Compiling and Installing libpointmatcher on Windows

Compiling using MSVC (Microsoft Visual Studio)

In Short...

If you are used to development project, here is what you need:

Name	Link	Version (Tested March 29, 2014)
Windows		7
git	http://windows.github.com/	v1.0
libpointmatcher sources	https://github.com/ethz-asl/libpointmatcher	
libnabo sources	https://github.com/ethz-asl/libnabo	
Visual Studio	http://www.microsoft.com/visualstudio/eng/downloads	Visual Studio 2012 Express for Windows Desktop
CMake	http://www.cmake.org/cmake/resources/software.html	cmake-2.8.11.2-win32-x86.exe
Eigen 3	http://eigen.tuxfamily.org/index.php?title=Main_Page#Download	v3.2.0
Boost	http://www.boost.org/users/download/	v1.54.0
yaml-cpp	https://code.google.com/p/yaml-cpp/downloads/list	v0.3.0, **not working with v0.5.0**
grep tool	http://gnuwin32.sourceforge.net/packages/grep.htm	v2.5.4
gtest	https://code.google.com/p/googletest/	v1.7.0

The rest of this tutorial will guide you through the different requirements step by step.

Install grep tool

Install `grep` by following the instructions in http://gnuwin32.sourceforge.net/packages/grep.htm. You might need to modify the Path environment variables to make sure that grep can be run from anywhere. To test:

```
cd\
grep --version
```

Building Boost

1. Open a console that knows the path to the MSVC compiler command (cl). We suggest to use **Windows PowerShell**. An alternative is from the Start menu in the Visual Studio section; for instance for VS 2012, it is called Developer Command Prompt for VS2012.
2. Go to your Boost source directory, and do:

   `$ bootstrap`

   `$ b2 install --prefix=build address-model=64`
3. It may take awhile to finish.

Build libnabo

1. Start **CMake Gui**
2. Add the path of your libnabo sources in the field *Where is the source code*.
3. Add a folder named build in the field *Where to build the binary*. This will allow you to do out-of-source compilation.
4. Click on the button Configure

   1. Select the generator for the project (Visual Studio 11 Win 64)
   2. Error will be reported, because CMake does not know yet where to find the libraries. The next steps will tell it where to find them.

5. Locate *your eigen folder* in the field **EIGEN_INCLUDE_DIR**
6. Add the following boolean variable and set it to `true`: **Boost_USE_STATIC_LIBS**
7. Add the following PATH variable and set it to *(your boost folder)*/build: **BOOST_ROOT**
8. Change the variable **CMAKE_CONFIGURATION_TYPES** to `RelWithDebInfo`
9. Click on the button Configure again, then on Generate
10. Locate the Microsoft Visual Studio Solution file in the your build folder (libnabo.sln) and open it. Visual Studio should open.
11. Build the solution: BUILD -> Build Solution

    Command line alternative for building: in *(your libnabo folder)*/build:

    `$ msbuild /m:2 libnabo.sln`

    Note that the flag /m:X defines the number of core to use.

Building yaml-cpp

1. Start CMake Gui, follow the same building step
2. Change the variable **CMAKE_CONFIGURATION_TYPES** to `RelWithDebInfo`
3. Click on the button Configure, then on Generate
4. In visual Studio, build the solution: BUILD -> Build Solution

Command line alternative: in *(your yaml-cpp folder)*/build:

```
$ msbuild /m:2 YAML_CPP.sln
```

Note that the flag /m:X defines the number of core to use.

Building gtest

1. Open the file CMakeList.txt and add at the end:

   ```
   if( MSVC ) # VS2012 does not support tuples correctly yet
         add_definitions( /D _VARIADIC_MAX=10 )
   endif()
   ```
2. Start CMake-Gui to generate a MSVC solution
3. Change the variable **CMAKE_CONFIGURATION_TYPES** to `RelWithDebInfo`
4. Change the variable **gtest_force_shared_crt** to `TRUE`
5. Click on the button Configure, then on Generate
6. In visual Studio, build the solution: BUILD -> Build Solution

   Command line alternative: in *(your gtest folder)*/build:

   ```
   $ msbuild /m:2 gtest.sln
   ```

   Note that the flag /m:X defines the number of core to use.

Build libpointmatcher

1. Start CMake Gui
2. Follow the same building steps
3. Add the following boolean variable and set it to true: Boost_USE_STATIC_LIBS
4. Add the following PATH variable and set it to [boost folder]/build: BOOST_ROOT
5. Change the variable **CMAKE_CONFIGURATION_TYPES** to `RelWithDebInfo`
6. You will need to fill manually the following variables:

   1. **EIGEN_INCLUDE_DIR** to *(your eigen folder)*
   2. **NABO_INCLUDE_DIR** to *(your nabo folder)*
   3. **NABO_LIBRARY** to *(your nabo folder)*/build/RelWithDebInfo/nabo.lib
   4. **yaml-cpp_INCLUDE_DIRS**
   5. **yaml-cpp_LIBRARIES**
   6. **GTEST_INCLUDE_DIR**
   7. **GTEST_LIBRARY**
   8. **GTEST_MAIN_LIBRARY**

7. In visual Studio, build the solution: BUILD -> Build Solution

   Command line alternative: in *(your libpointmatcher folder)*/build:

   ```
   $ msbuild /m:2 libpointmatcher.sln
   ```

Note that the flag /m:X defines the number of core to use.

Reporting Issues

Currently, we don't have a developer fully supporting compilation on Windows. If you can help refreshing this documentation, your help is more than welcome.

Before reporting new building issues, have a look in the current/past list of issues. Add as much details as you can since you will most probably receive answers from developers that cannot reproduce the problem on their side.

Limited version of libpointmatcher in `C#`

The user braddodson ported a version of libpointmacher with a limited set of features here: https://github.com/braddodson/pointmatcher.net