indigo | kinetic | lunar | melodic *Show EOL distros:*

Documentation Status

navigation (/navigation?distro=melodic)*: amcl | base_local_planner (/base_local_planner?distro=melodic) | carrot_planner (/carrot_planner?distro=melodic) | clear_costmap_recovery (/clear_costmap_recovery?distro=melodic) | costmap_2d (/costmap_2d?distro=melodic) | dwa_local_planner (/dwa_local_planner?distro=melodic) | fake_localization (/fake_localization?distro=melodic) | global_planner (/global_planner?distro=melodic) | map_server (/map_server?distro=melodic) | move_base (/move_base?distro=melodic) | move_base_msgs (/move_base_msgs?distro=melodic) | move_slow_and_clear (/move_slow_and_clear?distro=melodic) | nav_core (/nav_core?distro=melodic) | navfn (/navfn?distro=melodic) | rotate_recovery (/rotate_recovery?distro=melodic) | voxel_grid (/voxel_grid?distro=melodic)*

Package Links

- **Code API (http://docs.ros.org/melodic/api/amcl/html)**
- Tutorials (/amcl/Tutorials)
- FAQ (http://answers.ros.org/questions/scope:all/sort:activity-desc/tags:amcl/page:1/)
- Changelog (http://docs.ros.org/melodic/changelogs/amcl/changelog.html)
- Change List (/navigation/ChangeList)
- Reviews (/amcl/Reviews)

Dependencies (14)
Used by (6)
Jenkins jobs (9)

Package Summary

✔ **Released** ✔ **Continuous Integration: 91 / 91** ✔ **Documented**

amcl is a probabilistic localization system for a robot moving in 2D. It implements the adaptive (or KLD-sampling) Monte Carlo localization approach (as described by Dieter Fox), which uses a particle filter to track the pose of a robot against a known map.

This node is derived, with thanks, from Andrew Howard's excellent 'amcl' Player driver.

- Maintainer status: maintained
- Maintainer: David V. Lu!! <davidvlu AT gmail DOT com>, Michael Ferguson <mfergs7 AT gmail DOT com>, Aaron Hoy <ahoy AT fetchrobotics DOT com>
- Author: Brian P. Gerkey, contradict@gmail.com
- License: LGPL
- Source: git https://github.com/ros-planning/navigation.git (https://github.com/ros-planning/navigation) (branch: melodic-devel)

Contents

1. Algorithms
2. Example
3. Nodes
    1. amcl
        1. Subscribed Topics
        2. Published Topics
        3. Services
        4. Services Called
        5. Parameters
        6. Transforms

1. Algorithms

Many of the algorithms and their parameters are well-described in the book Probabilistic Robotics, by Thrun, Burgard, and Fox. The user is advised to check there for more detail. In particular, we use the following algorithms from that book: **`sample_motion_model_odometry`**, **`beam_range_finder_model`**, **`likelihood_field_range_finder_model`**, **`Augmented_MCL`**, and **`KLD_Sampling_MCL`**.

As currently implemented, this node works only with laser scans and laser maps. It could be extended to work with other sensor data.

2. Example

To localize using laser data on the `base_scan` topic:

```
amcl scan:=base_scan
```

3. Nodes

3.1 amcl

`amcl` takes in a laser-based map, laser scans, and transform messages, and outputs pose estimates. On startup, `amcl` initializes its particle filter according to the parameters provided. Note that, because of the defaults, if no parameters are set, the initial filter state will be a moderately sized particle cloud centered about (0,0,0).

3.1.1 Subscribed Topics

`scan` (sensor_msgs/LaserScan (http://docs.ros.org/api/sensor_msgs/html/msg/LaserScan.html))
: Laser scans.

`tf` (tf/tfMessage (http://docs.ros.org/api/tf/html/msg/tfMessage.html))
: Transforms.

`initialpose` (geometry_msgs/PoseWithCovarianceStamped (http://docs.ros.org/api/geometry_msgs/html/msg/PoseWithCovarianceStamped.html))

Mean and covariance with which to (re-)initialize the particle filter.

`map` (nav_msgs/OccupancyGrid (http://docs.ros.org/api/nav_msgs/html/msg/OccupancyGrid.html))
When the `use_map_topic` parameter is set, AMCL subscribes to this topic to retrieve the map used for laser-based localization. **New in navigation 1.4.2.**

3.1.2 Published Topics

`amcl_pose` (geometry_msgs/PoseWithCovarianceStamped (http://docs.ros.org/api/geometry_msgs/html/msg/PoseWithCovarianceStamped.html))
Robot's estimated pose in the map, with covariance.

`particlecloud` (geometry_msgs/PoseArray (http://docs.ros.org/api/geometry_msgs/html/msg/PoseArray.html))
The set of pose estimates being maintained by the filter.

`tf` (tf/tfMessage (http://docs.ros.org/api/tf/html/msg/tfMessage.html))
Publishes the transform from `odom` (which can be remapped via the ~odom_frame_id parameter) to `map`.

3.1.3 Services

`global_localization` (std_srvs/Empty (http://docs.ros.org/api/std_srvs/html/srv/Empty.html))
Initiate global localization, wherein all particles are dispersed randomly through the free space in the map.

`request_nomotion_update` (std_srvs/Empty (http://docs.ros.org/api/std_srvs/html/srv/Empty.html))
Service to manually perform update and publish updated particles.

`set_map` (nav_msgs/SetMap (http://docs.ros.org/api/nav_msgs/html/srv/SetMap.html))
Service to manually set a new map and pose.

3.1.4 Services Called

`static_map` (nav_msgs/GetMap (http://docs.ros.org/api/nav_msgs/html/srv/GetMap.html))
amcl calls this service to retrieve the map that is used for laser-based localization; startup blocks on getting the map from this service.

3.1.5 Parameters

There are three categories of ROS Parameters (/Parameters) that can be used to configure the `amcl` node: overall filter, laser model, and odometery model.

Overall filter parameters

`~min_particles` (`int`, default: 100)

Minimum allowed number of particles.

`~max_particles` (`int`, default: 5000)
Maximum allowed number of particles.

`~kld_err` (`double`, default: 0.01)
Maximum error between the true distribution and the estimated distribution.

~kld_z (double, default: 0.99)
Upper standard normal quantile for (1 - p), where p is the probability that the error on the estimated distrubition will be less than kld_err.

~update_min_d (double, default: 0.2 meters)
Translational movement required before performing a filter update.

~update_min_a (double, default: π/6.0 radians)
Rotational movement required before performing a filter update.

~resample_interval (int, default: 2)
Number of filter updates required before resampling.

~transform_tolerance (double, default: 0.1 seconds)
Time with which to post-date the transform that is published, to indicate that this transform is valid into the future.

~recovery_alpha_slow (double, default: 0.0 (disabled))
Exponential decay rate for the slow average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.001.

~recovery_alpha_fast (double, default: 0.0 (disabled))
Exponential decay rate for the fast average weight filter, used in deciding when to recover by adding random poses. A good value might be 0.1.

~initial_pose_x (double, default: 0.0 meters)
Initial pose mean (x), used to initialize filter with Gaussian distribution.

~initial_pose_y (double, default: 0.0 meters)
Initial pose mean (y), used to initialize filter with Gaussian distribution.

~initial_pose_a (double, default: 0.0 radians)
Initial pose mean (yaw), used to initialize filter with Gaussian distribution.

~initial_cov_xx (double, default: 0.5*0.5 meters)
Initial pose covariance (x*x), used to initialize filter with Gaussian distribution.

~initial_cov_yy (double, default: 0.5*0.5 meters)
Initial pose covariance (y*y), used to initialize filter with Gaussian distribution.

~initial_cov_aa (double, default: (π/12)*(π/12) radian)
Initial pose covariance (yaw*yaw), used to initialize filter with Gaussian distribution.

~gui_publish_rate (double, default: -1.0 Hz)
Maximum rate (Hz) at which scans and paths are published for visualization, -1.0 to disable.

~save_pose_rate (double, default: 0.5 Hz)
Maximum rate (Hz) at which to store the last estimated pose and covariance to the parameter server, in the variables ~initial_pose_* and ~initial_cov_*. This saved pose will be used on subsequent runs to initialize the filter. -1.0 to disable.

~use_map_topic (bool, default: false)

When set to true, AMCL will subscribe to the `map` topic rather than making a service call to receive its map. **New in navigation 1.4.2**

`~first_map_only` (`bool`, default: false)
When set to true, AMCL will only use the first map it subscribes to, rather than updating each time a new one is received. **New in navigation 1.4.2**

Laser model parameters

Note that whichever mixture weights are in use should sum to 1. The beam model uses all 4: z_hit, z_short, z_max, and z_rand. The likelihood_field model uses only 2: z_hit and z_rand.

`~laser_min_range` (`double`, default: -1.0)

Minimum scan range to be considered; -1.0 will cause the laser's reported minimum range to be used.

`~laser_max_range` (`double`, default: -1.0)
Maximum scan range to be considered; -1.0 will cause the laser's reported maximum range to be used.

`~laser_max_beams` (`int`, default: 30)
How many evenly-spaced beams in each scan to be used when updating the filter.

`~laser_z_hit` (`double`, default: 0.95)
Mixture weight for the z_hit part of the model.

`~laser_z_short` (`double`, default: 0.1)
Mixture weight for the z_short part of the model.

`~laser_z_max` (`double`, default: 0.05)
Mixture weight for the z_max part of the model.

`~laser_z_rand` (`double`, default: 0.05)
Mixture weight for the z_rand part of the model.

`~laser_sigma_hit` (`double`, default: 0.2 meters)
Standard deviation for Gaussian model used in z_hit part of the model.

`~laser_lambda_short` (`double`, default: 0.1)
Exponential decay parameter for z_short part of model.

`~laser_likelihood_max_dist` (`double`, default: 2.0 meters)
Maximum distance to do obstacle inflation on map, for use in likelihood_field model.

`~laser_model_type` (`string`, default: `"likelihood_field"`)
Which model to use, either `beam`, `likelihood_field`, or `likelihood_field_prob` (same as `likelihood_field` but incorporates the beamskip feature, if enabled).

Odometry model parameters

If `~odom_model_type` is `"diff"` then we use the `sample_motion_model_odometry algorithm` from Probabilistic Robotics, p136; this model uses the noise parameters `odom_alpha_1` through `odom_alpha4`, as defined in the book.

If `~odom_model_type` is `"omni"` then we use a custom model for an omni-directional base, which uses `odom_alpha_1` through `odom_alpha_5`. The meaning of the first four parameters is similar to that for the `"diff"` model. The fifth parameter capture the tendency of the robot to translate (without rotating) perpendicular to the observed direction of travel.

A bug (https://github.com/ros-planning/navigation/issues/20) was found and fixed. But fixing the old models would have changed or broken the localisation of already tuned robot systems, so the new fixed odometry models were added as new types `"diff-corrected"` and `"omni-corrected"`. The default settings of the `odom_alpha` parameters only fit the old models, for the new model these values probably need to be a lot smaller, see http://answers.ros.org/question/227811/tuning-amcls-diff-corrected-and-omni-corrected-odom-models/ (http://answers.ros.org/question/227811/tuning-amcls-diff-corrected-and-omni-corrected-odom-models/).

`~odom_model_type` (`string`, default: `"diff"`)

Which model to use, either `"diff"`, `"omni"`, `"diff-corrected"` or `"omni-corrected"`.

`~odom_alpha1` (`double`, default: 0.2)
Specifies the expected noise in odometry's rotation estimate from the rotational component of the robot's motion.

`~odom_alpha2` (`double`, default: 0.2)
Specifies the expected noise in odometry's rotation estimate from translational component of the robot's motion.

`~odom_alpha3` (`double`, default: 0.2)
Specifies the expected noise in odometry's translation estimate from the translational component of the robot's motion.

`~odom_alpha4` (`double`, default: 0.2)
Specifies the expected noise in odometry's translation estimate from the rotational component of the robot's motion.

`~odom_alpha5` (`double`, default: 0.2)
Translation-related noise parameter (only used if model is `"omni"`).

`~odom_frame_id` (`string`, default: `"odom"`)
Which frame to use for odometry.

`~base_frame_id` (`string`, default: `"base_link"`)
Which frame to use for the robot base

`~global_frame_id` (`string`, default: `"map"`)
The name of the coordinate frame published by the localization system

`~tf_broadcast` (`bool`, default: true)
Set this to `false` to prevent amcl from publishing the transform between the global frame and the odometry frame.

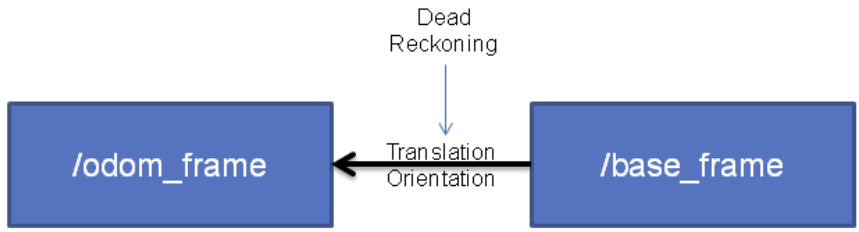
3.1.6 Transforms

`amcl` transforms incoming laser scans to the odometry frame (`~odom_frame_id`). So there must exist a path through the tf (/tf) tree from the frame in which the laser scans are published to the odometry frame.

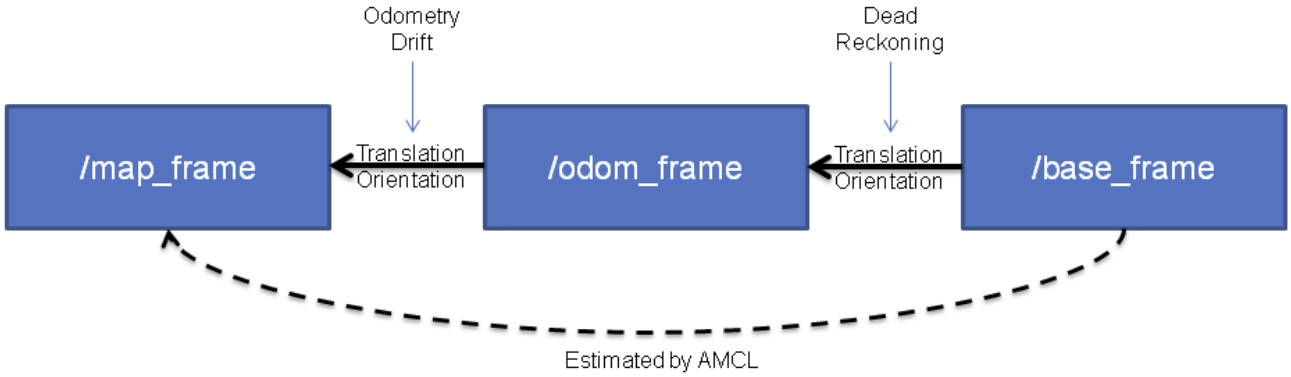
An implementation detail: on receipt of the first laser scan, `amcl` looks up the transform between the laser's frame and the base frame (~base_frame_id), and latches it forever. So `amcl` cannot handle a laser that moves with respect to the base.

The drawing below shows the difference between localization using odometry and `amcl`. During operation `amcl` estimates the transformation of the base frame (~base_frame_id) in respect to the global frame (`~global_frame_id`) but it only publishes the transform between the global frame and the odometry frame (`~odom_frame_id`). Essentially, this transform accounts for the drift that occurs using Dead Reckoning. The published transforms are future dated (/tf/FAQ#Why_do_I_see_negative_average_delay_in_tf_monitor.3F).

Odometry Localization



AMCL Map Localization



Wiki: amcl (last edited 2018-09-24 23:18:22 by SteveMacenski (/SteveMacenski))

(http://www.osrfoundation.org)