My Project

Generated by Doxygen 1.8.7

Sun May 18 2014 13:59:45

Contents

Chapter 1

OctoMap

1.1 Introduction

The OctoMap library implements a 3D occupancy grid mapping approach. It provides data structures and mapping algorithms. The map is implemented using an Octree. It is designed to meet the following requirements:

- **Full 3D model.** The map is able to model arbitrary environments without prior assumptions about it. The representation models occupied areas as well as free space. If no information is available about an area (commonly denoted as *unknown areas*), this information is encoded as well. While the distinction between free and occupied space is essential for safe robot navigation, information about unknown areas is important, e.g., for autonomous exploration of an environment.

- **Updatable.** It is possible to add new information or sensor readings at any time. Modeling and updating is done in a *probabilistic* fashion. This accounts for sensor noise or measurements which result from dynamic changes in the environment, e.g., because of dynamic objects. Furthermore, multiple robots are able to contribute to the same map and a previously recorded map is extendable when new areas are explored.

- **Flexible.** The extent of the map does not have to be known in advance. Instead, the map is dynamically expanded as needed. The map is multi-resolution so that, for instance, a high-level planner is able to use a coarse map, while a local planner may operate using a fine resolution. This also allows for efficient visualizations which scale from coarse overviews to detailed close-up views.

- **Compact.** The is stored efficiently, both in memory and on disk. It is possible to generate compressed files for later usage or convenient exchange between robots even under bandwidth constraints.

Octomap was developed by Kai M. Wurm and Armin Hornung, and is currently maintained by Armin Hornung. A tracker for bug reports and feature requests is available available on GitHub. You can find an overview at http://octomap.github.com/ and the code repository at https://github.com/OctoMap/octomap.

1.2 Installation

See the file README.txt in the main folder.

1.3 Changelog

See the file CHANGELOG.txt in the main folder or the latest version online.

1.4 Getting Started

Jump right in and have a look at the main class octomap::OcTree OcTree and the examples in src/octomap/simple↩
_example.cpp. To integrate single measurements into the 3D map have a look at OcTree::insertRay(...), to insert full 3D scans (pointclouds) please have a look at OcTree::insertPointCloud(...). Queries can be performed e.g. with OcTree::search(...) or OcTree::castRay(...). The preferred way to batch-access or process nodes in an Octree is with the iterators leaf_iterator, tree_iterator, or leaf_bbx_iterator.

The OcTree class is derived from OccupancyOcTreeBase, with most functionality in the parent class. Also derive from OccupancyOcTreeBase if you you want to implement your own Octree and node classes. You can have a look at the classes OcTreeStamped and OcTreeNodeStamped as examples.

Start the 3D visualization with: **bin/octovis**

You will find an example 3D scan (please bunzip2 first) and an example OctoMap .bt file in the directory **share/data** to try. More data sets are available at `http://ais.informatik.uni-freiburg.↩
de/projects/datasets/octomap/`.

Chapter 2

Octomap - A probabilistic, flexible, and compact 3D mapping library for robotic systems

Authors: Kai M. Wurm and Armin Hornung, University of Freiburg, Copyright (C) 2009-2013. http://octomap.github.com

See the list of contributors for further authors.

REQUIREMENTS

- For only the octomap library: cmake and a regular build environment (gcc)
- For HTML documentation: doxygen (optional)
- For the viewer octovis: Qt4, OpenGL, QGLViewer (optional)

Skip to WINDOWS for tips on compilation under Windows. You can install all dependencies on Ubuntu by running:

```
sudo apt-get install cmake doxygen libqt4-dev libqt4-opengl-dev libqglviewer-qt4-dev
```

INSTALLATION

See http://www.ros.org/wiki/octomap if you want to use OctoMap in ROS! There are pre-compiled packages for octomap, octovis, and ROS integration available.
Build the complete project by changing into the "build" directory and running cmake:

```
mkdir build && cd build
cmake ..
```

Type `make` to compile afterwards. This will create all CMake files cleanly in the `build` folder (Out-of-source build). Executables will end up in `bin`, libraries in `lib`.
A debug configuration can be created by running:

```
cmake -DCMAKE_BUILD_TYPE=Debug ..
```

in `build` or a different directory (e.g. `build-debug`).

You can install the library by running `make install`, though it is usually not necessary. Be sure to adjust `CMAKE_INSTALL_PREFIX` before.

The target `make test` executes the unit tests for the octomap library, if you are interested in verifying the functionality on your machine.

DOCUMENTATION

The documentation for the latest stable release is available online: http://octomap.github.↩
com/octomap/doc/index.html

You can build the most current HTML-Documentation for your current source with Doxygen by running `make docs` in the build directory. The documentation will end up in `doc/html/index.html` in the main directory.

GETTING STARTED

Jump right in and have a look at the example src/octomap/simple_example.cpp

Or start the 3D viewer with `bin/octovis`

You will find an example scan and binary tree to load in the directory `share`. Further examples can be downloaded from the project website.

USE IN OTHER PROJECTS

A CMake-project config is generated for OctoMap which allows OctoMap to be used from other CMake-Projects easily.

Point CMake to your octomap installation so that it finds the file octomap/lib/cmake/octomap/octomap-config.cmake, e.g. by setting the environment variable `octomap_DIR`to the directory containing it.
Then add the following to your CMakeLists.txt:

```
find_package(octomap REQUIRED)
include_directories(${OCTOMAP_INCLUDE_DIRS})
link_libraries(${OCTOMAP_LIBRARIES})
```

In addition to this cmake-module we also provide a pkgconfig-file.

For convenience, there is a minimal example project included in the file share/example-project.tgz

ECLIPSE PROJECT FILES

Eclipse project files can be generated (with some limitations, see: http://www.vtk.org/Wiki/Eclipse↩
_CDT4_Generator) by running:

```
cmake -G"Eclipse CDT4 - Unix Makefiles" ..
```

Import the project (existing project, root is the build folder, do not copy contents) into Eclipse afterwards. For full Eclipse compatibility, it might be necessary to build in the main source directory.

WINDOWS

The octomap library and tools can be compiled and used under Windows although this has not been tested in-depth. Feedback is welcome.

To compile the library you need cmake (http://www.cmake.org).

MinGW

1. Download the MinGW distribution (http://www.mingw.org)
2. Install C++ compiler and add MingGW/bin to your system PATH
3. Start the cmake-gui and set the code directory to the library root (e.g. `/octomap`)

4. Set the build directory to, e.g., `/octomap/build`.
5. Press "Generate", select the appropriate generator, "MinGW Makefiles".
6. Start a command shell and "make" the project:

   ```
   octomap> cd build
   octomap/build> mingw32-make.exe
   ```

You can run the unit tests using ctest on the command prompt:

```
octomap/build> ctest.exe
```

Microsoft Visual Studio 2010

1. Start the cmake-gui and set the code directory to the library root (e.g. `/octomap`)
2. Set the build directory to, e.g., /octomap/build.
3. Press "Generate", select the appropriate generator, e.g. "Visual Studio 10". This generates a solution file octomap.sln
4. Load this file and build the project

You can run the unit tests using ctest on the command prompt:

```
octomap/build> ctest.exe -C Release
```

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 4

Namespace Documentation

4.1 octomap Namespace Reference

4.1.1 Detailed Description

Namespace the OctoMap library and visualization tools

4.2 octomath Namespace Reference

4.2.1 Detailed Description

Namespace of the math library in OctoMap