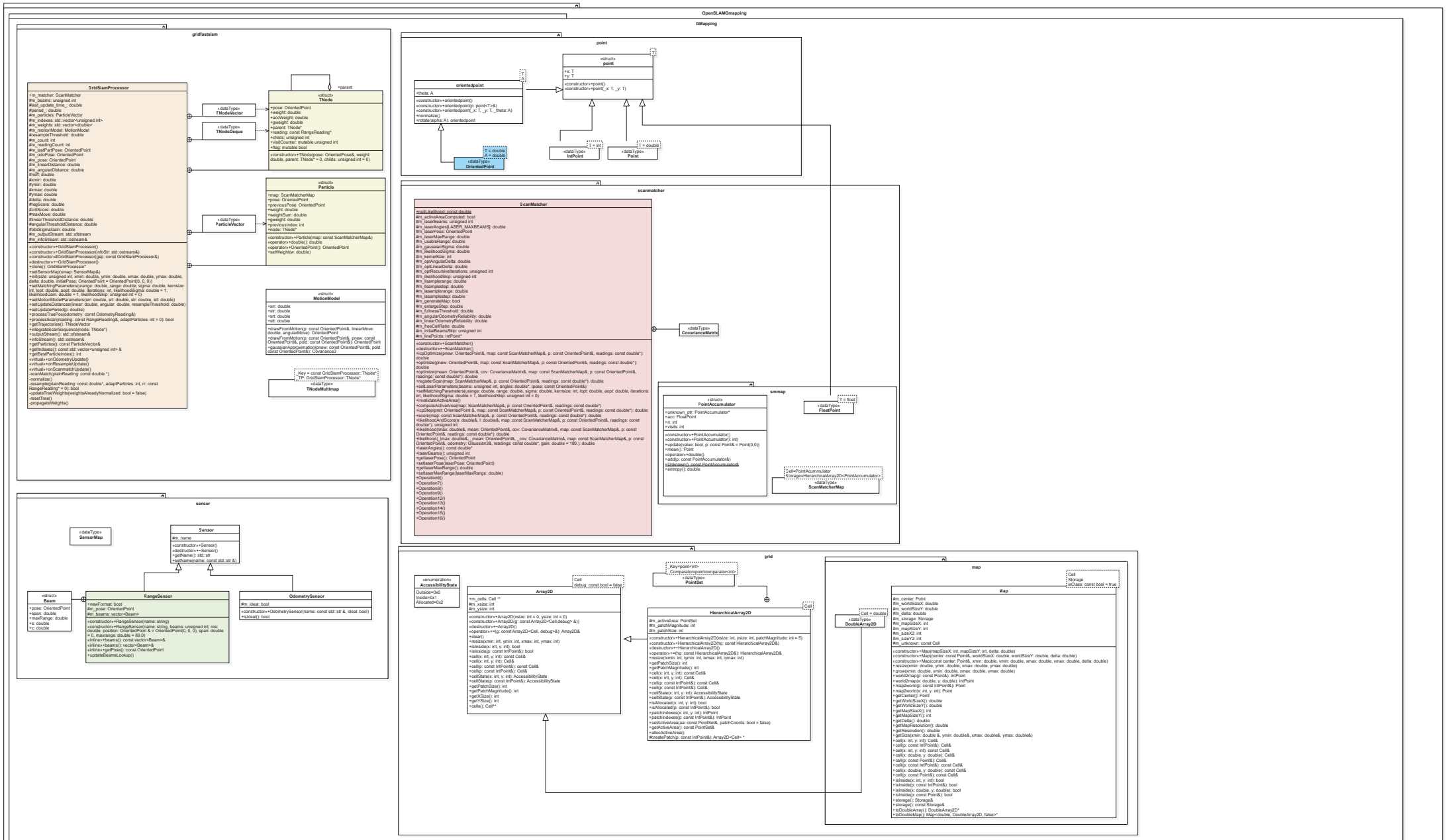
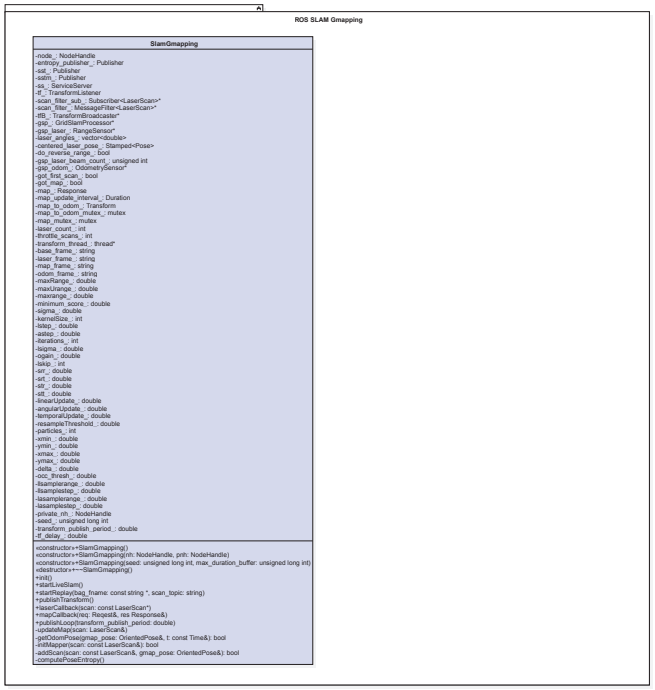
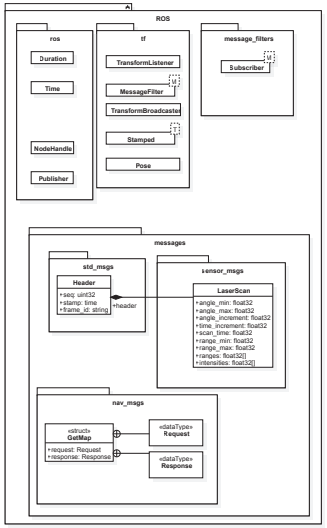
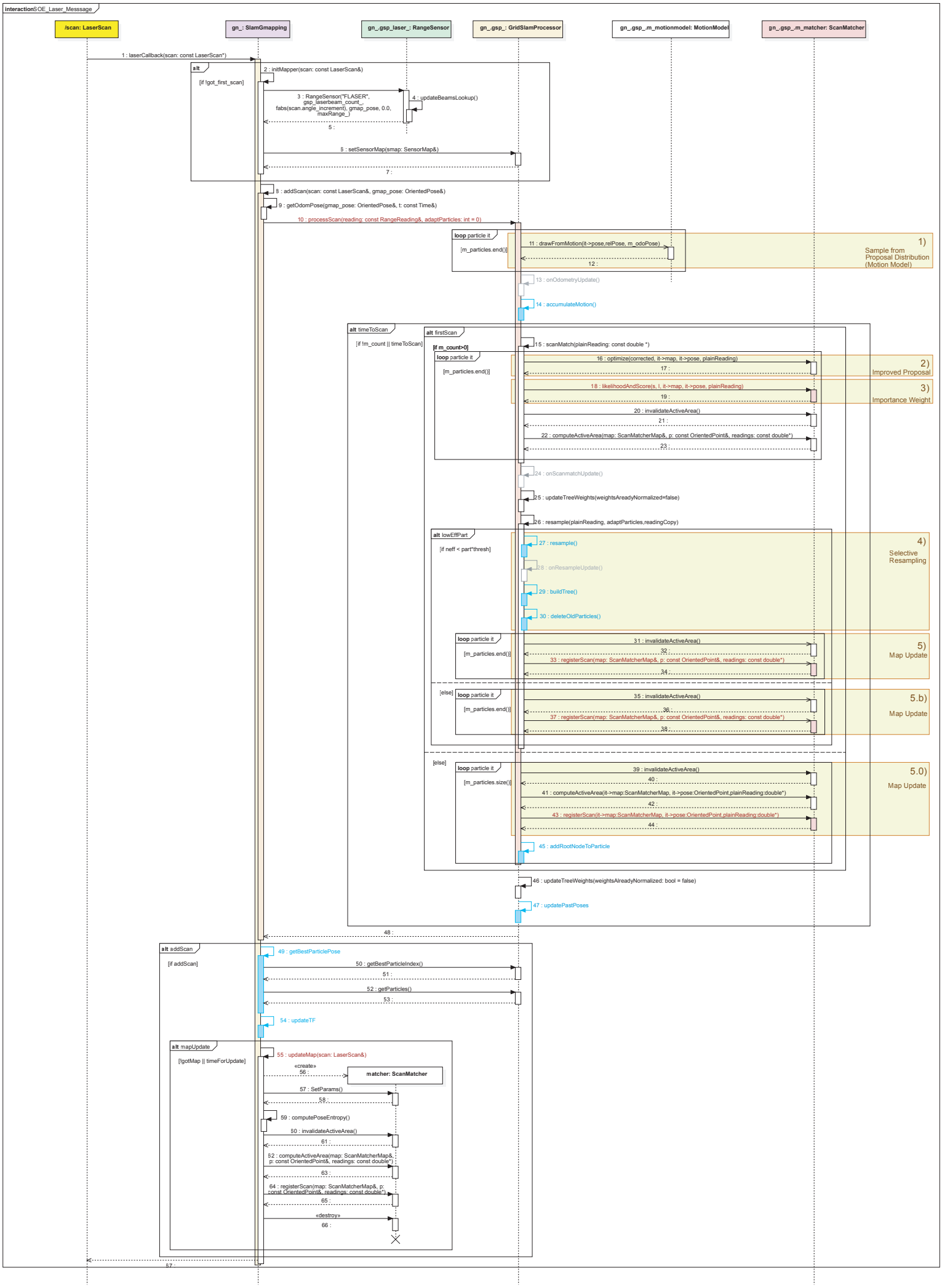
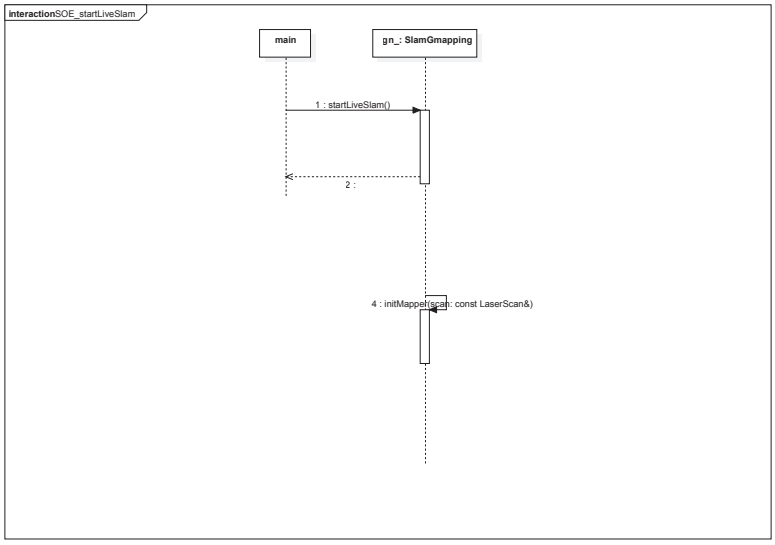
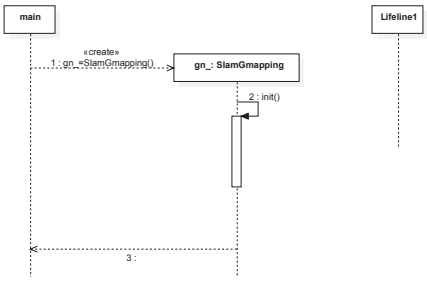

OpenSLAMGmapping
GMapping
gridfastslam
GridSlamProcessor
+m_matcher: ScanMatcher
#m_beams: unsigned int
#last_update_time_: double
#period_: double
#m_particles: ParticleVector
#m_indexes: std::vector<unsigned int>
#m_weights: std::vector<double>
#m_motionModel: MotionModel
#resampleThreshold: double
#m_count: int
#m_readingCount: int
#m_lastPartPose: OrientedPoint
#m_odoPose: OrientedPoint
#m_pose: OrientedPoint
#m_linearDistance: double
#m_angularDistance: double
#neff: double
#xmin: double
#ymin: double
#xmax: double
#ymax: double
#delta: double
#regScore: double
#crtScore: double
#maxMove: double
#linearThresholdDistance: double
#angularThresholdDistance: double
#totoSigmaGain: double
#m_outputStream: std::ofstream
#m_infoStream: std::ostream&
+constructor+GridSlamProcessor()
+constructor+GridSlamProcessor(infoStr: std::ostream&)
+constructor#GridSlamProcessor(gsp: const GridSlamProcessor&)
+destructor+~GridSlamProcessor()
+clone(): GridSlamProcessor*
+setSensorMap(smap: SensorMap&)
+init(siza: unsigned int, xmin: double, ymin: double, xmax: double, ymax: double, delta: double, initialPose: OrientedPoint = OrientedPoint(0, 0, 0))
+setMatchingParameters(urange: double, range: double, sigma: double, kernsize: int, lopt: double, aopt: double, iterations: int, likelihoodSigma: double = 1, likelihoodGain: double = 1, likelihoodSkip: unsigned int = 0)
+setMotionModelParameters(srr: double, srt: double, str: double, stt: double)
+setUpdateDistances(linear: double, angular: double, resampleThreshold: double)
+setUpdatePeriod(p: double)
+processTruePos(odometry: const OdometryReading&)
+processScan(reading: const RangeReading&, adaptParticles: int = 0): bool
+getTrajectories(): TNodeVector
+integrateScanSequence(node: TNode*)
+outputStream(): std::ofstream&
+infoStream(): std::ostream&
+getParticles(): const ParticleVector&
+getIndexes(): const std::vector<unsigned int> &
+getBestParticleIndex(): int
+virtual+onOdometryUpdate()
+virtual+onResampleUpdate()
+virtual+onScanmatchUpdate()
-scanMatch(plainReading: const double *)
-normalize()
-resample(plainReading: const double*, adaptParticles: int, rr: const RangeReading* = 0): bool
-updateTreeWeights(weightsAreaedyNormalized: bool = false)
-resetTree()
-propagateWeights()
«dataType» TNodeVector
«dataType» TNodeDeque
«dataType» ParticleVector
+parent
«struct» TNode
+pose: OrientedPoint
+weight: double
+accWeight: double
+gweight: double
+parent: TNode*
+reading: const RangeReading*
+childs: unsigned int
+visitCounter: mutable unsigned int
+flag: mutable bool
+constructor+TNode(pose: OrientedPose&, weight: double, parent: TNode* = 0, childs: unsigned int = 0)
«struct» Particle
+map: ScanMatcherMap
+pose: OrientedPoint
+previousPose: OrientedPoint
+weight: double
+weightSum: double
+gweight: double
+previousIndex: int
+node: TNode*
+constructor+Particle(map: const ScanMatcherMap&)
+operator+=double():: double
+operator+=OrientedPoint(): OrientedPoint
+setWeight(w: double)
«struct» MotionModel
+srr: double
+str: double
+srt: double
+stt: double
+drawFromMotion(p: const OrientedPoint&, linearMove: double, angularMove): OrientedPoint
+drawFromMotion(p: const OrientedPoint&, pnew: const OrientedPoint&, pold: const OrientedPoint&) OrientedPoint
+gaussianApproximation(pnew: const OrientedPoint&, pold: const OrientedPoint&): Covariance3
Key = const GridSlamProcessor::TNode*
TP: GridSlamProcessor::TNode*
«dataType» TNodeMultimap
point
«struct» point
+x: T
+y: T
+constructor+point()
+constructor+point(_x: T, _y: T)
orientedpoint
+theta: A
+constructor+orientedpoint()
+constructor+orientedpoint(p: point<T>&)
+constructor+orientedpoint(_x: T, _y: T, _theta: A)
+normalize()
+rotate(alpha: A): orientedpoint
«dataType» OrientedPoint
«dataType» IntPoint
«dataType» Point
scanmatcher
ScanMatcher
+null_likelihood: const double
#m_activeAreaComputed: bool
#m_laserBeams: unsigned int
#m_laserAngles[LASER_MAXBEAMS]: double
#m_laserPose: OrientedPoint
#m_laserMaxRange: double
#m_usableRange: double
#m_gaussianSigma: double
#m_likelihoodSigma: double
#m_kernelSize: int
#m_optAngularDelta: double
#m_optLinearDelta: double
#m_optRecursiveIterations: unsigned int
#m_likelihoodSkip: unsigned int
#m_llsamplerange: double
#m_llsamplestep: double
#m_lasamplerange: double
#m_lasamplestep: double
#m_generateMap: bool
#m_enlargeStep: double
#m_fullness Threshold: double
#m_angularOdometryReliability: double
#m_linearOdometryReliability: double
#m_freeCellRatio: double
#m_initialBeamsSkip: unsigned int
#m_linePoints: IntPoint*
+constructor+ScanMatcher()
+destructor+~ScanMatcher()
+icpOptimize(pnew: OrientedPoint&, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+optimize(pnew: OrientedPoint&, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+optimize(mean: OrientedPoint&, cov: CovarianceMatrix&, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+registerScan(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+setLaserParameters(beams: unsigned int, angles: double*, lpose: const OrientedPoint&)
+setMatchingParameters(urange: double, range: double, sigma: double, kernsize: int, lopt: double, aopt: double, iterations: int, likelihoodSigma: double = 1, likelihoodSkip: unsigned int = 0)
+invalidateActiveArea()
+computeActiveArea(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
+icpStep(pnew: OrientedPoint &, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+score(map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+likelihoodAndScore(s: double&, l: double&, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): unsigned int
+likelihood(lmax: double&, mean: OrientedPoint&, cov: CovarianceMatrix&, map: const ScanMatcherMap&, p: const OrientedPoint&, readings: const double*): double
+likelihood(_lmax: double&, _mean: OrientedPoint&, _cov: CovarianceMatrix&, map: const ScanMatcherMap&, p: const OrientedPoint&, odometry: Gaussian3&, readings: const double*, gain: double = 180.): double
+laserAngles(): const double*
+laserBeams(): unsigned int
+setlaserPose(): OrientedPoint
+setlaserPose(laserPose: OrientedPoint)
+setlaserMaxRange(): double
+setlaserMaxRange(laserMaxRange: double)
+Operation6()
+Operation7()
+Operation8()
+Operation9()
+Operation12()
+Operation13()
+Operation14()
+Operation15()
+Operation16()
«dataType» CovarianceMatrix
smmap
«struct» PointAccumulator
+unknown_ptr: PointAccumulator*
+acc: FloatPoint
+n: int
+visits: int
+constructor+PointAccumulator()
+constructor+PointAccumulator(): int)
+update(value: bool, p: const PointA& = Point(0,0))
+mean(): Point
+operator+=double()
+add(p: const PointAccumulator&)
+Unknown(): const PointAccumulator&
+entropy(): double
«dataType» FloatPoint
T = float
Cell=PointAccumulator
Storage=HierarchicalArray2D<PointAccumulator>
«dataType» ScanMatcherMap
sensor
«dataType» SensorMap
Sensor
#m_name
+constructor+Sensor()
+destructor+~Sensor()
+getName(): std::str
+setName(name: const std::str &)
«struct» Beam
+pose: OrientedPoint
+span: double
+maxRange: double
+s: double
+c: double
RangeSensor
+newFormat: bool
#m_pose: OrientedPoint
#m_beams: vector<Beam>
+constructor+RangeSensor(name: string)
+constructor+RangeSensor(name: string, beams: unsigned int, res: double, position: OrientedPoint & = OrientedPoint(0, 0, 0), span: double = 0, maxrange: double = 89.0)
+inline+beams(): const vector<Beam>&
+inline+beams(): vector<Beam>&
+inline+getPose(): const OrientedPoint
+updateBeamsLookup()
OdometrySensor
#m_ideal: bool
+constructor+OdometrySensor(name: const std::str &, ideal: bool)
+bIdeal(): bool
grid
«enumeration» AccessibilityState
Outside=0x0
Inside=0x1
Allocated=0x2
Cell
debug: const bool = false
Array2D
+m_cells: Cell **
#m_xsize: int
#m_ysize: int
+constructor+Array2D(xsize: int = 0, ysize: int = 0)
+constructor+Array2D(g: const Array2D<Cell,debug> &))
+destructor+~Array2D()
+operator+=(g: const Array2D<Cell, debug>&): Array2D&
+clear()
+resize(xmin: int, ymin: int, xmax: int, ymax: int)
+sinside(x: int, y: int): bool
+sinside(p: const IntPoint&): bool
+cell(x: int, y: int): const Cell&
+cell(x: int, y: int): Cell&
+cell(p: const IntPoint&): const Cell&
+cell(p: const IntPoint&): Cell&
+cellState(x: int, y: int): AccessibilityState
+cellState(p: const IntPoint&): AccessibilityState
+getPatchSize(): int
+getPatchMagnitude(): int
+getXSize(): int
+getYSize(): int
+cells(): Cell**
«dataType» PointSet
_Key=point<int>
_Comparator=pointcomparator<int>
Cell
HierarchicalArray2D
#m_activeArea: PointSet
#m_patchMagnitude: int
#m_patchSize: int
+constructor+HierarchicalArray2D(xsize: int, ysize: int, patchMagnitude: int = 5)
+constructor+HierarchicalArray2D(hg: const HierarchicalArray2D&)
+destructor+~HierarchicalArray2D()
+operator+=(hg: const HierarchicalArray2D&): HierarchicalArray2D&
+resize(xmin: int, ymin: int, xmax: int, ymax: int)
+getPatchSize(): int
+getPatchMagnitude(): int
+cell(x: int, y: int): const Cell&
+cell(x: int, y: int): Cell&
+cell(p: const IntPoint&): const Cell&
+cell(p: const IntPoint&): Cell&
+cellState(x: int, y: int): AccessibilityState
+cellState(p: const IntPoint&): AccessibilityState
+isAllocated(x: int, y: int): bool
+isAllocated(p: const IntPoint&): bool
+patchIndexes(x: int, y: int): IntPoint
+patchIndexes(p: const IntPoint&): IntPoint
+setActiveArea(aa: const PointSet&, patchCoords: bool = false)
+getActiveArea(): const PointSet&
+allocActiveArea()
#createPatch(p: const IntPoint&): Array2D<Cell> *
map
«dataType» DoubleArray2D
Cell = double
Cell
Storage
bClass: const bool = true
Map
#m_center: Point
#m_worldSizeX: double
#m_worldSizeY: double
#m_delta: double
#m_storage: Storage
#m_mapSizeX: int
#m_mapSizeY: int
#m_sizeX2: int
#m_sizeY2: int
#m_unknown: const Cell
+constructor+Map(mapSizeX: int, mapSizeY: int, delta: double)
+constructor+Map(center: const PointS, worldSizeX: double, worldSizeY: double, delta: double)
+constructor+Map(const center: PointS, xmin: double, ymin: double, xmax: double, ymax: double, delta: double)
+resize(xmin: double, ymin: double, xmax: double, ymax: double)
+grow(xmin: double, ymin: double, xmax: double, ymax: double)
+world2map(p: const PointS): IntPoint
+world2map(x: double, y: double): IntPoint
+map2world(p: const IntPoint&): Point
+map2world(x: int, y: int): Point
+getCenter(): Point
+getWorldSizeX(): double
+getWorldSizeY(): double
+getMapSizeX(): int
+getMapSizeY(): int
+getDelta(): double
+getMapResolution(): double
+getResolution(): double
+getSize(xmin: double &, ymin: double&, xmax: double&, ymax: double&)
+cell(x: int, y: int): Cell&
+cell(p: const IntPoint&): Cell&
+cell(x: int, y: int): const Cell&
+cell(x: double, y: double): Cell&
+cell(p: const Point&): Cell&
+cell(p: const IntPoint&): const Cell&
+cell(x: double, y: double): const Cell&
+cell(p: const Point&): const Cell&
+isInside(x: int, y: int): bool
+isInside(p: const IntPoint&): bool
+isInside(x: double, y: double): bool
+isInside(p: const Point&): bool
+storage(): Storage&
+storage(): const Storage&
+toDoubleArray(): DoubleArray2D*
+toDoubleMap(): Map<double, DoubleArray2D, false>*


ROS SLAM Gmapping
SlamGmapping
-node_: NodeHandle
-entropy_publisher_: Publisher
-sst_: Publisher
-sstm_: Publisher
-ss_: ServiceServer
-tf_: TransformListener
-scan_filter_sub_: Subscriber<LaserScan>*
-scan_filter_: MessageFilter<LaserScan>*
-tfB_: TransformBroadcaster*
-gsp_: GridSlamProcessor*
-gsp_laser_: RangeSensor*
-laser_angles_: vector<double>
-centered_laser_pose_: Stamped<Pose>
-do_reverse_range_: bool
-gsp_laser_beam_count_: unsigned int
-gsp_odom_: OdometrySensor*
-got_first_scan_: bool
-got_map_: bool
-map_: Response
-map_update_interval_: Duration
-map_to_odom_: Transform
-map_to_odom_mutex_: mutex
-map_mutex_: mutex
-laser_count_: int
-throttle_scans_: int
-transform_thread_: thread*
-base_frame_: string
-laser_frame_: string
-map_frame_: string
-odom_frame_: string
-maxRange_: double
-maxUrange_: double
-maxrange_: double
-minimum_score_: double
-sigma_: double
-kernelSize_: int
-lstep_: double
-astep_: double
-iterations_: int
-lsigma_: double
-ogain_: double
-lskip_: int
-srr_: double
-srt_: double
-str_: double
-stt_: double
-linearUpdate_: double
-angularUpdate_: double
-temporalUpdate_: double
-resampleThreshold_: double
-particles_: int
-xmin_: double
-ymin_: double
-xmax_: double
-ymax_: double
-delta_: double
-occ_thresh_: double
-llsamplerange_: double
-llsamplestep_: double
-lasamplerange_: double
-lasamplestep_: double
-private_nh_: NodeHandle
-seed_: unsigned long int
-transform_publish_period_: double
-tf_delay_: double
«constructor»+SlamGmapping()
«constructor»+SlamGmapping(nh: NodeHandle, pnh: NodeHandle)
«constructor»+SlamGmapping(seed: unsigned long int, max_duration_buffer: unsigned long int)
«destructor»+~~SlamGmapping()
+init()
+startLiveSlam()
+startReplay(bag_fname: const string *, scan_topic: string)
+publishTransform()
+laserCallback(scan: const LaserScan*)
+mapCallback(req: Request&, res Response&)
+publishLoop(transform_publish_period: double)
-updateMap(scan: LaserScan&)
-getOdomPose(gmap_pose: OrientedPose&, t: const Time&): bool
-initMapper(scan: const LaserScan&): bool
-addScan(scan: const LaserScan&, gmap_pose: OrientedPose&): bool
-computePoseEntropy()


ROS
ros
Duration
Time
NodeHandle
Publisher
tf
TransformListener
MessageFilter
M
TransformBroadcaster
Stampsd
T
Pose
message_filters
Subscriber
M
messages
std_msgs
Header
+seq: uint32
+stamp: time
+frame_id: string
+header
sensor_msgs
LaserScan
+angle_min: float32
+angle_max: float32
+angle_increment: float32
+time_increment: float32
+scan_time: float32
+range_min: float32
+range_max: float32
+ranpes: float32[]
+intensities: float32[]
nav_msgs
«struct»
GetMap
+request: Request
+response: Response
«dataType»
Request
«dataType»
Response


interaction SOE_Laser_Messsage
/scan: LaserScan
gn_: SlamGmapping
gn_.gsp_laser_: RangeSensor
gn_.gsp_: GridSlamProcessor
gn_.gsp_.m_motionmodel: MotionModel
gn_.gsp_.m_matcher: ScanMatcher
1 : laserCallback(scan: const LaserScan*)
alt
[if !got_first_scan]
2 : initMapper(scan: const LaserScan&)
3 : RangeSensor("FLASER", gsp_laserbeam_count_, fabs(scan.angle_increment), gmap_pose, 0.0, maxRange_)
4 : updateBeamsLookup()
5 :
6 : setSensorMap(smap: SensorMap&)
7 :
8 : addScan(scan: const LaserScan&, gmap_pose: OrientedPose&)
9 : getOdomPose(gmap_pose: OrientedPose&, t: const Time&)
10 : processScan(reading: const RangeReading&, adaptParticles: int = 0)
loop particle it
[m_particles.end()]
11 : drawFromMotion(it->pose,relPose, m_odoPose)
12 :
1)
Sample from Proposal Distribution (Motion Model)
13 : onOdometryUpdate()
14 : accumulateMotion()
alt timeToScan
[if !m_count || timeToScan]
alt firstScan
15 : scanMatch(plainReading: const double *)
[if m_count>0]
loop particle it
[m_particles.end()]
16 : optimize(corrected, it->map, it->pose, plainReading)
17 :
2)
Improved Proposal
18 : likelihoodAndScore(s, l, it->map, it->pose, plainReading)
19 :
3)
Importance Weight
20 : invalidateActiveArea()
21 :
22 : computeActiveArea(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
23 :
24 : onScanmatchUpdate()
25 : updateTreeWeights(weightsAreadyNormalized=false)
26 : resample(plainReading, adaptParticles,readingCopy)
alt lowEffPart
[if neff < part*thresh]
27 : resample()
28 : onResampleUpdate()
29 : buildTree()
30 : deleteOldParticles()
4)
Selective Resampling
loop particle it
[m_particles.end()]
31 : invalidateActiveArea()
32 :
33 : registerScan(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
34 :
5)
Map Update
[else]
loop particle it
[m_particles.end()]
35 : invalidateActiveArea()
36 :
37 : registerScan(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
38 :
5.b)
Map Update
[else]
loop particle it
[m_particles.size()]
39 : invalidateActiveArea()
40 :
41 : computeActiveArea(it->map:ScanMatcherMap, it->pose:OrientedPoint,plainReading:double*)
42 :
43 : registerScan(it->map:ScanMatcherMap, it->pose:OrientedPoint,plainReading:double*)
44 :
5.0)
Map Update
45 : addRootNodeToParticle
46 : updateTreeWeights(weightsAlreadyNormalized: bool = false)
47 : updatePastPoses
48 :
alt addScan
[if addScan]
49 : getBestParticlePose
50 : getBestParticleIndex()
51 :
52 : getParticles()
53 :
54 : updateTF
alt mapUpdate
[!gotMap || timeForUpdate]
55 : updateMap(scan: LaserScan&)
«create»
56 :
matcher: ScanMatcher
57 : SetParams()
58 :
59 : computePoseEntropy()
60 : invalidateActiveArea()
61 :
62 : computeActiveArea(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
63 :
64 : registerScan(map: ScanMatcherMap&, p: const OrientedPoint&, readings: const double*)
65 :
«destroy»
66 :
67 :


interaction SOE_startLiveSlam
main
gn_: SlamGmapping
1 : startLiveSlam()
2 :
4 : initMappel(scan: const LaserScan&)


main
Lifeline1
«create»
1 : gn_=SlamGmapping()
gn_: SlamGmapping
2 : init()
3 :