = FMT\_DOWNSAMPLE\_FACTOR,
    maxRangeClipM = FMT\_RANGE\_CLIP\_M )
```

```
@brief Get rotation using the Fourier-Mellin Transform
@note We attempt to downsample in the range direction.
      Since we are already in the polar domain, we just need to convert to a logpolar image
      apply phase correlation to get the rotation (which is a "\Delta Y" translation)

@param[in] srcPolarImg Source image in polar (not log-polar) form
@param[in] targetPolarImg Target image in polar (not log-polar) form
@param[in] How much to further downsample in

@return angleRad Angle in radians, where 'R(angleRad) @ src = target`
@return scaling Scaling factor
@return response Response value (indicates confidence)
```

5.3.1.2 getTranslationUsingPhaseCorrelation()

```
tuple[tuple[float, float], float] FMT.getTranslationUsingPhaseCorrelation (
    np.ndarray srcImg,
    np.ndarray targetImg )

@brief Using phase correlation, find the translation delta between 2 images
@param[in] srcImg Source image
@param[in] targetImg Target image

@return deltas Delta sub-pixel translation
@return response Response of phase correlation window (indicating confidence)
```

5.3.1.3 plotCartPolar()

```
def FMT.plotCartPolar (
    prevImgPolar,
    currImgPolar,
    prevImgCart,
    currImgCart )
```

5.3.1.4 plotCartPolarWithRotation()

```
def FMT.plotCartPolarWithRotation (
    prevImgCart,
    currImgCart,
    rotRad )
```

5.3.1.5 rotateImg()

```
def FMT.rotateImg (
    image,
    angle_degrees )
```

5.3.2 Variable Documentation

5.3.2.1 currImgCart

currImgCart

Initial value:

```
1 = convertPolarImageToCartesian(currImgPolar,  
2                               downsampleFactor=20)
```

5.3.2.2 currImgPolar

```
currImgPolar = getPolarImageFromImgPaths(imgPathArr, seqInd)
```

5.3.2.3 dataPath

```
dataPath = os.path.join("data", sequenceName, "radar")
```

5.3.2.4 endSeqInd

```
int endSeqInd = int(sys.argv[3]) if len(sys.argv) > 3 else -1
```

5.3.2.5 exist_ok

exist_ok

5.3.2.6 FMT_DOWNSAMPLE_FACTOR

```
int FMT_DOWNSAMPLE_FACTOR = 10
```

5.3.2.7 FMT_RANGE_CLIP_M

```
float FMT_RANGE_CLIP_M = 87.5
```

5.3.2.8 imgPathArr

```
imgPathArr = getRadarImgPaths(dataPath, timestampPath)
```

5.3.2.9 imgSavePath

```
imgSavePath
```

Initial value:

```
1 = os.path.join(".", "img", "fmt",  
2               sequenceName).strip(os.path.sep)
```

5.3.2.10 imgSavePathInd

```
imgSavePathInd = os.path.join(imgSavePath, f"{seqInd:04d}_5.jpg")
```

5.3.2.11 prevImgCart

```
prevImgCart
```

Initial value:

```
1 = convertPolarImageToCartesian(prevImgPolar,  
2                               downsampleFactor=20)
```

5.3.2.12 prevImgPolar

```
prevImgPolar = getPolarImageFromImgPaths(imgPathArr, startSeqInd)
```

5.3.2.13 REMOVE_OLD_RESULTS

```
bool REMOVE_OLD_RESULTS = False
```

5.3.2.14 response

```
response
```

5.3.2.15 rotRad

```
rotRad
```

5.3.2.16 scale

```
scale
```

5.3.2.17 sequenceName

```
int sequenceName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.3.2.18 sequenceSize

```
sequenceSize = len(imgPathArr)
```

5.3.2.19 startSeqInd

```
int startSeqInd = int(sys.argv[2]) if len(sys.argv) > 2 else 0
```

5.3.2.20 stepSize

```
int stepSize = 1
```

5.3.2.21 timestampPath

```
timestampPath = os.path.join("data", sequenceName, "radar.timestamps")
```

5.4 genFakeData Namespace Reference

Functions

- def [transformCoords](#) (srcCoord, A, h)
- def [plotFakeFeatures](#) (srcCoord, targetCoord, targetCoord2=None, title_append="", alpha=1, clear=False, show=False, plotDisplace=False)
- def [generateFakeCorrespondences](#) (srcCoord=None, n_points=100, theta_max_deg=20, max_translation↵_m=3)
- def [convertPolarPointsToCartesian](#) (points)
- def [generateFakeCorrespondencesPolar](#) (currentFrame=None, n_points=100, theta_max_deg=20, max_↵translation_m=3)
- def [distort](#) (coords, velocity, frequency, h)
- def [addNoise](#) (data, variance=2.5)
- def [createOutliers](#) (data, n_outliers, noiseToAdd=10)
- def [generateTranslationVector](#) (max_range_m=10)
- def [generateFakeFeatures](#) (n_points=100, max_range_m=10)
- def [generateFakeFeaturesPolar](#) (n_points=100, max_range_m=10)

5.4.1 Function Documentation

5.4.1.1 [addNoise\(\)](#)

```
def genFakeData.addNoise (
    data,
    variance = 2.5 )

@brief Add 0-mean Gaussian random noise to correspondence data
@param[in] data Data to add noise to
@param[in] variance Variance for Gaussian noise
```

5.4.1.2 [convertPolarPointsToCartesian\(\)](#)

```
def genFakeData.convertPolarPointsToCartesian (
    points )
```

5.4.1.3 createOutliers()

```
def genFakeData.createOutliers (
    data,
    n_outliers,
    noiseToAdd = 10 )

@brief Create outliers by adding a lot of noise to randomly chosen n_outliers
@param[in] data Data to create outliers in
@param[in] n_outliers Number of outliers forced into data
@param[in] noiseToAdd Amount of guaranteed base noise to add

@return noisy_data Noisy data with outliers
@return outlier_ind Indices of outliers
```

5.4.1.4 distort()

```
def genFakeData.distort (
    coords,
    velocity,
    frequency,
    h )
```

5.4.1.5 generateFakeCorrespondences()

```
def genFakeData.generateFakeCorrespondences (
    srcCoord = None,
    n_points = 100,
    theta_max_deg = 20,
    max_translation_m = 3 )

@brief Generate fake correspondences with transform, randomly generated from max range and degree
@param[in] srcCoord Source coordinate to transform from. If none, will randomly generate features
@param[in] n_points Number of points to generate, only applies if srcCoord = None
@param[in] theta_max_deg Maximum degree of rotation
@param[in] max_range_m Maximum range (for translation) in meters

@return srcCoord Generated or passed in srcCoord
@return targetCoord Corresponding targetCoord generated using (theta_deg, h)
@return theta_deg Theta component of transform
@return h Translation component of transform
```


5.4.1.6 generateFakeCorrespondencesPolar()

```
def genFakeData.generateFakeCorrespondencesPolar (
    currentFrame = None,
    n_points = 100,
    theta_max_deg = 20,
    max_translation_m = 3 )

@brief Generate fake correspondences with transform, randomly generated from max range and degree
@param[in] currentFrame Source coordinate to transform from. If none, will randomly generate features
@param[in] n_points Number of points to generate, only applies if currentFrame = None
@param[in] theta_max_deg Maximum degree of rotation
@param[in] max_range_m Maximum range (for translation) in meters

@return currentFrame Generated or passed in currentFrame
@return targetCoord Corresponding targetCoord generated using (theta_deg, h)
@return theta_deg Theta component of transform
@return h Translation component of transform
```

5.4.1.7 generateFakeFeatures()

```
def genFakeData.generateFakeFeatures (
    n_points = 100,
    max_range_m = 10 )
```

5.4.1.8 generateFakeFeaturesPolar()

```
def genFakeData.generateFakeFeaturesPolar (
    n_points = 100,
    max_range_m = 10 )
```

5.4.1.9 generateTranslationVector()

```
def genFakeData.generateTranslationVector (
    max_range_m = 10 )
```

5.4.1.10 plotFakeFeatures()

```
def genFakeData.plotFakeFeatures (
    srcCoord,
    targetCoord,
    targetCoord2 = None,
    title_append = "",
    alpha = 1,
    clear = False,
    show = False,
    plotDisplace = False )
```

5.4.1.11 transformCoords()

```
def genFakeData.transformCoords (
    srcCoord,
    A,
    h )

@brief Transform coordinates to get correspondence points given A, h transformation matrix
@param[in] srcCoord Source coordinates (K x 2)
@param[in] A Rotation matrix (2 x 2)
@param[in] h Translation matrix (2 x 1)

@return targetCoord = A @ srcCoord + h (K x 2)
```

5.5 getFeatures Namespace Reference

Functions

- np.ndarray [getBlobsFromCart](#) (np.ndarray cartImage, int min_sigma=1, int max_sigma=30, int num_←sigma=10, threshold=0.01, method="doh")
- def [calculateFeatureLossThreshold](#) (nInitialFeatures)
- def [adaptiveNMS](#) (img, blobs, ret_points=200, tolerance=0.1)
- def [getFeatures](#) (img, dict feature_params=DEFAULT_FEATURE_PARAMS)
- def [appendNewFeatures](#) (srcImg, oldFeaturesCoord)

Variables

- [DEFAULT_FEATURE_PARAMS](#)
- float [PERCENT_FEATURE_LOSS_THRESHOLD](#) = 0.75
- int [N_FEATURES_BEFORE_RETRACK](#) = 60
- int [datasetName](#) = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- [dataPath](#) = os.path.join("data", datasetName, "radar")
- [timestampPath](#) = os.path.join("data", datasetName, "radar.timestamps")
- [end](#)
- [flush](#)
- [streamArr](#) = getRadarStreamPolar([dataPath](#), [timestampPath](#))
- [nImgs](#) = streamArr.shape[2]
- [toSavePath](#) = os.path.join(".", "img", "blob", datasetName)
- [exist_ok](#)
- [imgPolar](#) = streamArr[:, :, imgNo]
- [imgCart](#) = convertPolarImageToCartesian([imgPolar](#))
- np.ndarray [blobIndices](#)
- def [s_blobIndices](#) = [adaptiveNMS](#)([imgCart](#), [blobIndices](#))
- int [imgCartBGR](#) = cv2.cvtColor([imgCart](#), cv2.COLOR_GRAY2BGR) * 255
- np.ndarray [nIndices](#) = [blobIndices](#).shape[0]
- def [nIndicesANMS](#) = [s_blobIndices](#).shape[0]
- [blobY](#)
- [blobX](#)
- [blobSigma](#)
- tuple [coord](#) = ([blobX](#), [blobY](#))
- tuple [color](#) = (0, 255, 0)
- [toSaveImgPath](#) = os.path.join([toSavePath](#), f"{imgNo:04d}.jpg")

5.5.1 Function Documentation

5.5.1.1 adaptiveNMS()

```
def getFeatures.adaptiveNMS (  
    img,  
    blobs,  
    ret_points = 200,  
    tolerance = 0.1 )
```

5.5.1.2 appendNewFeatures()

```
def getFeatures.appendNewFeatures (  
    srcImg,  
    oldFeaturesCoord )  
  
@brief Append new features obtained from srcImg onto oldFeaturesCoord array  
@see getFeatures()  
  
@param[in] srcImg Source image to obtain features on  
@param[in] oldFeaturesCoord (K x 2) array of [x, y] coordinate of features
```

5.5.1.3 calculateFeatureLossThreshold()

```
def getFeatures.calculateFeatureLossThreshold (  
    nInitialFeatures )
```

5.5.1.4 getBlobsFromCart()

```
np.ndarray getFeatures.getBlobsFromCart (  
    np.ndarray cartImage,  
    int min_sigma = 1,  
    int max_sigma = 30,  
    int num_sigma = 10,  
    threshold = 0.01,  
    method = "doh" )  
  
@brief Given a radar image, generate a list of (K x 3)  
    blob indices based on Determinant of Hessian  
@note Uses default params from skimage.features function  
  
@param[in] cartImage Cartesian radar image  
  
@return (K x 3) Np array of blob coordinates with each row [r, c, sigma] being coordinates and sigma of detect
```

5.5.1.5 getFeatures()

```
def getFeatures.getFeatures (
    img,
    dict feature_params = DEFAULT\_FEATURE\_PARAMS )

@brief Get features from image using Hessian blob detector
@param[in] img Image to detect features from
@param[in] feature_params Parameters for feature detection, @see DEFAULT\_FEATURE\_PARAMS

@return blobCoord (K x 2) array of [x, y] coordinates of center of blobs on the image
@return blobRadii (K x 1) array of radius of blobs
```

5.5.2 Variable Documentation

5.5.2.1 blobIndices

np.ndarray blobIndices

Initial value:

```
1 = getBlobsFromCart (imgCart,
2
3 min_sigma=0.01,
4 max_sigma=10,
5 num_sigma=3,
6 threshold=.0005,
method="doh")
```

5.5.2.2 blobSigma

blobSigma

5.5.2.3 blobX

blobX

5.5.2.4 blobY

blobY

5.5.2.5 color

```
tuple color = (0, 255, 0)
```

5.5.2.6 coord

```
tuple coord = (blobX, blobY)
```

5.5.2.7 dataPath

```
dataPath = os.path.join("data", datasetName, "radar")
```

5.5.2.8 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.5.2.9 DEFAULT_FEATURE_PARAMS

```
DEFAULT_FEATURE_PARAMS
```

Initial value:

```
1 = dict(  
2     min_sigma=0.01,  
3     max_sigma=10,  
4     num_sigma=3,  
5     threshold=.0005, # lower threshold for more features  
6     method="doh")
```

5.5.2.10 end

```
end
```

5.5.2.11 exist_ok

```
exist_ok
```

5.5.2.12 flush

```
flush
```

5.5.2.13 imgCart

```
imgCart = convertPolarImageToCartesian(imgPolar)
```

5.5.2.14 imgCartBGR

```
int imgCartBGR = cv2.cvtColor(imgCart, cv2.COLOR_GRAY2BGR) * 255
```

5.5.2.15 imgPolar

```
imgPolar = streamArr[:, :, imgNo]
```

5.5.2.16 N_FEATURES_BEFORE_RETRACK

```
int N_FEATURES_BEFORE_RETRACK = 60
```

5.5.2.17 nImgs

```
nImgs = streamArr.shape[2]
```

5.5.2.18 nIndices

```
np.ndarray nIndices = blobIndices.shape[0]
```

5.5.2.19 nIndicesANMS

```
def nIndicesANMS = s_blobIndices.shape[0]
```

5.5.2.20 PERCENT_FEATURE_LOSS_THRESHOLD

```
float PERCENT_FEATURE_LOSS_THRESHOLD = 0.75
```

5.5.2.21 s_blobIndices

```
def s_blobIndices = adaptiveNMS(imgCart, blobIndices)
```

5.5.2.22 streamArr

```
streamArr = getRadarStreamPolar(dataPath, timestampPath)
```

5.5.2.23 timestampPath

```
timestampPath = os.path.join("data", datasetName, "radar.timestamps")
```

5.5.2.24 toSaveImgPath

```
toSaveImgPath = os.path.join(toSavePath, f"{imgNo:04d}.jpg")
```

5.5.2.25 toSavePath

```
toSavePath = os.path.join(".", "img", "blob", datasetName)
```

5.6 getPointCloud Namespace Reference

Functions

- np.ndarray [getPointCloudPolarInd](#) (np.ndarray polarImage, float peakDistance=None, float peakProminence=None)

Variables

- `int datasetName = sys.argv[1]` if `len(sys.argv) > 1` else "tiny"
- `dataPath = os.path.join("data", datasetName, "radar")`
- `timestampPath = os.path.join("data", datasetName, "radar.timestamps")`
- `streamArr = getRadarStreamPolar(dataPath, timestampPath)`
- `nimgs = streamArr.shape[2]`
- `imgPolar = streamArr[:, :, i]`
- `np.ndarray featurePolarIndices = getPointCloudPolarInd(imgPolar)`
- `featurePolarImage = np.zeros_like(imgPolar)`
- `featureAzim`
- `featureRange`
- `imgCart = convertPolarImageToCartesian(imgPolar)`
- `imgCartRGB = cv2.cvtColor(imgCart, cv2.COLOR_GRAY2BGR)`
- `featureImgCart = convertPolarImageToCartesian(featurePolarImage)`
- `c = cv2.waitKey(100)`

5.6.1 Function Documentation

5.6.1.1 getPointCloudPolarInd()

```
np.ndarray getPointCloud.getPointCloudPolarInd (
    np.ndarray polarImage,
    float peakDistance = None,
    float peakProminence = None )

@brief Given a radar image, generate a list of polar indices
    based on peak detection with pruning

@param[in] peakDistance Minimum distance to be counted as a peak
@param[in] peakProminence Minimum prominence to be counted as a peak

@return (K x 2) Np array of polar coordinates with each row [thetaInd, rInd] being indices in the polar image
```

5.6.2 Variable Documentation

5.6.2.1 c

```
c = cv2.waitKey(100)
```

5.6.2.2 dataPath

```
dataPath = os.path.join("data", datasetName, "radar")
```


5.6.2.3 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.6.2.4 featureAzim

```
featureAzim
```

5.6.2.5 featureImgCart

```
featureImgCart = convertPolarImageToCartesian(featurePolarImage)
```

5.6.2.6 featurePolarImage

```
featurePolarImage = np.zeros_like(imgPolar)
```

5.6.2.7 featurePolarIndices

```
np.ndarray featurePolarIndices = getPointCloudPolarInd(imgPolar)
```

5.6.2.8 featureRange

```
featureRange
```

5.6.2.9 imgCart

```
imgCart = convertPolarImageToCartesian(imgPolar)
```

5.6.2.10 imgCartRGB

```
imgCartRGB = cv2.cvtColor(imgCart, cv2.COLOR_GRAY2BGR)
```

5.6.2.11 imgPolar

```
imgPolar = streamArr[:, :, i]
```

5.6.2.12 nImgs

```
nImgs = streamArr.shape[2]
```

5.6.2.13 streamArr

```
streamArr = getRadarStreamPolar(dataPath, timestampPath)
```

5.6.2.14 timestampPath

```
timestampPath = os.path.join("data", datasetName, "radar.timestamps")
```

5.7 getTransformKLT Namespace Reference

Functions

- None [visualize_transform](#) (np.ndarray [prevImg](#), np.ndarray [currImg](#), np.ndarray prevFeatureCoord, np.ndarray newFeatureCoord, float [alpha](#)=1, str [extraLabel](#)="", bool [show](#)=False)
- def [estimateTransformUsingDelats](#) (np.ndarray srcCoords, np.ndarray targetCoords)
- tuple[np.ndarray, np.ndarray] [calculateTransformSVD](#) (np.ndarray srcCoords, np.ndarray targetCoords)
- tuple[np.ndarray, np.ndarray] [calculateTransformDth](#) (np.ndarray srcCoords, np.ndarray targetCoords)
- tuple[np.ndarray, np.ndarray] [calculateTransformDxDth](#) (np.ndarray srcCoords, np.ndarray targetCoords)
- tuple[np.ndarray, np.ndarray] [calculateTransform](#) (np.ndarray srcCoords, np.ndarray targetCoords)
- tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray] [getTrackedPointsKLT](#) (np.ndarray srcImg, np.ndarray targetImg, np.ndarray blobCoordSrc)

Variables

- bool `PLOT_BAD_FEATURES` = False
- int `N_FEATURES_BEFORE_RETRACK` = 60
- `LK_PARAMS`
- int `ERR_THRESHOLD` = 10
- int `datasetName` = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- int `startImgInd` = int(sys.argv[2]) if len(sys.argv) > 2 else 0
- int `REMOVE_OLD_RESULTS` = bool(int(sys.argv[3])) if len(sys.argv) > 3 else False
- `dataPath` = os.path.join("data", datasetName, "radar")
- `timestampPath` = os.path.join("data", datasetName, "radar.timestamps")
- `imgPathArr` = getRadarImgPaths(`dataPath`, `timestampPath`)
- `nImgs` = len(`imgPathArr`)
- `imgSavePath`
- `trajSavePath`
- `saveFeaturePath`
- `exist_ok`
- `prevImg` = getCartImageFromImgPaths(`imgPathArr`, imgNo)
- `blobCoord` = data["blobCoord"]
- `gtTrajPath` = os.path.join("data", datasetName, "gt", "radar_odometry.csv")
- `gtTraj` = getGroundTruthTrajectory(`gtTrajPath`)
- `initTimestamp` = radarImgPathToTimestamp(`imgPathArr`[`startImgInd`])
- `initPose` = gtTraj.getPoseAtTimes(`initTimestamp`)
- `estTraj` = Trajectory([`initTimestamp`], [`initPose`])
- `good_old` = None
- `start` = tic()
- `currImg` = getCartImageFromImgPaths(`imgPathArr`, imgNo)
- `prev_good_old` = `good_old`
- `good_new`
- `bad_new`
- `bad_old`
- `corrStatus`
- `nGoodFeatures` = `good_new`.shape[0]
- `nBadFeatures` = `bad_new`.shape[0]
- `nFeatures` = `nGoodFeatures` + `nBadFeatures`
- `R`
- `h`
- tuple `transformed_pts` = (`R` @ `good_new`.T + `h`).T
- `show`
- `currTimestamp` = radarImgPathToTimestamp(`imgPathArr`[imgNo])
- `gt_deltas` = gtTraj.getGroundTruthDeltasAtTime(`currTimestamp`)
- `est_deltas` = convertRandHtoDeltas(`R`, `h`)
- `alpha`
- `extraLabel`
- `toSaveImgPath` = os.path.join(`imgSavePath`, f"{imgNo:04d}.jpg")
- `timestamp` = radarImgPathToTimestamp(`imgPathArr`[imgNo])
- `dx` = `est_deltas`[0]
- `dth` = `est_deltas`[2]
- `toSaveTrajPath` = os.path.join(`trajSavePath`, f"{imgNo:04d}.jpg")
- `savePath`

5.7.1 Function Documentation

5.7.1.1 calculateTransform()

```
tuple[np.ndarray, np.ndarray] getTransformKLT.calculateTransform (
    np.ndarray srcCoords,
    np.ndarray targetCoords )

@brief Calculate transform given 2 point correspondences.

TODO: Make this work with SVD
@see getCorrespondences.py
Inputs:
srcCoords      - (N, 2) array of source points
targetCoords   - (N, 2) array of target points
Outputs:
(R, h)         - (2 x 2), (2 x 1) arrays: rotation and translation. Apply
                  to old points srcCoords to get new points targetCoords, i.e.
                   $R * srcCoords + h = targetCoords$ 
```

5.7.1.2 calculateTransformDth()

```
tuple[np.ndarray, np.ndarray] getTransformKLT.calculateTransformDth (
    np.ndarray srcCoords,
    np.ndarray targetCoords )
```

5.7.1.3 calculateTransformDxDth()

```
tuple[np.ndarray, np.ndarray] getTransformKLT.calculateTransformDxDth (
    np.ndarray srcCoords,
    np.ndarray targetCoords )
```

5.7.1.4 calculateTransformSVD()

```
tuple[np.ndarray, np.ndarray] getTransformKLT.calculateTransformSVD (
    np.ndarray srcCoords,
    np.ndarray targetCoords )

@brief Calculate transform given 2 point correspondences using SVD.
Conventions:
 $Rx_1 + h = x_0$ 

Reference: https://www.sciencedirect.com/science/article/pii/002192909400116L
          http://nghiaho.com/?page\_id=671
@see getCorrespondences.py
Inputs:
srcCoords      - (N, 2) array of source points,  $x_0$ 
targetCoords   - (N, 2) array of target points,  $x_1$ 
Outputs:
(R, h)         - (2 x 2), (2 x 1) arrays: rotation and translation. Apply
                  to old points srcCoords to get new points targetCoords, i.e.
                   $R * srcCoords + h = targetCoords$ 
```

5.7.1.5 estimateTransformUsingDelats()

```
def getTransformKLT.estimateTransformUsingDelats (
    np.ndarray srcCoords,
    np.ndarray targetCoords )

@brief Estimate KLT [x, y] frame translation by taking average of deltaX and deltaYs from source
```

5.7.1.6 getTrackedPointsKLT()

```
tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray, np.ndarray] getTransformKLT.getTracked←
PointsKLT (
    np.ndarray srcImg,
    np.ndarray targetImg,
    np.ndarray blobCoordSrc )

@brief Get tracked points using the OpenCV KLT algorithm given the
    src and target img, and points from the src img to track

@param[in] srcImg      (M x N) Source image
@param[in] targetImg   (M x N) Target image
@param[in] blobIndicesSrc Indices source features (K x 2) (potentially (K x 3)) @note [x, y] format

@note Will append k more features if it finds that there are not enough features to track
@note Will also prune away features. Hence might have K' points instead

@return good_new      (K' x 2) New points considered as good correspondences
@return good_old      (K' x 2) Old points considered as good correspondences

@return bad_new       (K'' x 2) New points considered as bad correspondences
@return bad_old       (K'' x 2) Old points considered as bad correspondences

@return correspondenceStatus ((K + k) x 2) Status of correspondences (1 for valid, 0 for invalid/error)
```

5.7.1.7 visualize_transform()

```
None getTransformKLT.visualize_transform (
    np.ndarray prevImg,
    np.ndarray currImg,
    np.ndarray prevFeatureCoord,
    np.ndarray newFeatureCoord,
    float alpha = 1,
    str extraLabel = "",
    bool show = False )

@brief Visualize transform of good and bad points in 2 images

plt.subplot(1, 2, 1)
plt.imshow(prevImg)
plt.scatter(prevFeatureInd[:, 1],
            prevFeatureInd[:, 0],
            marker='.',
            color='red')
plt.title("Old Image")
plt.axis("off")

plt.subplot(1, 2, 2)
```

5.7.2 Variable Documentation

5.7.2.1 alpha

alpha

5.7.2.2 bad_new

bad_new

5.7.2.3 bad_old

bad_old

5.7.2.4 blobCoord

```
blobCoord = data["blobCoord"]
```

5.7.2.5 corrStatus

corrStatus

5.7.2.6 currImg

```
currImg = getCartImageFromImgPaths(imgPathArr, imgNo)
```

5.7.2.7 currTimestamp

```
currTimestamp = radarImgPathToTimestamp(imgPathArr[imgNo])
```

5.7.2.8 dataPath

```
dataPath = os.path.join("data", datasetName, "radar")
```

5.7.2.9 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.7.2.10 dth

```
dth = est_deltas[2]
```

5.7.2.11 dx

```
dx = est_deltas[0]
```

5.7.2.12 ERR_THRESHOLD

```
int ERR_THRESHOLD = 10
```

5.7.2.13 est_deltas

```
est_deltas = convertRandHtoDeltas(R, h)
```

5.7.2.14 estTraj

```
estTraj = Trajectory([initTimestamp], [initPose])
```

5.7.2.15 exist_ok

```
exist_ok
```

5.7.2.16 extraLabel

```
extraLabel
```

5.7.2.17 good_new

```
good_new
```

5.7.2.18 good_old

```
good_old = None
```

5.7.2.19 gt_deltas

```
gt_deltas = gtTraj.getGroundTruthDeltasAtTime(currTimestamp)
```

5.7.2.20 gtTraj

```
gtTraj = getGroundTruthTrajectory(gtTrajPath)
```

5.7.2.21 gtTrajPath

```
gtTrajPath = os.path.join("data", datasetName, "gt", "radar_odometry.csv")
```

5.7.2.22 h

```
h
```

5.7.2.23 imgPathArr

```
imgPathArr = getRadarImgPaths(dataPath, timestampPath)
```


5.7.2.24 imgSavePath

imgSavePath

Initial value:

```
1 = os.path.join(".", "img", "track_klt_thresholding",
2                      datasetName)
```

5.7.2.25 initPose

```
initPose = gtTraj.getPoseAtTimes(initTimestamp)
```

5.7.2.26 initTimestamp

```
initTimestamp = radarImgPathToTimestamp(imgPathArr[startImgInd])
```

5.7.2.27 LK_PARAMS

LK_PARAMS

Initial value:

```
1 = dict(
2     # level of pyramid search
3     maxLevel=3,
4     # termination criteria
5     criteria=(cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 0.03))
```

5.7.2.28 N_FEATURES_BEFORE_RETRACK

```
int N_FEATURES_BEFORE_RETRACK = 60
```

5.7.2.29 nBadFeatures

```
nBadFeatures = bad_new.shape[0]
```

5.7.2.30 nFeatures

```
nFeatures = nGoodFeatures + nBadFeatures
```

5.7.2.31 nGoodFeatures

```
nGoodFeatures = good_new.shape[0]
```

5.7.2.32 nImgs

```
nImgs = len(imgPathArr)
```

5.7.2.33 PLOT_BAD_FEATURES

```
bool PLOT_BAD_FEATURES = False
```

5.7.2.34 prev_good_old

```
prev_good_old = good_old
```

5.7.2.35 prevImg

```
prevImg = getCartImageFromImgPaths(imgPathArr, imgNo)
```

5.7.2.36 R

```
R
```

5.7.2.37 REMOVE_OLD_RESULTS

```
int REMOVE_OLD_RESULTS = bool(int(sys.argv[3])) if len(sys.argv) > 3 else False
```

5.7.2.38 saveFeaturePath

saveFeaturePath

Initial value:

```
1 = os.path.join(  
2     imgSavePath.strip(os.path.sep) + f"_{imgNo}.npz")
```

5.7.2.39 savePath

savePath

5.7.2.40 show

show

5.7.2.41 start

start = tic()

5.7.2.42 startImgInd

```
int startImgInd = int(sys.argv[2]) if len(sys.argv) > 2 else 0
```

5.7.2.43 timestamp

```
timestamp = radarImgPathToTimestamp(imgPathArr[imgNo])
```

5.7.2.44 timestampPath

```
timestampPath = os.path.join("data", datasetName, "radar.timestamps")
```

5.7.2.45 toSaveImgPath

```
toSaveImgPath = os.path.join(imgSavePath, f"{imgNo:04d}.jpg")
```

5.7.2.46 toSaveTrajPath

```
toSaveTrajPath = os.path.join(trajSavePath, f"{imgNo:04d}.jpg")
```

5.7.2.47 trajSavePath

```
trajSavePath
```

Initial value:

```
1 = os.path.join(".", "img", "track_klt_thresholding",
2                      datasetName + '_traj')
```

5.7.2.48 transformed_pts

```
transformed_pts = (R @ good_new.T + h).T
```

5.8 Mapping Namespace Reference

Classes

- class [Keyframe](#)
- class [Map](#)

Variables

- float [ROT_THRESHOLD](#) = 0.2
- float [TRANS_THRESHOLD](#) = 2.0
- float [TRANS_THRESHOLD_SQ](#) = [TRANS_THRESHOLD](#) * [TRANS_THRESHOLD](#)
- [RADAR_CART_CENTER](#) = None

5.8.1 Variable Documentation

5.8.1.1 RADAR_CART_CENTER

```
RADAR_CART_CENTER = None
```

5.8.1.2 ROT_THRESHOLD

```
float ROT_THRESHOLD = 0.2
```

5.8.1.3 TRANS_THRESHOLD

```
float TRANS_THRESHOLD = 2.0
```

5.8.1.4 TRANS_THRESHOLD_SQ

```
float TRANS_THRESHOLD_SQ = TRANS_THRESHOLD * TRANS_THRESHOLD
```

5.9 motionDistortion Namespace Reference

Classes

- class [MotionDistortionSolver](#)

Variables

- int [RADAR_SCAN_FREQUENCY](#) = 4
- bool [VERBOSE](#) = False

5.9.1 Variable Documentation

5.9.1.1 RADAR_SCAN_FREQUENCY

```
int RADAR_SCAN_FREQUENCY = 4
```

5.9.1.2 VERBOSE

```
bool VERBOSE = False
```

5.10 outlierRejection Namespace Reference

Functions

- tuple[np.ndarray, np.ndarray, np.ndarray] [rejectOutliers](#) (np.ndarray [prev_coord](#), np.ndarray [new_coord](#))

Variables

- float [DIST_THRESHOLD_M](#) = 0.5
- float [DIST_THRESHOLD_PX](#) = [DIST_THRESHOLD_M](#) / [RANGE_RESOLUTION_CART_M](#)
- float [DISTSQ_THRESHOLD_PX](#) = [DIST_THRESHOLD_PX](#) * [DIST_THRESHOLD_PX](#)
- bool [FORCE_OUTLIERS](#) = True
- int [n_points](#) = 100
- int [n_outliers](#) = int([n_points](#) * 0.2)
- int [theta_max_deg](#) = 20
- int [max_translation_m](#) = 5
- [prev_coord](#)
- [new_coord](#) = addNoise([new_coord](#), variance=[DIST_THRESHOLD_PX](#) / 10)
- [theta_deg](#)
- [trans_vec](#)
- [new_coord_perfect](#) = [new_coord](#).copy()
- [outlier_ind](#)
- [noiseToAdd](#)
- [pruned_prev_coord](#)
- [pruned_new_coord](#)
- [alpha](#)
- [show](#)
- [title_append](#)

5.10.1 Function Documentation

5.10.1.1 rejectOutliers()

```
tuple[np.ndarray, np.ndarray, np.ndarray] outlierRejection.rejectOutliers (
    np.ndarray prev_coord,
    np.ndarray new_coord )
```

@brief Reject outliers by using radar geometry to find dynamic/moving features

@details For the first and second feature set, form a graph G1 and G2, where each point is a feature, and each edge is the distance between the 2 points. Because of radar geometry, the distance between any 2 points in G1 should be the same (within threshold) as the same points in G2.

This is thus equivalent to forming an unweighted graph G, expressed as an adjacency matrix where if the difference in distance between i and j < thresh, then entry = 1, 0 otherwise. We then form the inlier set by finding the maximal clique in G.

@note It is assumed that the points correspond with each other

@param[in] prev_coord (K x 2) Coordinates of features which are being tracked from [x,y]
@param[in] new_coord (K x 2) New coordinates of features which are tracked to [x,y]

@return pruned_prev_coord (k x 2) Pruned previous coordinates
@return pruned_new_coord (k x 2) Pruned current/new coordinates
@return pruning_mask (K x 2) Pruning mask

5.10.2 Variable Documentation

5.10.2.1 alpha

alpha

5.10.2.2 DIST_THRESHOLD_M

float DIST_THRESHOLD_M = 0.5

5.10.2.3 DIST_THRESHOLD_PX

float DIST_THRESHOLD_PX = DIST_THRESHOLD_M / RANGE_RESOLUTION_CART_M

5.10.2.4 DISTSQ_THRESHOLD_PX

float DISTSQ_THRESHOLD_PX = DIST_THRESHOLD_PX * DIST_THRESHOLD_PX

5.10.2.5 FORCE_OUTLIERS

```
bool FORCE_OUTLIERS = True
```

5.10.2.6 max_translation_m

```
max_translation_m = 5
```

5.10.2.7 n_outliers

```
n_outliers = int(n_points * 0.2)
```

5.10.2.8 n_points

```
n_points = 100
```

5.10.2.9 new_coord

```
new_coord = addNoise(new_coord, variance=DIST_THRESHOLD_PX / 10)
```

5.10.2.10 new_coord_perfect

```
new_coord_perfect = new_coord.copy()
```

5.10.2.11 noiseToAdd

```
noiseToAdd
```

5.10.2.12 outlier_ind

```
outlier_ind
```


5.10.2.13 prev_coord

prev_coord

5.10.2.14 pruned_new_coord

pruned_new_coord

5.10.2.15 pruned_prev_coord

pruned_prev_coord

5.10.2.16 show

show

5.10.2.17 theta_deg

theta_deg

5.10.2.18 theta_max_deg

theta_max_deg = 20

5.10.2.19 title_append

title_append

5.10.2.20 trans_vec

trans_vec

5.11 parseData Namespace Reference

Functions

- Tuple[np.ndarray, float, float, np.ndarray, np.ndarray, np.ndarray] [extractDataFromRadarImage](#) (np.ndarray polarImgData, float maxRangeClipM=[MAX_RANGE_CLIP_DEFAULT](#))
- def [drawCVPoint](#) (np.ndarray img, [CartCoord](#) point, Tuple[int, int, int] point_color=(0, 0, 255))
- np.ndarray [convertCartesianImageToPolar](#) (np.ndarray [imgCart](#), bool logPolarMode=False, Tuple[int, int] shapeHW=None)
- np.ndarray [convertPolarImageToCartesian](#) (np.ndarray [imgPolar](#), bool logPolarMode=False, int downsample↔Factor=[DOWNSAMPLE_FACTOR](#), bool changeGlobalRangeResolution=False)
- def [convertPolarImgToLogPolar](#) (np.ndarray [imgPolar](#))
- Tuple[np.ndarray, float, float, np.ndarray, np.ndarray, np.ndarray] [getDataFromImgPathsByIndex](#) (List[str] imgPathArr, int index)
- np.ndarray [getPolarImageFromImgPaths](#) (List[str] imgPathArr, int index)
- np.ndarray [getCartImageFromImgPaths](#) (List[str] imgPathArr, int index)
- List[str] [getRadarImgPaths](#) (str [dataPath](#), str [timestampPath](#))
- np.ndarray [getRadarStreamPolar](#) (str [dataPath](#), str [timestampPath](#))

Variables

- float [RANGE_RESOLUTION_M](#) = 0.0432
- int [DOWNSAMPLE_FACTOR](#) = 2
- float [RANGE_RESOLUTION_CART_M](#) = [RANGE_RESOLUTION_M](#) * [DOWNSAMPLE_FACTOR](#)
- float [MAX_RANGE_CLIP_DEFAULT](#) = 87.5
- int [datasetName](#) = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- [dataPath](#) = os.path.join("data", [datasetName](#), "radar")
- [timestampPath](#) = os.path.join("data", [datasetName](#), "radar.timestamps")
- np.ndarray [streamArr](#) = [getRadarStreamPolar](#)([dataPath](#), [timestampPath](#))
- np.ndarray [nlmgs](#) = [streamArr](#).shape[2]
- np.ndarray [imgPolar](#) = [streamArr](#)[:, :, i]
- np.ndarray [imgCart](#) = [convertPolarImageToCartesian](#)([imgPolar](#))
- [c](#) = cv2.waitKey(100)

5.11.1 Function Documentation

5.11.1.1 [convertCartesianImageToPolar\(\)](#)

```
np.ndarray parseData.convertCartesianImageToPolar (
    np.ndarray imgCart,
    bool logPolarMode = False,
    Tuple[int, int] shapeHW = None )
```

```
@brief Converts Cartesian image to (potentially log) polar
@param[in] imgPolar Polar image to convert
@param[in] logPolarMode Whether to convert in log-polar mode

@return imgCart Converted Cartesian image
```

5.11.1.2 convertPolarImageToCartesian()

```
np.ndarray parseData.convertPolarImageToCartesian (
    np.ndarray imgPolar,
    bool logPolarMode = False,
    int downsampleFactor = DOWNSAMPLE_FACTOR,
    bool changeGlobalRangeResolution = False )

@brief Converts polar image to Cartesian formats
@param[in] imgPolar Polar image to convert
@param[in] logPolarMode Whether to convert in log-polar mode
@param[in] downsampleFactor How much to downsample Cartesian image for performance improvements
@param[in] changeGlobalRangeResolution Whether or not to change the
    global range resolution needed
    for accurate px to m conversions

@return imgCart Converted Cartesian image
```

5.11.1.3 convertPolarImgToLogPolar()

```
def parseData.convertPolarImgToLogPolar (
    np.ndarray imgPolar )

@brief Convert an image in polar form into log-polar form
@note Involves converting from polar to Cartesian to back again
@see convertPolarImageToCartesian(), convertCartesianImageToPolar()
```

5.11.1.4 drawCVPoint()

```
def parseData.drawCVPoint (
    np.ndarray img,
    CartCoord point,
    Tuple[int, int, int] point_color = (0, 0, 255) )
```

5.11.1.5 extractDataFromRadarImage()

```
Tuple[np.ndarray, float, float, np.ndarray, np.ndarray, np.ndarray] parseData.extractData←
FromRadarImage (
    np.ndarray polarImgData,
    float maxRangeClipM = MAX_RANGE_CLIP_DEFAULT )

@brief Decode a single Oxford Radar RobotCar Dataset radar example
@param[in] polarImgData cv image
@param[in] maxRangeClipM Max range to clip data, in meters. Negative number for no clip
@return
    range_azimuth_data (np.ndarray): Radar power readings along each azimuth
    range_resolution (float): Range resolution of the polar radar data (metres per pixel)
    azimuth_resolution (float): Azimuth resolution of the polar radar data (radians per pixel)
    azimuths (np.ndarray): Rotation for each polar radar azimuth (radians)
    valid (np.ndarray) Mask of whether azimuth data is an original sensor reading or interpolated from adjacent
    azimuths
    timestamps (np.ndarray): Timestamp for each azimuth in int64 (UNIX time)
```

5.11.1.6 getCartImageFromImgPaths()

```
np.ndarray parseData.getCartImageFromImgPaths (
    List[str] imgPathArr,
    int index )

@brief Get polar image from image path array, indexing accordingly
@param[in] imgPathArr List of image path as strings
@param[in] index Index to index into

@return imgCart Cartesian image
```

5.11.1.7 getDataFromImgPathsByIndex()

```
Tuple[np.ndarray, float, float, np.ndarray, np.ndarray, np.ndarray] parseData.getDataFromImg↵
PathsByIndex (
    List[str] imgPathArr,
    int index )

@brief Get information from image path array, indexing accordingly
@param[in] imgPathArr List of image path as strings
@param[in] index Index to index into

@return
    imgPolar (np.ndarray): Radar power readings along each azimuth
    azimuth_resolution (float): Azimuth resolution of the polar radar data (radians per pixel)
    range_resolution (float): Range resolution of the polar radar data (metres per pixel)
    azimuths (np.ndarray): Rotation for each polar radar azimuth (radians)
    valid (np.ndarray) Mask of whether azimuth data is an original sensor reading or interpolated from adjacent
        azimuths
    timestamps (np.ndarray): Timestamp for each azimuth in int64 (UNIX time)
```

5.11.1.8 getPolarImageFromImgPaths()

```
np.ndarray parseData.getPolarImageFromImgPaths (
    List[str] imgPathArr,
    int index )

@brief Get polar image from image path array, indexing accordingly
@param[in] imgPathArr List of image path as strings
@param[in] index Index to index into

@return imgPolar Polar image
```

5.11.1.9 getRadarImgPaths()

```
List[str] parseData.getRadarImgPaths (
    str dataPath,
    str timestampPath )

@brief Obtain list of radar image paths

@param[in] dataPath Path to radar image data
@param[in] timestampPath Path to radar timestamp data

@return list of strings containing paths to radar image
```

5.11.1.10 getRadarStreamPolar()

```
np.ndarray parseData.getRadarStreamPolar (
    str dataPath,
    str timestampPath )

@brief Returns np array of radar images in POLAR format
@param[in] dataPath Path to radar image data
@param[in] timestampPath Path to radar timestamp data

@return radar range-azimuth image (W x H x N)
```

5.11.2 Variable Documentation

5.11.2.1 c

```
c = cv2.waitKey(100)
```

5.11.2.2 dataPath

```
dataPath = os.path.join("data", datasetName, "radar")
```

5.11.2.3 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.11.2.4 DOWNSAMPLE_FACTOR

```
int DOWNSAMPLE_FACTOR = 2
```

5.11.2.5 imgCart

```
np.ndarray imgCart = convertPolarImageToCartesian(imgPolar)
```

5.11.2.6 imgPolar

```
np.ndarray imgPolar = streamArr[:, :, i]
```

5.11.2.7 MAX_RANGE_CLIP_DEFAULT

```
float MAX_RANGE_CLIP_DEFAULT = 87.5
```

5.11.2.8 nImgs

```
np.ndarray nImgs = streamArr.shape[2]
```

5.11.2.9 RANGE_RESOLUTION_CART_M

```
float RANGE_RESOLUTION_CART_M = RANGE_RESOLUTION_M * DOWNSAMPLE_FACTOR
```

5.11.2.10 RANGE_RESOLUTION_M

```
float RANGE_RESOLUTION_M = 0.0432
```

5.11.2.11 streamArr

```
np.ndarray streamArr = getRadarStreamPolar(dataPath, timestampPath)
```

5.11.2.12 timestampPath

```
timestampPath = os.path.join("data", datasetName, "radar.timestamps")
```

5.12 PoseGraphLib Namespace Reference

Classes

- class [BundleAdjustment](#)
- class [PoseGraphOptimization](#)

5.13 RawROAMSystem Namespace Reference

Classes

- class [RawROAMSystem](#)

Variables

- [RADAR_CART_CENTER](#) = np.array([1012, 1012])
- int [wantToPlot](#) = -1
- int [datasetName](#) = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- int [startSeqInd](#) = int(sys.argv[2]) if len(sys.argv) > 2 else 0
- int [endSeqInd](#) = int(sys.argv[3]) if len(sys.argv) > 3 else -1
- int [REMOVE_OLD_RESULTS](#) = bool(int(sys.argv[4])) if len(sys.argv) > 4 else False
- dictionary [paramFlags](#)
- [system](#)
- [imgSavePath](#) = system.filePaths["imgSave"]
- [trajSavePath](#) = system.filePaths["trajSave"]

5.13.1 Variable Documentation

5.13.1.1 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.13.1.2 endSeqInd

```
int endSeqInd = int(sys.argv[3]) if len(sys.argv) > 3 else -1
```

5.13.1.3 imgSavePath

```
imgSavePath = system.filePaths["imgSave"]
```

5.13.1.4 paramFlags

```
dictionary paramFlags
```

Initial value:

```
1 = {  
2     "rejectOutliers": True,  
3     "useFMT": False,  
4     # Below all currently unused actually  
5     "useANMS": False,  
6     "correctMotionDistortion": False  
7 }
```

5.13.1.5 RADAR_CART_CENTER

```
RADAR_CART_CENTER = np.array([1012, 1012])
```

5.13.1.6 REMOVE_OLD_RESULTS

```
int REMOVE_OLD_RESULTS = bool(int(sys.argv[4])) if len(sys.argv) > 4 else False
```

5.13.1.7 startSeqInd

```
int startSeqInd = int(sys.argv[2]) if len(sys.argv) > 2 else 0
```

5.13.1.8 system

```
system
```

Initial value:

```
1 = RawROAMSystem(datasetName,  
2                   paramFlags=paramFlags,  
3                   hasGroundTruth=True)
```


5.13.1.9 trajSavePath

```
trajSavePath = system.filePaths["trajSave"]
```

5.13.1.10 wantToPlot

```
int wantToPlot = -1
```

5.14 testMotionDistortion Namespace Reference

Variables

- int `N` = 100
- float `outlier_rate` = 0.4
- bool `noisy` = False
- bool `useOld` = False
- int `frequency` = 4
- int `period` = 1 / `frequency`
- `groundTruth`
- `currentFrame`
- `theta_deg`
- `h`
- `n_points`
- int `velocity` = np.array([`h`[0, 0], `h`[1, 0], `theta_deg`]) / `period`
- `distorted` = distort(`currentFrame`, `velocity`, `frequency`, `h`)
- `outlier_ind`
- `noiseToAdd`
- `R_fit` = `A_inv`[:, :2]
- `h_fit` = `A_inv`[:, 2:]
- `A`
- `A_inv` = np.linalg.inv(`A`)
- int `theta_fit` = np.arctan2(`R_fit`[1, 0], `R_fit`[0, 0]) * 180 / np.pi
- tuple `srcCoord2` = (`R_fit` @ `distorted.T` + `h_fit`).T
- `title_append`
- `alpha`
- `clear`
- `False`
- `show`
- `plotDisplace`
- `T_wj0` = np.eye(3)
- `p_w` = `groundTruth`
- `p_jt` = `distorted`
- int `v_j0` = np.array([`h_fit`[0,0], `h_fit`[1,0], `theta_fit` * np.pi / 180]) / `period`
- `T_wj`
- `cov_p` = np.diag([4, 4])
- `cov_v` = np.diag([1, 1, (5 * np.pi / 180) ** 2])
- `MDS` = `MotionDistortionSolver`(`T_wj0`, `p_w`, `p_jt`, `v_j0`, `T_wj`, `cov_p`, `cov_v`)
- `undistorted` = `MDS.undistort`(`v_j0`)
- tuple `srcCoord3` = (`R_fit` @ `undistorted.T` + `h_fit`).T
- `params` = `MDS.optimize_library`()
- `final_undistorted` = `MDS.undistort`(`params`[:3])
- `transform` = `convertPoseToTransform`(`params`[3:])
- tuple `solution` = (`transform` @ `final_undistorted`)[:, :2, 0]

5.14.1 Variable Documentation

5.14.1.1 A

A

Initial value:

```
1 = np.block([[R_fit, h_fit],
2             [np.zeros((1, 2)), 1]])
```

5.14.1.2 A_inv

```
A_inv = np.linalg.inv(A)
```

5.14.1.3 alpha

alpha

5.14.1.4 clear

clear

5.14.1.5 cov_p

```
cov_p = np.diag([4, 4])
```

5.14.1.6 cov_v

```
cov_v = np.diag([1, 1, (5 * np.pi / 180) ** 2])
```

5.14.1.7 currentFrame

currentFrame

5.14.1.8 distorted

```
distorted = distort(currentFrame, velocity, frequency, h)
```

5.14.1.9 False

False

5.14.1.10 final_undistorted

```
final_undistorted = MDS.undistort(params[:3])
```

5.14.1.11 frequency

```
int frequency = 4
```

5.14.1.12 groundTruth

groundTruth

5.14.1.13 h

h

5.14.1.14 h_fit

```
h_fit = A_inv[:2, 2:]
```

5.14.1.15 MDS

```
MDS = MotionDistortionSolver(T_wj0, p_w, p_jt, v_j0, T_wj, cov_p, cov_v)
```

5.14.1.16 N

```
int N = 100
```

5.14.1.17 n_points

```
n_points
```

5.14.1.18 noiseToAdd

```
noiseToAdd
```

5.14.1.19 noisy

```
bool noisy = False
```

5.14.1.20 outlier_ind

```
outlier_ind
```

5.14.1.21 outlier_rate

```
float outlier_rate = 0.4
```

5.14.1.22 p_jt

```
p_jt = distorted
```

5.14.1.23 p_w

```
p_w = groundTruth
```

5.14.1.24 params

```
params = MDS.optimize_library()
```

5.14.1.25 period

```
int period = 1 / frequency
```

5.14.1.26 plotDisplace

```
plotDisplace
```

5.14.1.27 R_fit

```
R_fit = A\_inv[:2, :2]
```

5.14.1.28 show

```
show
```

5.14.1.29 solution

```
tuple solution = (transform @ final\_undistorted)[: , :2, 0]
```

5.14.1.30 srcCoord2

```
tuple srcCoord2 = (R\_fit @ distorted.T + h\_fit).T
```

5.14.1.31 srcCoord3

```
tuple srcCoord3 = (R_fit @ undistorted.T + h_fit).T
```

5.14.1.32 T_wj

```
T_wj
```

Initial value:

```
1 = np.block([[R_fit, h_fit],
2             [np.zeros((2,)), 1]])
```

5.14.1.33 T_wj0

```
T_wj0 = np.eye(3)
```

5.14.1.34 theta_deg

```
theta_deg
```

5.14.1.35 theta_fit

```
int theta_fit = np.arctan2(R_fit[1, 0], R_fit[0, 0]) * 180 / np.pi
```

5.14.1.36 title_append

```
title_append
```

5.14.1.37 transform

```
transform = convertPoseToTransform(params[3:])
```

5.14.1.38 undistorted

```
undistorted = MDS.undistort(v_j0)
```

5.14.1.39 useOld

```
bool useOld = False
```

5.14.1.40 v_j0

```
int v_j0 = np.array([h_fit[0,0], h_fit[1,0], theta_fit * np.pi / 180]) / period
```

5.14.1.41 velocity

```
int velocity = np.array([h[0, 0], h[1, 0], theta_deg]) / period
```

5.15 testTransform Namespace Reference**Variables**

- int `N` = 100
- float `outlier_rate` = 0.4
- bool `noisy` = `False`
- bool `useOld` = `False`
- `srcCoord`
- `targetCoord`
- `theta_deg`
- `h`
- `n_points`
- `outlier_ind`
- `noiseToAdd`
- `R_fit` = `A_inv[:2, :2]`
- `h_fit` = `A_inv[:2, 2:]`
- `A`
- `A_inv` = `np.linalg.inv(A)`
- int `theta_fit` = `np.arctan2(R_fit[1, 0], R_fit[0, 0]) * 180 / np.pi`
- tuple `srcCoord2` = `(R_fit @ targetCoord.T + h_fit).T`
- `title_append`
- `alpha`
- `clear`
- `False`
- `show`

5.15.1 Variable Documentation

5.15.1.1 A

A

Initial value:

```
1 = np.block([[R_fit, h_fit],
2             [np.zeros((1, 2)), 1]])
```

5.15.1.2 A_inv

```
A_inv = np.linalg.inv(A)
```

5.15.1.3 alpha

alpha

5.15.1.4 clear

clear

5.15.1.5 False

False

5.15.1.6 h

h

5.15.1.7 h_fit

```
h_fit = A_inv[:2, 2:]
```

5.15.1.8 N

```
int N = 100
```

5.15.1.9 n_points

```
n_points
```

5.15.1.10 noiseToAdd

```
noiseToAdd
```

5.15.1.11 noisy

```
bool noisy = False
```

5.15.1.12 outlier_ind

```
outlier_ind
```

5.15.1.13 outlier_rate

```
float outlier_rate = 0.4
```

5.15.1.14 R_fit

```
R_fit = A_inv[:2, :2]
```

5.15.1.15 show

```
show
```

5.15.1.16 srcCoord

```
srcCoord
```

5.15.1.17 srcCoord2

```
tuple srcCoord2 = (R_fit @ targetCoord.T + h_fit).T
```

5.15.1.18 targetCoord

```
targetCoord
```

5.15.1.19 theta_deg

```
theta_deg
```

5.15.1.20 theta_fit

```
int theta_fit = np.arctan2(R_fit[1, 0], R_fit[0, 0]) * 180 / np.pi
```

5.15.1.21 title_append

```
title_append
```

5.15.1.22 useOld

```
bool useOld = False
```

5.16 Tracker Namespace Reference

Classes

- class [Tracker](#)

5.17 trajectoryPlotting Namespace Reference

Classes

- class [Trajectory](#)

Functions

- def [computePosesRMSE](#) ([gtPoses](#), [estPoses](#))
- def [plotGtAndEstTrajectory](#) ([gtTraj](#), [estTraj](#), title='GT and EST Trajectories', info=None, savePath=None, arrow=False)
- def [getGroundTruthTrajectory](#) ([gtPath](#))
- def [getGroundTruthTrajectoryGPS](#) ([gtPath](#))

Variables

- int [datasetName](#) = sys.argv[1] if len(sys.argv) > 1 else "tiny"
- [timestampPath](#) = os.path.join("data", datasetName, "radar.timestamps")
- [gtPath](#) = os.path.join("data", datasetName, "gps", "gps.csv")
- def [gtTraj](#) = [getGroundTruthTrajectoryGPS](#)([gtPath](#))
- [keyframe_timestamps](#)
- def [estPoses](#) = [gtTraj](#).getPoseAtTimes([keyframe_timestamps](#))
- [noise](#)
- [estTraj](#) = [Trajectory](#)([keyframe_timestamps](#), [estPoses](#))
- [block](#)

5.17.1 Function Documentation

5.17.1.1 [computePosesRMSE\(\)](#)

```
def trajectoryPlotting.computePosesRMSE (
    gtPoses,
    estPoses )
```

@brief Compute the Root Mean Square Error between the prediction and the actual poses

5.17.1.2 getGroundTruthTrajectory()

```
def trajectoryPlotting.getGroundTruthTrajectory (
    gtPath )

@brief Returns ground truth trajectory given radar_odometry.csv
@param[in] gtPath Path to ground truth file
@return Trajectory object
```

5.17.1.3 getGroundTruthTrajectoryGPS()

```
def trajectoryPlotting.getGroundTruthTrajectoryGPS (
    gtPath )

@brief Returns ground truth trajectory given gps.csv
@param[in] gtPath Path to ground truth file
@return Trajectory object
```

5.17.1.4 plotGtAndEstTrajectory()

```
def trajectoryPlotting.plotGtAndEstTrajectory (
    gtTraj,
    estTraj,
    title = 'GT and EST Trajectories',
    info = None,
    savePath = None,
    arrow = False )

@brief Plot ground truth trajectory and estimated trajectory
@param[in] gtTrajectory Ground truth trajectory
@param[in] estTrajectory Estimated trajectory
@param[in] title Title of the plot
@param[in] info Extra information to write in text
```

5.17.2 Variable Documentation

5.17.2.1 block

block

5.17.2.2 datasetName

```
int datasetName = sys.argv[1] if len(sys.argv) > 1 else "tiny"
```

5.17.2.3 estPoses

```
def estPoses = gtTraj.getPoseAtTimes(keyframe_timestamps)
```

5.17.2.4 estTraj

```
estTraj = Trajectory(keyframe_timestamps, estPoses)
```

5.17.2.5 gtPath

```
gtPath = os.path.join("data", datasetName, "gps", "gps.csv")
```

5.17.2.6 gtTraj

```
def gtTraj = getGroundTruthTrajectoryGPS(gtPath)
```

5.17.2.7 keyframe_timestamps

```
keyframe_timestamps
```

Initial value:

```
1 = np.arange(  
2     gtTraj.timestamps[0], gtTraj.timestamps[-1],  
3     (gtTraj.timestamps[-1] - gtTraj.timestamps[0]) / 1000)
```

5.17.2.8 noise

```
noise
```

Initial value:

```
1 = np.random.multivariate_normal(mean=(.01, .05),  
2     cov=np.array([[.8, .2], [.2, .8]]) *  
3     1e-2,  
4     size=(keyframe_timestamps.shape[0]))
```

5.17.2.9 timestampPath

```
timestampPath = os.path.join("data", datasetName, "radar.timestamps")
```

5.18 utils Namespace Reference

Functions

- def [tic](#) ()
- def [toc](#) (tic)
- def [f_arr](#) (xs, th_deg=False)
- def [radarImgPathToTimestamp](#) (radarImgPath)
- def [normalize_angles](#) (th)
- def [getRotationMatrix](#) (th, degrees=False)
- def [convertPoseToTransform](#) (poses)
- def [convertTransformToPose](#) (pose_transforms)
- def [flatten](#) (x)
- def [convertRandHtoDeltas](#) (R, h)
- def [quiver](#) (poses, c='r', label=None)
- def [plt_full_extent](#) (ax, pad=0.0)
- def [plt_savefig_by_axis](#) (filePath, fig, ax, pad=0.0)
- def [invert_transform](#) (T)
- def [homogenize](#) (points)

5.18.1 Function Documentation

5.18.1.1 convertPoseToTransform()

```
def utils.convertPoseToTransform (
    poses )

@param[in] poses np.ndarray of (3,) or (N x 3)
@return pose_transforms np.ndarray of (3 x 3) or (N x 3 x 3)
```

5.18.1.2 convertRandHtoDeltas()

```
def utils.convertRandHtoDeltas (
    R,
    h )
```

5.18.1.3 convertTransformToPose()

```
def utils.convertTransformToPose (
    pose_transforms )

@param[in] pose_transforms np.ndarray of (3 x 3) or (N x 3 x 3)
@return poses np.ndarray of (3,) or (N x 3)
```

5.18.1.4 f_arr()

```
def utils.f_arr (
    xs,
    th_deg = False )
```

5.18.1.5 flatten()

```
def utils.flatten (
    x )
```

5.18.1.6 getRotationMatrix()

```
def utils.getRotationMatrix (
    th,
    degrees = False )
```

5.18.1.7 homogenize()

```
def utils.homogenize (
    points )
```

5.18.1.8 invert_transform()

```
def utils.invert_transform (
    T )
```

5.18.1.9 `normalize_angles()`

```
def utils.normalize_angles (
    th )
```

Normalize an angle to be between $-\pi$ and π

5.18.1.10 `plt_full_extent()`

```
def utils.plt_full_extent (
    ax,
    pad = 0.0 )
```

@brief Get the full extent of a plt axes, including axes labels, tick labels, and titles.

5.18.1.11 `plt_savefig_by_axis()`

```
def utils.plt_savefig_by_axis (
    filePath,
    fig,
    ax,
    pad = 0.0 )
```

@brief Save a plt figure by extent of its axis (allows us to save subplots)
@param[in] *filePath* Path to save figure to
@param[in] *fig* Overall figure containing axis
@param[in] *ax* Axis to save

5.18.1.12 `quiver()`

```
def utils.quiver (
    poses,
    c = 'r',
    label = None )
```

5.18.1.13 `radarImgPathToTimestamp()`

```
def utils.radarImgPathToTimestamp (
    radarImgPath )
```

eg: `radarImgPathToTimestamp('data\\tiny\\radar\\1547131046353776.png')` -> 1547131046353776

5.18.1.14 tic()

```
def utils.tic ( )
```

5.18.1.15 toc()

```
def utils.toc (
    tic )
```


Chapter 6

Class Documentation

6.1 BundleAdjustment Class Reference

Public Member Functions

- def `__init__` (self)
- def `optimize` (self, max_iterations=10)
- def `add_pose` (self, pose_id, pose, cam, fixed=False)
- def `add_point` (self, point_id, point, fixed=False, marginalized=True)
- def `add_edge` (self, point_id, pose_id, measurement, information=np.eye(2), robust_kernel=g2o.RobustKernelHuber(np.sqrt(5.991)))
- def `get_pose` (self, pose_id)
- def `get_point` (self, point_id)

6.1.1 Constructor & Destructor Documentation

6.1.1.1 `__init__()`

```
def __init__ (  
    self )
```

6.1.2 Member Function Documentation

6.1.2.1 `add_edge()`

```
def add_edge (  
    self,  
    point_id,  
    pose_id,  
    measurement,  
    information = np.eye(2),  
    robust_kernel = g2o.RobustKernelHuber(  
        np.sqrt(5.991)) )
```

6.1.2.2 add_point()

```
def add_point (
    self,
    point_id,
    point,
    fixed = False,
    marginalized = True )
```

6.1.2.3 add_pose()

```
def add_pose (
    self,
    pose_id,
    pose,
    cam,
    fixed = False )
```

6.1.2.4 get_point()

```
def get_point (
    self,
    point_id )
```

6.1.2.5 get_pose()

```
def get_pose (
    self,
    pose_id )
```

6.1.2.6 optimize()

```
def optimize (
    self,
    max_iterations = 10 )
```

6.2 CartCoord Class Reference

Public Member Functions

- def `__init__` (self, float `x`, float `y`)
- def `__str__` (self)
- None `addCoord` (self, other)
- None `add` (self, float `dx`, float `dy`)
- None `sub` (self, float `dx`, float `dy`)
- None `scale` (self, float `scaleFactor`)
- None `scaleX` (self, float `scaleFactor`)
- None `scaleY` (self, float `scaleFactor`)
- float `getX` (self)
- float `getY` (self)
- float `getDistance` (self, other)
- def `getAngle` (self, other)
- tuple `asTuple` (self)

Public Attributes

- `x`
- `y`

6.2.1 Detailed Description

Creates a point on a Cartesian coordinate plane with values `x` and `y`.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 `__init__()`

```
def __init__ (  
    self,  
    float x,  
    float y )
```

Defines `x` and `y` variables

6.2.3 Member Function Documentation

6.2.3.1 `__str__()`

```
def __str__ (
    self )
```

6.2.3.2 `add()`

```
None add (
    self,
    float dx,
    float dy )
```

6.2.3.3 `addCoord()`

```
None addCoord (
    self,
    other )
```

6.2.3.4 `asTuple()`

```
tuple asTuple (
    self )
```

6.2.3.5 `getAngle()`

```
def getAngle (
    self,
    other )
```

6.2.3.6 `getDistance()`

```
float getDistance (
    self,
    other )
```

6.2.3.7 getX()

```
float getX (
    self )
```

6.2.3.8 getY()

```
float getY (
    self )
```

6.2.3.9 scale()

```
None scale (
    self,
    float scaleFactor )
```

6.2.3.10 scaleX()

```
None scaleX (
    self,
    float scaleFactor )
```

6.2.3.11 scaleY()

```
None scaleY (
    self,
    float scaleFactor )
```

6.2.3.12 sub()

```
None sub (
    self,
    float dx,
    float dy )
```

6.2.4 Member Data Documentation

6.2.4.1 x

x

6.2.4.2 y

y

6.3 Keyframe Class Reference

Public Member Functions

- None `__init__` (self, np.ndarray globalPose, np.ndarray [featurePointsLocal](#), np.ndarray [radarPolarImg](#), np.ndarray [velocity](#))
- None `updateInfo` (self, np.ndarray globalPose, np.ndarray [featurePointsLocal](#), np.ndarray [radarPolarImg](#), np.ndarray [velocity](#))
- None `copyFromOtherKeyframe` (self, keyframe)
- np.ndarray `convertFeaturesLocalToGlobal` (self, np.ndarray [featurePointsLocal](#))
- np.ndarray `getPrunedFeaturesGlobalPosition` (self)
- None `pruneFeaturePoints` (self, np.ndarray corrStatus)

Public Attributes

- [pose](#)
- [radarPolarImg](#)
- [featurePointsLocal](#)
- [prunedFeaturePoints](#)
- [pointCloud](#)
- [velocity](#)
- [featurePointsLocalUndistorted](#)
- [prunedUndistortedLocals](#)

6.3.1 Constructor & Destructor Documentation

6.3.1.1 `__init__()`

```
None __init__ (
    self,
    np.ndarray globalPose,
    np.ndarray featurePointsLocal,
    np.ndarray radarPolarImg,
    np.ndarray velocity )

@brief Keyframe class. Contains pose, feature points and point cloud information
@param[in] globalPose      (3 x 1) Pose information [x, y, th] in global coordinates,
                           units of [m, m, rad] # TODO: Confirm these units
@param[in] featurePointsLocal (K x 2) Tracked feature points from previous keyframe,
                           in local coordinates (pixels)
@param[in] radarPolarImg    (M x N) Radar polar (range-azimuth) image

@see updateInfo()
```


6.3.2 Member Function Documentation

6.3.2.1 convertFeaturesLocalToGlobal()

```
np.ndarray convertFeaturesLocalToGlobal (
    self,
    np.ndarray featurePointsLocal )

@brief Local feature points to convert into global coordinates, given internal pose
@param[in] featurePointsLocal    (K x 2) Tracked feature points from previous keyframe,
                                in local coordinates (pixels)
@return featurePointsGlobal      (K x 2) Feature points in global coordinates, meters
```

6.3.2.2 copyFromOtherKeyframe()

```
None copyFromOtherKeyframe (
    self,
    keyframe )
```

6.3.2.3 getPrunedFeaturesGlobalPosition()

```
np.ndarray getPrunedFeaturesGlobalPosition (
    self )

@brief Get global position of pruned features (stored internally)
@return Global position of pruned features (K x 2)
```

6.3.2.4 pruneFeaturePoints()

```
None pruneFeaturePoints (
    self,
    np.ndarray corrStatus )

@brief Prune feature points based on correspondence status
@param[in] corrStatus
@note In place changing of 'self.prunedFeaturePoints' function, which aims to track and prune away the feature
```

6.3.2.5 updateInfo()

```

None updateInfo (
    self,
    np.ndarray globalPose,
    np.ndarray featurePointsLocal,
    np.ndarray radarPolarImg,
    np.ndarray velocity )

@brief Update internal information: pose, feature points and point cloud information
@param[in] globalPose      (3 x 1) Pose information [x, y, th] in global coordinates,
                           units of [m, m, rad] # TODO: Confirm these units
@param[in] featurePointsLocal (K x 2) Tracked feature points from previous keyframe,
                           in local coordinates (pixels)
@param[in] radarPolarImg    (M x N) Radar polar (range-azimuth) image

```

6.3.3 Member Data Documentation

6.3.3.1 featurePointsLocal

featurePointsLocal

6.3.3.2 featurePointsLocalUndistorted

featurePointsLocalUndistorted

6.3.3.3 pointCloud

pointCloud

6.3.3.4 pose

pose

6.3.3.5 prunedFeaturePoints

prunedFeaturePoints

6.3.3.6 prunedUndistortedLocals

prunedUndistortedLocals

6.3.3.7 radarPolarImg

radarPolarImg

6.3.3.8 velocity

velocity

6.4 Map Class Reference

Public Member Functions

- None `__init__` (self, str `sequenceName`, `Trajectory` `estTraj`, list[str] `imgPathArr`, dict[str] `filePaths`)
- def `updateInternalTraj` (self, `Trajectory` `traj`)
- bool `isGoodKeyframe` (self, `Keyframe` `keyframe`)
- None `addKeyframe` (self, `Keyframe` `keyframe`)
- None `plot` (self, plt.figure `fig`, int `subsampleFactor`=5, bool `show`=False)

Public Attributes

- `sequenceName`
- `imgPathArr`
- `sequenceSize`
- `filePaths`
- `estTraj`
- `mapPoints`
- `keyframes`

6.4.1 Constructor & Destructor Documentation

6.4.1.1 `__init__()`

```
None __init__ (
    self,
    str sequenceName,
    Trajectory estTraj,
    list[str] imgPathArr,
    dict[str] filePaths )
```

6.4.2 Member Function Documentation

6.4.2.1 addKeyframe()

```
None addKeyframe (
    self,
    Keyframe keyframe )
```

@brief Add a keyframe to the running pose graph
@param[in] keyframe Keyframe to add

6.4.2.2 isGoodKeyframe()

```
bool isGoodKeyframe (
    self,
    Keyframe keyframe )
```

@brief Check if a keyframe is good for adding using information about relative rotation and translation
@return If keyframe passes checks

6.4.2.3 plot()

```
None plot (
    self,
    plt.figure fig,
    int subsampleFactor = 5,
    bool show = False )
```

@brief Plot map points on plt figure
@param[in] fig plt figure to plot on @todo Currently unused
@param[in] subsampleFactor Subsampling amount to do for feature points plotting
Controls density of plotted points. Higher = less dense
@param[in] show Whether to plt.show()

6.4.2.4 updateInternalTraj()

```
def updateInternalTraj (
    self,
    Trajectory traj )
```

6.4.3 Member Data Documentation

6.4.3.1 estTraj

estTraj

6.4.3.2 filePaths

filePaths

6.4.3.3 imgPathArr

imgPathArr

6.4.3.4 keyframes

keyframes

6.4.3.5 mapPoints

mapPoints

6.4.3.6 sequenceName

sequenceName

6.4.3.7 sequenceSize

sequenceSize

6.5 MotionDistortionSolver Class Reference

Public Member Functions

- def `__init__` (self, `T_wj0`, `p_w`, `p_jt`, `T_wj`, `sigma_p`, `sigma_v`, frequency=`RADAR_SCAN_FREQUENCY`)
- def `__init__` (self, `sigma_p`, `sigma_v`, frequency=`RADAR_SCAN_FREQUENCY`)
- def `update_problem` (self, `T_wj0`, `p_w`, `p_jt`, `T_wj`, `debug`=False)
- def `infer_velocity` (self, transform)
- def `expected_observed_pts` (self, `T_wj`)
- def `error_vector` (self, params)
- def `error` (self, `v_j`, `T_wj`)
- def `jacobian_vector` (self, params)
- def `jacobian` (self, `v_j`, `T_wj`)
- def `optimize` (self, max_iters=20)
- def `optimize_library` (self)

Static Public Member Functions

- def `compute_time_deltas` (period, points)
- def `undistort` (`v_j`, points, period=1/`RADAR_SCAN_FREQUENCY`, times=None)

Public Attributes

- `T_wj0`
- `T_wj0_inv`
- `p_w`
- `p_jt`
- `T_wj_initial`
- `total_scan_time`
- `v_j_initial`
- `sigma_p`
- `sigma_v`
- `info_vector`
- `dT`
- `debug`

6.5.1 Constructor & Destructor Documentation

6.5.1.1 `__init__()` [1/2]

```
def __init__ (
    self,
    T_wj0,
    p_w,
    p_jt,
    T_wj,
    sigma_p,
    sigma_v,
    frequency = RADAR_SCAN_FREQUENCY )
```

6.5.1.2 `__init__()` [2/2]

```
def __init__ (
    self,
    sigma_p,
    sigma_v,
    frequency = RADAR_SCAN_FREQUENCY )
```

6.5.2 Member Function Documentation

6.5.2.1 `compute_time_deltas()`

```
def compute_time_deltas (
    period,
    points ) [static]
```

Get the time deltas for each point. This depends solely on where the points are in scan angle. The further away from center, the greater the time displacement, and therefore the higher time delta. We use this time delta to help us transform the points into an undistorted frame. Note that this is an estimate computed from distorted images. It is a good idea to re-run this function once an undistorted frame is obtained for better estimates.

6.5.2.2 `error()`

```
def error (
    self,
    v_j,
    T_wj )
```

Return the Cauchy robust error between the undistorted points and their estimated observed positions and the velocity error.

6.5.2.3 `error_vector()`

```
def error_vector (
    self,
    params )
```

Because we are optimizing over rotations, we choose to keep the rotation in a theta form, we have to do matrix exponential in here to convert into the SO(1) form, then augment to the rotation-translation transform

6.5.2.4 expected_observed_pts()

```
def expected_observed_pts (
    self,
    T_wj )
```

Returns the estimated positions of points based on their world location estimates and the current best fit transform

6.5.2.5 infer_velocity()

```
def infer_velocity (
    self,
    transform )
```

6.5.2.6 jacobian()

```
def jacobian (
    self,
    v_j,
    T_wj )
```

Compute the Jacobian. This has two parts, as defined by the RadarSLAM paper:

```
J_p - gradient of point error and velocity error wrt pose terms Tx,
      Ty, Ttheta
J_v - gradient of point error and velocity error wrt velocity terms
      vx, vy, vtheta
```

6.5.2.7 jacobian_vector()

```
def jacobian_vector (
    self,
    params )
```

6.5.2.8 optimize()

```
def optimize (
    self,
    max_iters = 20 )
```


6.5.2.9 optimize_library()

```
def optimize_library (
    self )
```

Optimize using the LM implementation in the scipy library.

6.5.2.10 undistort()

```
def undistort (
    v_j,
    points,
    period = 1 / RADAR\_SCAN\_FREQUENCY,
    times = None ) [static]
```

Computes a new set of undistorted observed points, based on the current best estimate of v_T , T_{wj} , dT

6.5.2.11 update_problem()

```
def update_problem (
    self,
    T_wj0,
    p_w,
    p_jt,
    T_wj,
    debug = False )
```

6.5.3 Member Data Documentation

6.5.3.1 debug

debug

6.5.3.2 dT

dT

6.5.3.3 info_vector

info_vector

6.5.3.4 p_jt

p_jt

6.5.3.5 p_w

p_w

6.5.3.6 sigma_p

sigma_p

6.5.3.7 sigma_v

sigma_v

6.5.3.8 T_wj0

T_wj0

6.5.3.9 T_wj0_inv

T_wj0_inv

6.5.3.10 T_wj_initial

T_wj_initial

6.5.3.11 total_scan_time

total_scan_time

6.5.3.12 v_j_initial

v_j_initial

6.6 PolarCoord Class Reference

Public Member Functions

- def `__init__` (self, float `r`, float `theta`)
- def `__str__` (self)
- float `getR` (self)
- float `getTheta` (self)
- None `scaleR` (self, scaleFactor)
- `CartCoord` `toCart` (self)
- tuple `asTuple` (self)

Public Attributes

- `r`
- `theta`

6.6.1 Detailed Description

Creates a point on a Cartesian coordinate plane with values `x` and `y`.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 `__init__()`

```
def __init__ (  
    self,  
    float r,  
    float theta )
```

Defines `x` and `y` variables

6.6.3 Member Function Documentation

6.6.3.1 `__str__()`

```
def __str__ (
    self )
```

6.6.3.2 `asTuple()`

```
tuple asTuple (
    self )
```

6.6.3.3 `getR()`

```
float getR (
    self )
```

6.6.3.4 `getTheta()`

```
float getTheta (
    self )
```

6.6.3.5 `scaleR()`

```
None scaleR (
    self,
    scaleFactor )
```

6.6.3.6 `toCart()`

```
CartCoord toCart (
    self )
```

6.6.4 Member Data Documentation

6.6.4.1 `r`

`r`

6.6.4.2 `theta`

`theta`

6.7 PoseGraphOptimization Class Reference

Public Member Functions

- `def __init__ (self)`
- `def optimize (self, max_iterations=20)`
- `def add_vertex (self, id, pose, fixed=False)`
- `def add_edge (self, vertices, measurement, information=np.eye(6), robust_kernel=None)`
- `def get_pose (self, id)`

6.7.1 Constructor & Destructor Documentation

6.7.1.1 `__init__()`

```
def __init__ (  
    self )
```

6.7.2 Member Function Documentation

6.7.2.1 `add_edge()`

```
def add_edge (  
    self,  
    vertices,  
    measurement,  
    information = np.eye(6),  
    robust_kernel = None )
```

6.7.2.2 add_vertex()

```
def add_vertex (
    self,
    id,
    pose,
    fixed = False )
```

6.7.2.3 get_pose()

```
def get_pose (
    self,
    id )
```

6.7.2.4 optimize()

```
def optimize (
    self,
    max_iterations = 20 )
```

6.8 RawROAMSystem Class Reference

Public Member Functions

- None `__init__` (self, str `sequenceName`, bool `paramFlags`=dict(), bool `hasGroundTruth`=True)
- def `updateTrajFromTracker` (self)
- None `run` (self, int `startSeqInd`=0, int `endSeqInd`=-1)
- None `updateTrajectory` (self, np.ndarray R, np.ndarray h, np.ndarray seqInd)
- None `updateTrajectoryAbsolute` (self, np.ndarray pose_vector, int seqInd)
- None `plot` (self, np.ndarray prevImg, np.ndarray currImg, np.ndarray good_old, np.ndarray good_new, np.ndarray R, np.ndarray h, np.ndarray seqInd, bool plotMapPoints=True, bool useArrow=False, bool save=True, bool show=False)
- None `plotTraj` (self, int seqInd, np.ndarray R, np.ndarray h, bool useArrow=False, bool save=False, bool show=False)

Public Attributes

- `sequenceName`
- `paramFlags`
- `hasGroundTruth`
- `imgPathArr`
- `sequenceSize`
- `filePaths`
- `fig`
- `gtTraj`
- `estTraj`
- `tracker`
- `map`

6.8.1 Constructor & Destructor Documentation

6.8.1.1 `__init__()`

```
None __init__ (
    self,
    str sequenceName,
    bool paramFlags = dict(),
    bool hasGroundTruth = True )

@brief Initializer for ROAM system
@param[in] sequenceName Name of sequence. Should be in ./data folder
@param[in] paramFlags Set of flags indicating turning on and off of certain algorithm features
    - rejectOutliers: Whether to use graph-based outlier rejection
    - useANMS: Whether to use ANMS
    - useFMT: Whether to use FMT to correct things
    - correctMotionDistortion: Whether to correct for motion distortion
@param[in] hasGroundTruth Whether sequence has ground truth to be used for plotting @deprecated
```

6.8.2 Member Function Documentation

6.8.2.1 `plot()`

```
None plot (
    self,
    np.ndarray prevImg,
    np.ndarray currImg,
    np.ndarray good_old,
    np.ndarray good_new,
    np.ndarray R,
    np.ndarray h,
    np.ndarray seqInd,
    bool plotMapPoints = True,
    bool useArrow = False,
    bool save = True,
    bool show = False )

@brief Perform global plotting of everything, including trajectory and map points

@param[in] prevImg (M x N) Previous Cartesian radar image to plot
@param[in] currImg (M x N) Current Cartesian radar image to plot

@param[in] good_old (K x 2) Good correspondence points from previous image in scan frame
@param[in] good_new (K x 2) Good correspondence points from current image in scan frame

@param[in] R (2 x 2) rotation matrix
@param[in] h (2 x 1) translation vector
@param[in] seqInd Sequence index

@param[in] useArrow Whether to plot with arrows/triangles to indicate pose direction.
    Otherwise uses plain lines.
@param[in] save Whether to save image as png/jpg
@param[in] show Whether to plt.show image
```

6.8.2.2 plotTraj()

```
None plotTraj (
    self,
    int seqInd,
    np.ndarray R,
    np.ndarray h,
    bool useArrow = False,
    bool save = False,
    bool show = False )

@brief Plot trajectory
@param[in] seqInd Sequence index
@param[in] R (2 x 2) rotation matrix
@param[in] h (2 x 1) translation vector
@param[in] useArrow Whether to plot with arrows/triangles to indicate pose direction.
                Otherwise uses plain lines.
@param[in] save Whether to save image as png/jpg
@param[in] show Whether to plt.show image
```

6.8.2.3 run()

```
None run (
    self,
    int startSeqInd = 0,
    int endSeqInd = -1 )

@brief Do a full run the ROAMing algorithm on sequence,
        starting from and ending at specified indices,
        incrementally calling the @see Tracker.track() function

@param[in] startImgInd Starting index of sequence. Default 0.
@param[in] endSeqInd Ending index of sequence. Default -1.
                Negative numbers to indicate end.
```

6.8.2.4 updateTrajectory()

```
None updateTrajectory (
    self,
    np.ndarray R,
    np.ndarray h,
    np.ndarray seqInd )

@brief Update trajectory using R,h matrices

@param[in] R (2 x 2) rotation matrix
@param[in] h (2 x 1) translation vector
@param[in] seqInd Sequence index, used for visualization/plotting

@note Updates internal trajectory
```


6.8.2.5 updateTrajectoryAbsolute()

```
None updateTrajectoryAbsolute (
    self,
    np.ndarray pose_vector,
    int seqInd )

@brief Update trajectory using pose vector

@param[in] pose_vector (3, ) pose vector
@param[in] seqInd Sequence index, used for visualization/plotting

@note Updates internal trajectory
```

6.8.2.6 updateTrajFromTracker()

```
def updateTrajFromTracker (
    self )
```

6.8.3 Member Data Documentation

6.8.3.1 estTraj

estTraj

6.8.3.2 fig

fig

6.8.3.3 filePaths

filePaths

6.8.3.4 gtTraj

gtTraj

6.8.3.5 hasGroundTruth

hasGroundTruth

6.8.3.6 imgPathArr

imgPathArr

6.8.3.7 map

map

6.8.3.8 paramFlags

paramFlags

6.8.3.9 sequenceName

sequenceName

6.8.3.10 sequenceSize

sequenceSize

6.8.3.11 tracker

tracker

6.9 Tracker Class Reference

Public Member Functions

- None `__init__` (self, str `sequenceName`, list[str] `imgPathArr`, dict[str] `filePaths`, dict[bool] `paramFlags`)
- def `initTraj` (self, `Trajectory` `estTraj`, `Trajectory` `gtTraj`=None)
- Tuple[np.ndarray, np.ndarray, np.ndarray] `track` (self, np.ndarray `prevImgCart`, np.ndarray `currImgCart`, np.ndarray `prevImgPolar`, np.ndarray `currImgPolar`, np.ndarray `featureCoord`, int `seqInd`)
- Tuple[np.ndarray, np.ndarray] `getTransform` (self, np.ndarray `srcCoord`, np.ndarray `targetCoord`, bool `pixel`)
- def `plot` (self, `prevImg`, `currImg`, `good_old`, `good_new`, `seqInd`, `save`=True, `show`=False)

Public Attributes

- `sequenceName`
- `imgPathArr`
- `sequenceSize`
- `filePaths`
- `paramFlags`
- `estTraj`
- `gtTraj`

6.9.1 Constructor & Destructor Documentation

6.9.1.1 `__init__()`

```
None __init__ (
    self,
    str sequenceName,
    list[str] imgPathArr,
    dict[str] filePaths,
    dict[bool] paramFlags )
```

6.9.2 Member Function Documentation

6.9.2.1 `getTransform()`

```
Tuple[np.ndarray, np.ndarray] getTransform (
    self,
    np.ndarray srcCoord,
    np.ndarray targetCoord,
    bool pixel )
```

@brief Obtain transform from coordinate correspondnces

@param[in] `srcCoord` Coordinates of src feature points ($K' \times 2$) in [x, y] format

@param[in] `targetCoord` Coordinates of target feature points ($K' \times 2$) in [x, y] format

@note $\text{target} = R @ \text{src} + h$

@return R rotation matrix (2×2)

@return h translation matrix (2×1), units in meters [m]

6.9.2.2 initTraj()

```
def initTraj (
    self,
    Trajectory estTraj,
    Trajectory gtTraj = None )
```

6.9.2.3 plot()

```
def plot (
    self,
    prevImg,
    currImg,
    good_old,
    good_new,
    seqInd,
    save = True,
    show = False )
```

6.9.2.4 track()

```
Tuple[np.ndarray, np.ndarray, np.ndarray] track (
    self,
    np.ndarray prevImgCart,
    np.ndarray currImgCart,
    np.ndarray prevImgPolar,
    np.ndarray currImgPolar,
    np.ndarray featureCoord,
    int seqInd )

@brief Track based on previous and current image

@param[in] prevImg Previous Cartesian radar image (N x N)
@param[in] prevImg Current Cartesian radar image (N x N)
@param[in] prevImg Previous polar radar image (? x ?)
@param[in] prevImg Current polar radar image (? x ?)

@param[in] blobCoord Coordinates of feature points (K x 2) in [x, y] format

@return good_old Coordinates of old good feature points (K' x 2) in [x, y] format
@return good_new Coordinates of new good feature points (K' x 2) in [x, y] format
@return angleRotRad Angle used to rotate image
@return corrStatus (K x 2) correspondence status @note Needed for mapping to track keyframe points
```

6.9.3 Member Data Documentation

6.9.3.1 estTraj

estTraj

6.9.3.2 filePaths

filePaths

6.9.3.3 gtTraj

gtTraj

6.9.3.4 imgPathArr

imgPathArr

6.9.3.5 paramFlags

paramFlags

6.9.3.6 sequenceName

sequenceName

6.9.3.7 sequenceSize

sequenceSize

6.10 Trajectory Class Reference

Public Member Functions

- `def __init__ (self, timestamps, poses)`
- `def getGroundTruthDeltasAtTime (self, time)`
- `def appendRelativeDeltas (self, time, d_xyth)`
- `def appendRelativeTransform (self, time, R, h)`
- `def appendAbsoluteTransform (self, time, pose)`
- `def getPoseAtTimes (self, times)`
- `def plot (self, title='My Trajectory', savePath=False)`

Public Attributes

- `timestamps`
- `poses`
- `pose_transform`

6.10.1 Constructor & Destructor Documentation

6.10.1.1 __init__()

```
def __init__ (
    self,
    timestamps,
    poses )

@param[in] timestamps np.ndarray of sorted timestamps (N)
@param[in] pose_matrices np.ndarray of poses (N x 3)
```

6.10.2 Member Function Documentation

6.10.2.1 appendAbsoluteTransform()

```
def appendAbsoluteTransform (
    self,
    time,
    pose )

@brief Append a relative transform to the trajectory
    h should already be scaled by radar resolution
@param[in] time timestamp of the transform
@param[in] pose pose vector (3, )
```

6.10.2.2 `appendRelativeDeltas()`

```
def appendRelativeDeltas (
    self,
    time,
    d_xyth )
```

6.10.2.3 `appendRelativeTransform()`

```
def appendRelativeTransform (
    self,
    time,
    R,
    h )

@brief Append a relative transform to the trajectory
    h should already be scaled by radar resolution
@param[in] time timestamp of the transform
@param[in] R rotation matrix (2 x 2)
@param[in] h translation vector (2 x 1)
```

6.10.2.4 `getGroundTruthDeltasAtTime()`

```
def getGroundTruthDeltasAtTime (
    self,
    time )
```

@brief Given a timestamp, return the ground truth deltas at that time in (dx, dy, dth) list for debugging

6.10.2.5 `getPoseAtTimes()`

```
def getPoseAtTimes (
    self,
    times )
```

@brief Given timestamps, return the pose at that time using cubic interpolation
@param[in] times np.ndarray of sorted timestamps

6.10.2.6 plot()

```
def plot (
    self,
    title = 'My Trajectory',
    savePath = False )
```

6.10.3 Member Data Documentation

6.10.3.1 pose_transform

pose_transform

6.10.3.2 poses

poses

6.10.3.3 timestamps

timestamps

Index

- `__init__`
 - BundleAdjustment, 67
 - CartCoord, 69
 - Keyframe, 72
 - Map, 75
 - MotionDistortionSolver, 78
 - PolarCoord, 83
 - PoseGraphOptimization, 85
 - RawROAMSystem, 87
 - Tracker, 91
 - Trajectory, 94
 - `__str__`
 - CartCoord, 69
 - PolarCoord, 84
- A
 - testMotionDistortion, 50
 - testTransform, 56
- A_inv
 - testMotionDistortion, 50
 - testTransform, 56
- adaptiveNMS
 - getFeatures, 19
- add
 - CartCoord, 70
- add_edge
 - BundleAdjustment, 67
 - PoseGraphOptimization, 85
- add_point
 - BundleAdjustment, 67
- add_pose
 - BundleAdjustment, 68
- add_vertex
 - PoseGraphOptimization, 85
- addCoord
 - CartCoord, 70
- addKeyframe
 - Map, 76
- addNoise
 - genFakeData, 15
- alpha
 - getTransformKLT, 30
 - outlierRejection, 39
 - testMotionDistortion, 50
 - testTransform, 56
- ANMS, 9
 - ssc, 9
- appendAbsoluteTransform
 - Trajectory, 94
- appendNewFeatures
 - getFeatures, 19
- appendRelativeDeltas
 - Trajectory, 94
- appendRelativeTransform
 - Trajectory, 95
- asTuple
 - CartCoord, 70
 - PolarCoord, 84
- bad_new
 - getTransformKLT, 30
- bad_old
 - getTransformKLT, 30
- blobCoord
 - getTransformKLT, 30
- blobIndices
 - getFeatures, 20
- blobSigma
 - getFeatures, 20
- blobX
 - getFeatures, 20
- blobY
 - getFeatures, 20
- block
 - trajectoryPlotting, 60
- BundleAdjustment, 67
 - `__init__`, 67
 - add_edge, 67
 - add_point, 67
 - add_pose, 68
 - get_point, 68
 - get_pose, 68
 - optimize, 68
- C
 - getPointCloud, 24
 - parseData, 45
- calculateFeatureLossThreshold
 - getFeatures, 19
- calculateTransform
 - getTransformKLT, 27
- calculateTransformDth
 - getTransformKLT, 28
- calculateTransformDxDth
 - getTransformKLT, 28
- calculateTransformSVD
 - getTransformKLT, 28
- CartCoord, 69
 - `__init__`, 69
 - `__str__`, 69

- add, 70
- addCoord, 70
- asTuple, 70
- getAngle, 70
- getDistance, 70
- getX, 70
- getY, 71
- scale, 71
- scaleX, 71
- scaleY, 71
- sub, 71
- x, 71
- y, 72
- clear
 - testMotionDistortion, 50
 - testTransform, 56
- color
 - getFeatures, 20
- compute_time_deltas
 - MotionDistortionSolver, 79
- computePosesRMSE
 - trajectoryPlotting, 59
- convertCartesianImageToPolar
 - parseData, 42
- convertFeaturesLocalToGlobal
 - Keyframe, 73
- convertPolarImageToCartesian
 - parseData, 42
- convertPolarImgToLogPolar
 - parseData, 43
- convertPolarPointsToCartesian
 - genFakeData, 15
- convertPoseToTransform
 - utils, 62
- convertRandHtoDeltas
 - utils, 62
- convertTransformToPose
 - utils, 62
- Coord, 9
- coord
 - getFeatures, 21
- copyFromOtherKeyframe
 - Keyframe, 73
- corrStatus
 - getTransformKLT, 30
- cov_p
 - testMotionDistortion, 50
- cov_v
 - testMotionDistortion, 50
- createOutliers
 - genFakeData, 15
- currentFrame
 - testMotionDistortion, 50
- currImg
 - getTransformKLT, 30
- currImgCart
 - FMT, 12
- currImgPolar
 - FMT, 12
- currTimestamp
 - getTransformKLT, 30
- dataPath
 - FMT, 12
 - getFeatures, 21
 - getPointCloud, 24
 - getTransformKLT, 30
 - parseData, 45
- datasetName
 - getFeatures, 21
 - getPointCloud, 24
 - getTransformKLT, 31
 - parseData, 45
 - RawROAMSystem, 47
 - trajectoryPlotting, 60
- debug
 - MotionDistortionSolver, 81
- DEFAULT_FEATURE_PARAMS
 - getFeatures, 21
- DIST_THRESHOLD_M
 - outlierRejection, 39
- DIST_THRESHOLD_PX
 - outlierRejection, 39
- distort
 - genFakeData, 16
- distorted
 - testMotionDistortion, 51
- DISTSQ_THRESHOLD_PX
 - outlierRejection, 39
- DOWNSAMPLE_FACTOR
 - parseData, 45
- drawCVPoint
 - parseData, 43
- dT
 - MotionDistortionSolver, 81
- dth
 - getTransformKLT, 31
- dx
 - getTransformKLT, 31
- end
 - getFeatures, 21
- endSeqInd
 - FMT, 12
 - RawROAMSystem, 47
- ERR_THRESHOLD
 - getTransformKLT, 31
- error
 - MotionDistortionSolver, 79
- error_vector
 - MotionDistortionSolver, 79
- est_deltas
 - getTransformKLT, 31
- estimateTransformUsingDelats
 - getTransformKLT, 28
- estPoses
 - trajectoryPlotting, 61

- estTraj
 - getTransformKLT, 31
 - Map, 77
 - RawROAMSystem, 89
 - Tracker, 92
 - trajectoryPlotting, 61
- exist_ok
 - FMT, 12
 - getFeatures, 21
 - getTransformKLT, 31
- expected_observed_pts
 - MotionDistortionSolver, 79
- extractDataFromRadarImage
 - parseData, 43
- extraLabel
 - getTransformKLT, 31
- f_arr
 - utils, 63
- False
 - testMotionDistortion, 51
 - testTransform, 56
- featureAzim
 - getPointCloud, 25
- featureImgCart
 - getPointCloud, 25
- featurePointsLocal
 - Keyframe, 74
- featurePointsLocalUndistorted
 - Keyframe, 74
- featurePolarImage
 - getPointCloud, 25
- featurePolarIndices
 - getPointCloud, 25
- featureRange
 - getPointCloud, 25
- fig
 - RawROAMSystem, 89
- filePaths
 - Map, 77
 - RawROAMSystem, 89
 - Tracker, 93
- final_undistorted
 - testMotionDistortion, 51
- flatten
 - utils, 63
- flush
 - getFeatures, 21
- FMT, 10
 - currImgCart, 12
 - currImgPolar, 12
 - dataPath, 12
 - endSeqInd, 12
 - exist_ok, 12
 - FMT_DOWNSAMPLE_FACTOR, 12
 - FMT_RANGE_CLIP_M, 12
 - getRotationUsingFMT, 10
 - getTranslationUsingPhaseCorrelation, 11
 - imgPathArr, 13
 - imgSavePath, 13
 - imgSavePathInd, 13
 - plotCartPolar, 11
 - plotCartPolarWithRotation, 11
 - prevImgCart, 13
 - prevImgPolar, 13
 - REMOVE_OLD_RESULTS, 13
 - response, 13
 - rotateImg, 11
 - rotRad, 14
 - scale, 14
 - sequenceName, 14
 - sequenceSize, 14
 - startSeqInd, 14
 - stepSize, 14
 - timestampPath, 14
- FMT_DOWNSAMPLE_FACTOR
 - FMT, 12
- FMT_RANGE_CLIP_M
 - FMT, 12
- FORCE_OUTLIERS
 - outlierRejection, 39
- frequency
 - testMotionDistortion, 51
- generateFakeCorrespondences
 - genFakeData, 16
- generateFakeCorrespondencesPolar
 - genFakeData, 16
- generateFakeFeatures
 - genFakeData, 17
- generateFakeFeaturesPolar
 - genFakeData, 17
- generateTranslationVector
 - genFakeData, 17
- genFakeData, 15
 - addNoise, 15
 - convertPolarPointsToCartesian, 15
 - createOutliers, 15
 - distort, 16
 - generateFakeCorrespondences, 16
 - generateFakeCorrespondencesPolar, 16
 - generateFakeFeatures, 17
 - generateFakeFeaturesPolar, 17
 - generateTranslationVector, 17
 - plotFakeFeatures, 17
 - transformCoords, 17
- get_point
 - BundleAdjustment, 68
- get_pose
 - BundleAdjustment, 68
 - PoseGraphOptimization, 86
- getAngle
 - CartCoord, 70
- getBlobsFromCart
 - getFeatures, 19
- getCartImageFromImgPaths
 - parseData, 43
- getDataFromImgPathsByIndex

- parseData, 44
- getDistance
 - CartCoord, 70
- getFeatures, 18
 - adaptiveNMS, 19
 - appendNewFeatures, 19
 - blobIndices, 20
 - blobSigma, 20
 - blobX, 20
 - blobY, 20
 - calculateFeatureLossThreshold, 19
 - color, 20
 - coord, 21
 - dataPath, 21
 - datasetName, 21
 - DEFAULT_FEATURE_PARAMS, 21
 - end, 21
 - exist_ok, 21
 - flush, 21
 - getBlobsFromCart, 19
 - getFeatures, 19
 - imgCart, 22
 - imgCartBGR, 22
 - imgPolar, 22
 - N_FEATURES_BEFORE_RETRACK, 22
 - nImgs, 22
 - nIndices, 22
 - nIndicesANMS, 22
 - PERCENT_FEATURE_LOSS_THRESHOLD, 22
 - s_blobIndices, 23
 - streamArr, 23
 - timestampPath, 23
 - toSaveImgPath, 23
 - toSavePath, 23
- getGroundTruthDeltasAtTime
 - Trajectory, 95
- getGroundTruthTrajectory
 - trajectoryPlotting, 59
- getGroundTruthTrajectoryGPS
 - trajectoryPlotting, 60
- getPointCloud, 23
 - c, 24
 - dataPath, 24
 - datasetName, 24
 - featureAzim, 25
 - featureImgCart, 25
 - featurePolarImage, 25
 - featurePolarIndices, 25
 - featureRange, 25
 - getPointCloudPolarInd, 24
 - imgCart, 25
 - imgCartRGB, 25
 - imgPolar, 25
 - nImgs, 26
 - streamArr, 26
 - timestampPath, 26
- getPointCloudPolarInd
 - getPointCloud, 24
- getPolarImageFromImgPaths
 - parseData, 44
- getPoseAtTimes
 - Trajectory, 95
- getPrunedFeaturesGlobalPosition
 - Keyframe, 73
- getR
 - PolarCoord, 84
- getRadarImgPaths
 - parseData, 44
- getRadarStreamPolar
 - parseData, 45
- getRotationMatrix
 - utils, 63
- getRotationUsingFMT
 - FMT, 10
- getTheta
 - PolarCoord, 84
- getTrackedPointsKLT
 - getTransformKLT, 29
- getTransform
 - Tracker, 91
- getTransformKLT, 26
 - alpha, 30
 - bad_new, 30
 - bad_old, 30
 - blobCoord, 30
 - calculateTransform, 27
 - calculateTransformDth, 28
 - calculateTransformDxDth, 28
 - calculateTransformSVD, 28
 - corrStatus, 30
 - currImg, 30
 - currTimestamp, 30
 - dataPath, 30
 - datasetName, 31
 - dth, 31
 - dx, 31
 - ERR_THRESHOLD, 31
 - est_deltas, 31
 - estimateTransformUsingDelats, 28
 - estTraj, 31
 - exist_ok, 31
 - extraLabel, 31
 - getTrackedPointsKLT, 29
 - good_new, 32
 - good_old, 32
 - gt_deltas, 32
 - gtTraj, 32
 - gtTrajPath, 32
 - h, 32
 - imgPathArr, 32
 - imgSavePath, 32
 - initPose, 33
 - initTimestamp, 33
 - LK_PARAMS, 33
 - N_FEATURES_BEFORE_RETRACK, 33
 - nBadFeatures, 33

- nFeatures, 33
- nGoodFeatures, 34
- nImgs, 34
- PLOT_BAD_FEATURES, 34
- prev_good_old, 34
- prevImg, 34
- R, 34
- REMOVE_OLD_RESULTS, 34
- saveFeaturePath, 34
- savePath, 35
- show, 35
- start, 35
- startImgInd, 35
- timestamp, 35
- timestampPath, 35
- toSaveImgPath, 35
- toSaveTrajPath, 36
- trajSavePath, 36
- transformed_pts, 36
- visualize_transform, 29
- getTranslationUsingPhaseCorrelation
 - FMT, 11
- getX
 - CartCoord, 70
- getY
 - CartCoord, 71
- good_new
 - getTransformKLT, 32
- good_old
 - getTransformKLT, 32
- groundTruth
 - testMotionDistortion, 51
- gt_deltas
 - getTransformKLT, 32
- gtPath
 - trajectoryPlotting, 61
- gtTraj
 - getTransformKLT, 32
 - RawROAMSystem, 89
 - Tracker, 93
 - trajectoryPlotting, 61
- gtTrajPath
 - getTransformKLT, 32
- h
 - getTransformKLT, 32
 - testMotionDistortion, 51
 - testTransform, 56
- h_fit
 - testMotionDistortion, 51
 - testTransform, 56
- hasGroundTruth
 - RawROAMSystem, 89
- homogenize
 - utils, 63
- imgCart
 - getFeatures, 22
 - getPointCloud, 25
- imgCartBGR
 - getFeatures, 22
- imgCartRGB
 - getPointCloud, 25
- imgPathArr
 - FMT, 13
 - getTransformKLT, 32
 - Map, 77
 - RawROAMSystem, 90
 - Tracker, 93
- imgPolar
 - getFeatures, 22
 - getPointCloud, 25
 - parseData, 46
- imgSavePath
 - FMT, 13
 - getTransformKLT, 32
 - RawROAMSystem, 47
- imgSavePathInd
 - FMT, 13
- infer_velocity
 - MotionDistortionSolver, 80
- info_vector
 - MotionDistortionSolver, 81
- initPose
 - getTransformKLT, 33
- initTimestamp
 - getTransformKLT, 33
- initTraj
 - Tracker, 91
- invert_transform
 - utils, 63
- isGoodKeyframe
 - Map, 76
- jacobian
 - MotionDistortionSolver, 80
- jacobian_vector
 - MotionDistortionSolver, 80
- Keyframe, 72
 - __init__, 72
 - convertFeaturesLocalToGlobal, 73
 - copyFromOtherKeyframe, 73
 - featurePointsLocal, 74
 - featurePointsLocalUndistorted, 74
 - getPrunedFeaturesGlobalPosition, 73
 - pointCloud, 74
 - pose, 74
 - prunedFeaturePoints, 74
 - prunedUndistortedLocals, 74
 - pruneFeaturePoints, 73
 - radarPolarImg, 75
 - updateInfo, 73
 - velocity, 75
- keyframe_timestamps
 - trajectoryPlotting, 61
- keyframes

- Map, 77
- LK_PARAMS
 - getTransformKLT, 33
- Map, 75
 - __init__, 75
 - addKeyframe, 76
 - estTraj, 77
 - filePaths, 77
 - imgPathArr, 77
 - isGoodKeyframe, 76
 - keyframes, 77
 - mapPoints, 77
 - plot, 76
 - sequenceName, 77
 - sequenceSize, 77
 - updateInternalTraj, 76
- map
 - RawROAMSystem, 90
- Mapping, 36
 - RADAR_CART_CENTER, 36
 - ROT_THRESHOLD, 37
 - TRANS_THRESHOLD, 37
 - TRANS_THRESHOLD_SQ, 37
- mapPoints
 - Map, 77
- MAX_RANGE_CLIP_DEFAULT
 - parseData, 46
- max_translation_m
 - outlierRejection, 40
- MDS
 - testMotionDistortion, 51
- motionDistortion, 37
 - RADAR_SCAN_FREQUENCY, 37
 - VERBOSE, 37
- MotionDistortionSolver, 78
 - __init__, 78
 - compute_time_deltas, 79
 - debug, 81
 - dT, 81
 - error, 79
 - error_vector, 79
 - expected_observed_pts, 79
 - infer_velocity, 80
 - info_vector, 81
 - jacobian, 80
 - jacobian_vector, 80
 - optimize, 80
 - optimize_library, 80
 - p_jt, 82
 - p_w, 82
 - sigma_p, 82
 - sigma_v, 82
 - T_wj0, 82
 - T_wj0_inv, 82
 - T_wj_initial, 82
 - total_scan_time, 82
 - undistort, 81
 - update_problem, 81
 - v_j_initial, 83
- N
 - testMotionDistortion, 52
 - testTransform, 57
- N_FEATURES_BEFORE_RETRACK
 - getFeatures, 22
 - getTransformKLT, 33
- n_outliers
 - outlierRejection, 40
- n_points
 - outlierRejection, 40
 - testMotionDistortion, 52
 - testTransform, 57
- nBadFeatures
 - getTransformKLT, 33
- new_coord
 - outlierRejection, 40
- new_coord_perfect
 - outlierRejection, 40
- nFeatures
 - getTransformKLT, 33
- nGoodFeatures
 - getTransformKLT, 34
- nImgs
 - getFeatures, 22
 - getPointCloud, 26
 - getTransformKLT, 34
 - parseData, 46
- nIndices
 - getFeatures, 22
- nIndicesANMS
 - getFeatures, 22
- noise
 - trajectoryPlotting, 61
- noiseToAdd
 - outlierRejection, 40
 - testMotionDistortion, 52
 - testTransform, 57
- noisy
 - testMotionDistortion, 52
 - testTransform, 57
- normalize_angles
 - utils, 63
- optimize
 - BundleAdjustment, 68
 - MotionDistortionSolver, 80
 - PoseGraphOptimization, 86
- optimize_library
 - MotionDistortionSolver, 80
- outlier_ind
 - outlierRejection, 40
 - testMotionDistortion, 52
 - testTransform, 57
- outlier_rate
 - testMotionDistortion, 52
 - testTransform, 57

- outlierRejection, 38
 - alpha, 39
 - DIST_THRESHOLD_M, 39
 - DIST_THRESHOLD_PX, 39
 - DISTSQ_THRESHOLD_PX, 39
 - FORCE_OUTLIERS, 39
 - max_translation_m, 40
 - n_outliers, 40
 - n_points, 40
 - new_coord, 40
 - new_coord_perfect, 40
 - noiseToAdd, 40
 - outlier_ind, 40
 - prev_coord, 40
 - pruned_new_coord, 41
 - pruned_prev_coord, 41
 - rejectOutliers, 38
 - show, 41
 - theta_deg, 41
 - theta_max_deg, 41
 - title_append, 41
 - trans_vec, 41
- p_it
 - MotionDistortionSolver, 82
 - testMotionDistortion, 52
- p_w
 - MotionDistortionSolver, 82
 - testMotionDistortion, 52
- paramFlags
 - RawROAMSystem, 48, 90
 - Tracker, 93
- params
 - testMotionDistortion, 53
- parseData, 42
 - c, 45
 - convertCartesianImageToPolar, 42
 - convertPolarImageToCartesian, 42
 - convertPolarImgToLogPolar, 43
 - dataPath, 45
 - datasetName, 45
 - DOWNSAMPLE_FACTOR, 45
 - drawCVPoint, 43
 - extractDataFromRadarImage, 43
 - getCartImageFromImgPaths, 43
 - getDataFromImgPathsByIndex, 44
 - getPolarImageFromImgPaths, 44
 - getRadarImgPaths, 44
 - getRadarStreamPolar, 45
 - imgCart, 46
 - imgPolar, 46
 - MAX_RANGE_CLIP_DEFAULT, 46
 - nImgs, 46
 - RANGE_RESOLUTION_CART_M, 46
 - RANGE_RESOLUTION_M, 46
 - streamArr, 46
 - timestampPath, 46
- PERCENT_FEATURE_LOSS_THRESHOLD
 - getFeatures, 22
- period
 - testMotionDistortion, 53
- plot
 - Map, 76
 - RawROAMSystem, 87
 - Tracker, 92
 - Trajectory, 95
- PLOT_BAD_FEATURES
 - getTransformKLT, 34
- plotCartPolar
 - FMT, 11
- plotCartPolarWithRotation
 - FMT, 11
- plotDisplace
 - testMotionDistortion, 53
- plotFakeFeatures
 - genFakeData, 17
- plotGtAndEstTrajectory
 - trajectoryPlotting, 60
- plotTraj
 - RawROAMSystem, 87
- plt_full_extent
 - utils, 64
- plt_savefig_by_axis
 - utils, 64
- pointCloud
 - Keyframe, 74
- PolarCoord, 83
 - __init__, 83
 - __str__, 84
 - asTuple, 84
 - getR, 84
 - getTheta, 84
 - r, 85
 - scaleR, 84
 - theta, 85
 - toCart, 84
- pose
 - Keyframe, 74
- pose_transform
 - Trajectory, 96
- PoseGraphLib, 47
- PoseGraphOptimization, 85
 - __init__, 85
 - add_edge, 85
 - add_vertex, 85
 - get_pose, 86
 - optimize, 86
- poses
 - Trajectory, 96
- prev_coord
 - outlierRejection, 40
- prev_good_old
 - getTransformKLT, 34
- prevImg
 - getTransformKLT, 34
- prevImgCart
 - FMT, 13

- prevImgPolar
 - FMT, 13
- pruned_new_coord
 - outlierRejection, 41
- pruned_prev_coord
 - outlierRejection, 41
- prunedFeaturePoints
 - Keyframe, 74
- prunedUndistortedLocals
 - Keyframe, 74
- pruneFeaturePoints
 - Keyframe, 73
- quiver
 - utils, 64
- R
 - getTransformKLT, 34
- r
 - PolarCoord, 85
- R_fit
 - testMotionDistortion, 53
 - testTransform, 57
- RADAR_CART_CENTER
 - Mapping, 36
 - RawROAMSystem, 48
- RADAR_SCAN_FREQUENCY
 - motionDistortion, 37
- radarImgPathToTimestamp
 - utils, 64
- radarPolarImg
 - Keyframe, 75
- RANGE_RESOLUTION_CART_M
 - parseData, 46
- RANGE_RESOLUTION_M
 - parseData, 46
- RawROAMSystem, 47, 86
 - __init__, 87
 - datasetName, 47
 - endSeqInd, 47
 - estTraj, 89
 - fig, 89
 - filePaths, 89
 - gtTraj, 89
 - hasGroundTruth, 89
 - imgPathArr, 90
 - imgSavePath, 47
 - map, 90
 - paramFlags, 48, 90
 - plot, 87
 - plotTraj, 87
 - RADAR_CART_CENTER, 48
 - REMOVE_OLD_RESULTS, 48
 - run, 88
 - sequenceName, 90
 - sequenceSize, 90
 - startSeqInd, 48
 - system, 48
 - tracker, 90
 - trajSavePath, 48
 - updateTrajectory, 88
 - updateTrajectoryAbsolute, 88
 - updateTrajFromTracker, 89
 - wantToPlot, 49
- rejectOutliers
 - outlierRejection, 38
- REMOVE_OLD_RESULTS
 - FMT, 13
 - getTransformKLT, 34
 - RawROAMSystem, 48
- response
 - FMT, 13
- ROT_THRESHOLD
 - Mapping, 37
- rotatImg
 - FMT, 11
- rotRad
 - FMT, 14
- run
 - RawROAMSystem, 88
- s_blobIndices
 - getFeatures, 23
- saveFeaturePath
 - getTransformKLT, 34
- savePath
 - getTransformKLT, 35
- scale
 - CartCoord, 71
 - FMT, 14
- scaleR
 - PolarCoord, 84
- scaleX
 - CartCoord, 71
- scaleY
 - CartCoord, 71
- sequenceName
 - FMT, 14
 - Map, 77
 - RawROAMSystem, 90
 - Tracker, 93
- sequenceSize
 - FMT, 14
 - Map, 77
 - RawROAMSystem, 90
 - Tracker, 93
- show
 - getTransformKLT, 35
 - outlierRejection, 41
 - testMotionDistortion, 53
 - testTransform, 57
- sigma_p
 - MotionDistortionSolver, 82
- sigma_v
 - MotionDistortionSolver, 82
- solution
 - testMotionDistortion, 53
- srcCoord

- testTransform, 58
- srcCoord2
 - testMotionDistortion, 53
 - testTransform, 58
- srcCoord3
 - testMotionDistortion, 53
- ssc
 - ANMS, 9
- start
 - getTransformKLT, 35
- startImgInd
 - getTransformKLT, 35
- startSeqInd
 - FMT, 14
 - RawROAMSystem, 48
- stepSize
 - FMT, 14
- streamArr
 - getFeatures, 23
 - getPointCloud, 26
 - parseData, 46
- sub
 - CartCoord, 71
- system
 - RawROAMSystem, 48
- T_wj
 - testMotionDistortion, 54
- T_wj0
 - MotionDistortionSolver, 82
 - testMotionDistortion, 54
- T_wj0_inv
 - MotionDistortionSolver, 82
- T_wj_initial
 - MotionDistortionSolver, 82
- targetCoord
 - testTransform, 58
- testMotionDistortion, 49
 - A, 50
 - A_inv, 50
 - alpha, 50
 - clear, 50
 - cov_p, 50
 - cov_v, 50
 - currentFrame, 50
 - distorted, 51
 - False, 51
 - final_undistorted, 51
 - frequency, 51
 - groundTruth, 51
 - h, 51
 - h_fit, 51
 - MDS, 51
 - N, 52
 - n_points, 52
 - noiseToAdd, 52
 - noisy, 52
 - outlier_ind, 52
 - outlier_rate, 52
 - p_jt, 52
 - p_w, 52
 - params, 53
 - period, 53
 - plotDisplace, 53
 - R_fit, 53
 - show, 53
 - solution, 53
 - srcCoord2, 53
 - srcCoord3, 53
 - T_wj, 54
 - T_wj0, 54
 - theta_deg, 54
 - theta_fit, 54
 - title_append, 54
 - transform, 54
 - undistorted, 54
 - useOld, 55
 - v_j0, 55
 - velocity, 55
- testTransform, 55
 - A, 56
 - A_inv, 56
 - alpha, 56
 - clear, 56
 - False, 56
 - h, 56
 - h_fit, 56
 - N, 57
 - n_points, 57
 - noiseToAdd, 57
 - noisy, 57
 - outlier_ind, 57
 - outlier_rate, 57
 - R_fit, 57
 - show, 57
 - srcCoord, 58
 - srcCoord2, 58
 - targetCoord, 58
 - theta_deg, 58
 - theta_fit, 58
 - title_append, 58
 - useOld, 58
- theta
 - PolarCoord, 85
- theta_deg
 - outlierRejection, 41
 - testMotionDistortion, 54
 - testTransform, 58
- theta_fit
 - testMotionDistortion, 54
 - testTransform, 58
- theta_max_deg
 - outlierRejection, 41
- tic
 - utils, 64
- timestamp
 - getTransformKLT, 35

- timestampPath
 - FMT, 14
 - getFeatures, 23
 - getPointCloud, 26
 - getTransformKLT, 35
 - parseData, 46
 - trajectoryPlotting, 61
- timestamps
 - Trajectory, 96
- title_append
 - outlierRejection, 41
 - testMotionDistortion, 54
 - testTransform, 58
- toc
 - utils, 65
- toCart
 - PolarCoord, 84
- toSaveImgPath
 - getFeatures, 23
 - getTransformKLT, 35
- toSavePath
 - getFeatures, 23
- toSaveTrajPath
 - getTransformKLT, 36
- total_scan_time
 - MotionDistortionSolver, 82
- track
 - Tracker, 92
- Tracker, 59, 91
 - __init__, 91
 - estTraj, 92
 - filePaths, 93
 - getTransform, 91
 - gtTraj, 93
 - imgPathArr, 93
 - initTraj, 91
 - paramFlags, 93
 - plot, 92
 - sequenceName, 93
 - sequenceSize, 93
 - track, 92
- tracker
 - RawROAMSystem, 90
- Trajectory, 94
 - __init__, 94
 - appendAbsoluteTransform, 94
 - appendRelativeDeltas, 94
 - appendRelativeTransform, 95
 - getGroundTruthDeltasAtTime, 95
 - getPoseAtTimes, 95
 - plot, 95
 - pose_transform, 96
 - poses, 96
 - timestamps, 96
- trajectoryPlotting, 59
 - block, 60
 - computePosesRMSE, 59
 - datasetName, 60
 - estPoses, 61
 - estTraj, 61
 - getGroundTruthTrajectory, 59
 - getGroundTruthTrajectoryGPS, 60
 - gtPath, 61
 - gtTraj, 61
 - keyframe_timestamps, 61
 - noise, 61
 - plotGtAndEstTrajectory, 60
 - timestampPath, 61
- trajSavePath
 - getTransformKLT, 36
 - RawROAMSystem, 48
- TRANS_THRESHOLD
 - Mapping, 37
- TRANS_THRESHOLD_SQ
 - Mapping, 37
- trans_vec
 - outlierRejection, 41
- transform
 - testMotionDistortion, 54
- transformCoords
 - genFakeData, 17
- transformed_pts
 - getTransformKLT, 36
- undistort
 - MotionDistortionSolver, 81
- undistorted
 - testMotionDistortion, 54
- update_problem
 - MotionDistortionSolver, 81
- updateInfo
 - Keyframe, 73
- updateInternalTraj
 - Map, 76
- updateTrajectory
 - RawROAMSystem, 88
- updateTrajectoryAbsolute
 - RawROAMSystem, 88
- updateTrajFromTracker
 - RawROAMSystem, 89
- useOld
 - testMotionDistortion, 55
 - testTransform, 58
- utils, 62
 - convertPoseToTransform, 62
 - convertRandHtoDeltas, 62
 - convertTransformToPose, 62
 - f_arr, 63
 - flatten, 63
 - getRotationMatrix, 63
 - homogenize, 63
 - invert_transform, 63
 - normalize_angles, 63
 - plt_full_extent, 64
 - plt_savefig_by_axis, 64
 - quiver, 64
 - radarImgPathToTimestamp, 64

- tic, [64](#)
- toc, [65](#)
- v_j0
 - testMotionDistortion, [55](#)
- v_j_initial
 - MotionDistortionSolver, [83](#)
- velocity
 - Keyframe, [75](#)
 - testMotionDistortion, [55](#)
- VERBOSE
 - motionDistortion, [37](#)
- visualize_transform
 - getTransformKLT, [29](#)
- wantToPlot
 - RawROAMSystem, [49](#)
- x
 - CartCoord, [71](#)
- y
 - CartCoord, [72](#)