

Testing and Refactoring NDT-MCL

Aleksandar Mitrevski, Santosh Thoduka

Hochschule Bonn Rhein Sieg

aleksandar.mitrevski@smail.inf.h-brs.de

santosh.thoduka@smail.inf.h-brs.de

June 24, 2014

Normal Distribution Transform - Monte Carlo Localisation

- Normal distribution transform Monte Carlo localisation (NDT-MCL) [1] is a localisation algorithm whose goal is providing an estimate of the true position of a robot in a given environment
- NDT-MCL parametrizes grid cells by Gaussian distributions, employing a so-called NDT representation as suggested by [2]
- Localisation involves calculating a pose likelihood, updating particle weights, and resampling particles.
- The only difference in NDT-MCL is the likelihood calculation, such that L_2 -likelihood is used to express the discrepancy between the estimated and the actual robot pose.

Node Description

mcl

scan

```

+~: float*
+~: float*
+~: int
+scan()
+scan(s:scan6)
+operator=(s:scan6): scan6
+allocate(n:int): void
+copy(new_scan:scan6): void
+set(r:float*,aa:float): void
    
```

pose

```

+~: float
+~: float
+~: float
+pose()
+pose(x:float, y:float, a:float)
+set(p:pose6): void
+set(x:float, y:float, a:float): void
+setTODifferentialPose(od:cur:pose,odo_ref:pose): void
+integratedDifferential(diff:pose): pose
+to2PI(): void
+toPI(): void
+~pose()
    
```

TPoseParticle

```

+~: float
+~: float
+~: float
+~: float
+~: float
+TPoseParticle()
+to2PI(): void
+toPI(): void
    
```

1...*

CParticleFilter

```

+Particles: TPoseParticle*
+Lik: float
+outliers: float
+NumOfParticles: int
+size: int
+average: mcl::pose
+variance: mcl::pose
+isAvgSet: bool
+myrand: ownRandom
+tmp: TPoseParticle*
+allocate(num_particles:int): void
+myfree(): void
+getDistributionMean(doweighting:bool=false): mcl::pose
+averageOverNBestAndRandomize(N:int,Mc:int=0, vx:float=0, vy:float=0): mcl::pose
+getDistributionVariances(): Eigen::Matrix3d
+initializeNormalRandom(pbmcl:pose, variance:mcl::pose, size:int): void
+initializeUniform(Pmin:mcl::pose,Pmax:mcl::pose, dPbmcl:pose): void
+SIRUpdate(): void
+normalize(): void
+predict(dp:mcl::pose,std:mcl::pose): void
+predict(mcl::pose:dP,Q:float[4]): void
+updateLikelihood(Lik:float*): void
+resize(n:int): void
+print(): void
+saveToFile(Fileind:int): void
+hpstr(indx:int*): void
    
```

ownRandom

```

+normalRandomCache: float*
+isOk: bool
+size: int
+ownRandom()
+ownRandom()
+allocate(n_size:int): void
+fillCache(): void
+getCachedUniformRandom(): float
+uniformRandom(): void
+normalRandom(): void
+covRandom(xRand:floats,yRand:floats,cov:float[3]): void
    
```

NdtMclNode

```

+nodeHandle: ros::NodeHandle
+paramHandle: ros::NodeHandle
+~: mcl::ros::Publisher
+~: mcl::ros::Publisher
+initial_pose_sub: ros::Subscriber
+scan_sub: ros::Subscriber
+userInitialPose: bool
+hasNewInitialPose: bool
+ipos_x: double
+ipos_y: double
+ipos_yaw: double
+ivar_x: double
+ivar_y: double
+ivar_yaw: double
+ndtmcl: NDTMCL::pcl::PointXYZ*
+offx: float
+offy: float
+offz: float
+Told: Eigen::Affine3d
+Told: Eigen::Affine3d
+TT: mrpt::utils::CicTac
+tf_odo: string
+tf_state: string
+tf_laser_link: string
+tf_world: string
+tf_timestamp_tolerance: ros::Duration
+tf_tolerance: double
+translation_noise_tolerance: double
+rotation_noise_tolerance: double
+callback(scan:sensor_msgs::LaserScan:ConstPtr6): void
+initialPoseReceived(msg:geometry_msgs::PoseWithCovarianceStampedConstPtr6): void
+mapToRviz(map:islGeneric::NDTMap::pcl::PointXYZ*): void
+sendROSodoMessage(mean:Eigen::Vector3d, cov:Eigen::Matrix3d,ts:ros::Time): bool
+getAsAffine(x:float,y:float,yaw:float): Eigen::Affine3d
    
```

NDTMCL

```

+map: islGeneric::NDTMap::PointT*
+~: mcl::CParticleFilter
+resolution: double
+counter: int
+zfilter_min: double
+forceSIR: bool
+isInit: bool
+sinceSIR: int
+NDTMCL(map_resolution:double,nd_map:islGeneric::NDTMap::PointT*6, zfilter:double)
+initializeFilter(x:double,y:double,yaw:double, x_e:double,y_e:double,yaw_e:double, numParticles:int): void
+updateAndPredict(Tmotion:Eigen::Affine3d, cloud:pcl::PointCloud::PointT*6): void
+getAsAffine(i:int): Eigen::Affine3d
    
```

Implemented Modifications

- Re-factored node as a class
- Removed use of ground truth
- Pose estimate published as transform between map and odom frame
- Added parameters to the launch file - such as tf frames, translational and rotational tolerances, etc.

Tested localisation both in simulation and in a real environment

- Poor localisation during mapping using the NDT Fuser node (for mapping)
- Map is not very accurate
- Incorrect pose estimation while robot is stationary
- Pose estimate “jumps”

Improvements and Future Work

- More parameters should be exposed in the launch file so the node can be tuned to different platforms and environments if necessary
- Resolution of map should be stored in the map file
- Investigate compatibility with navigation planners
- Port visualization to Rviz

- [1] J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "Normal Distributions Transform Monte-Carlo Localization (NDT-MCL)," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, Tokyo, Japan, 2013, pp. 382-389.
- [2] P. Biber, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Las Vegas, NV, 2003, pp. 2743-2748.