kinetic | melodic | noetic *Show EOL distros:* □

Documentation Status

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

Package Links

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

Dependencies (4)

Used by (8)

Jenkins jobs (10)

Package Summary

✓ Released **✓ Continuous Integration: 92 / 92** ▾ **✓ Documented**

map_server provides the `map_server` ROS Node (http://www.ros.org/wiki/Nodes), which offers map data as a ROS Service (http://www.ros.org/wiki/Services). It also provides the `map_saver` command-line utility, which allows dynamically generated maps to be saved to file.

- 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 Gerkey, Tony Pratkanis, contradict@gmail.com
- License: BSD
- Source: git https://github.com/ros-planning/navigation.git (https://github.com/ros-planning/navigation) (branch: noetic-devel)

Contents

1. Map format

Maps manipulated by the tools in this package are stored in a pair of files. The YAML file describes the map meta-data, and names the image file. The image file encodes the occupancy data.

1.1 Image format

The image describes the occupancy state of each cell of the world in the color of the corresponding pixel. In the standard configuration, whiter pixels are free, blacker pixels are occupied, and pixels in between are unknown. Color images are accepted, but the color values are averaged to a gray value.

Image data is read in via SDL_Image (http://www.libsdl.org/projects/SDL_image/docs/index.html); supported formats vary, depending on what SDL_Image provides on a specific platform. Generally speaking, most popular image formats are widely supported. A notable exception is that PNG is not supported on OS X.

1.2 YAML format

The YAML format is best explained with a simple, complete example:

```
image: testmap.png
resolution: 0.1
origin: [0.0, 0.0, 0.0]
occupied_thresh: 0.65
free_thresh: 0.196
negate: 0
```

Required fields:

- **image** : Path to the image file containing the occupancy data; can be absolute, or relative to the location of the YAML file
- **resolution** : Resolution of the map, meters / pixel

- **origin** : The 2-D pose of the lower-left pixel in the map, as (x, y, yaw), with yaw as counterclockwise rotation (yaw=0 means no rotation). Many parts of the system currently ignore yaw.
- **occupied_thresh** : Pixels with occupancy probability greater than this threshold are considered completely occupied.
- **free_thresh** : Pixels with occupancy probability less than this threshold are considered completely free.
- **negate** : Whether the white/black free/occupied semantics should be reversed (interpretation of thresholds is unaffected)

Optional parameter:

- **mode** : Can have one of three values: trinary, scale, or raw. Trinary is the default. More information on how this changes the value interpretation is in the next section.

1.3 Value Interpretation

Given a pixel that has a COLOR value x in the range `[0, 256)`, how should we interpret this value when put into the ROS message? First we convert integer x to a floating point number p depending on the interpretation of the `negate` flag from the yaml.

- If `negate` is false, `p = (255 - x) / 255.0`. This means that black (0) now has the highest value (1.0) and white (255) has the lowest (0.0).
- If `negate` is true, `p = x / 255.0`. This is the non-standard interpretation of images, which is why it is called negate, even though the math indicates that x is not negated. Nomenclature is hard.

1.3.1 Trinary

The standard interpretation is the trinary interpretation, i.e. interpret all values so that the output ends up being one of three values.

- If `p > occupied_thresh`, output the value `100` to indicate the cell is occupied.
- If `p < free_thresh`, output the value `0` to indicate the cell is free.
- Otherwise, output `-1` a.k.a. `255` (as an unsigned char), to indicate that the cell is unknown.

1.3.2 Scale

This tweaks the above interpretation to allow for more output values than trinary.

- As before if `p > occupied_thresh`, output the value `100`. If `p < free_thresh`, output the value `0`.
- Otherwise, output
  `99 * (p - free_thresh) / (occupied_thresh - free_thresh)`

This will allow you to output a full gradient of values ranging from `[0, 100]`. To output `-1`, simply use the alpha channel of a png, where any transparency will be interpreted as unknown.

1.3.3 Raw

This mode will output x for each pixel, so output values are `[0, 255]`.

2. Command-line Tools

2.1 map_server

`map_server` is a ROS node that reads a map from disk and offers it via a ROS service.
The current implementation of the map_server converts color values in the map image data into ternary occupancy values: free (0), occupied (100), and unknown (-1). Future versions of this tool may use the values between 0 and 100 to communicate finer gradations of occupancy.

2.1.1 Usage

```
map_server <map.yaml>
```

2.1.2 Example

```
rosrun map_server map_server mymap.yaml
```

Note that the map data may be retrieved via either latched topic (meaning that it is sent once to each new subscriber), or via service. The service may eventually be phased out.

2.1.3 Published Topics

`map_metadata` (nav_msgs/MapMetaData (http://docs.ros.org/en/api/nav_msgs/html/msg/MapMetaData.html))

Receive the map metadata via this latched topic.

`map` (nav_msgs/OccupancyGrid (http://docs.ros.org/en/api/nav_msgs/html/msg/OccupancyGrid.html))

Receive the map via this latched topic.

2.1.4 Services

`static_map` (nav_msgs/GetMap (http://docs.ros.org/en/api/nav_msgs/html/srv/GetMap.html))

Retrieve the map via this service.

2.1.5 Parameters

`~frame_id` (`string`, default: `"map"`)

The frame to set in the header of the published map.

2.2 map_saver

`map_saver` saves a map to disk, e.g., from a SLAM mapping service.

2.2.1 Usage

```
rosrun map_server map_saver [--occ <threshold_occupied>] [--free <threshold_fr
ee>] [-f <mapname>] map:=/your/costmap/topic
```

`map_saver` retrieves map data and writes it out to **map.pgm** and **map.yaml**. Use the **-f** option to provide a different base name for the output files. The **--occ** and **--free** options take values between 0 and 100. To save different map topic set **map** to your costmap topic.

2.2.2 Examples

```
rosrun map_server map_saver -f mymap
rosrun map_server map_saver --occ 90 --free 10 -f mymap map:=/move_base/global
_costmap/costmap
```

2.2.3 Subscribed Topics

`map` (nav_msgs/OccupancyGrid (http://docs.ros.org/en/api/nav_msgs/html/msg/OccupancyGrid.html))
Map will be retrieved via this latched topic.

Wiki: map_server (last edited 2020-03-23 11:39:33 by Combinacijus (/Combinacijus))