Installation Guide
Bayesian Filtering Library

Tinne De Laet, Wim Meeussen, Erwin Aertbeliën and Klaas Gadeyne

March 13, 2020

Abstract

This document contains installation instructions for the Bayesian Filtering Library. Section **??** contains installation instructions for Linux, while Section **??** explains installation with Visual Studio for Windows.

1 Linux installation

This section contains installation instructions for BFL for Linux. The first Section **??** explains the most easy way to install, which is through the use of precompiled Ubuntu/Debian packages. For non Ubuntu/Debian users, the next Section **??** contains the instructions to install BFL from source.

1.1 Precompiled Ubuntu/Debian packages

For Ubuntu and Debian users, the easiest way to install BFL is by using precompiled packages. FIXME

1.2 Installation from source

An alternative way to get BFL is to install it from source, e.g. if you are not using Ubuntu or Debian. This section gives step-by-step instructions to compile BFL from source. You'll not only need to compile BFL, but also a matrix computation library and a random number generator library. A few things you need to know before getting started:

1. BFL is developed and regularly tested for GNU Linux on Debian and Ubuntu. However we've received success reports from other distributions and architectures, such as Gentoo, MAC OS X and MS Windows.
2. BFL compiles without warnings with g++ 2.95.0 up to g++ 4.1.2 However, it is preferred to use the latest g++ version.
3. We use cmake in the build system of BFL. You'll need cmake 2.4 or later.
4. To download the latest version from our subversion server, you will need a subversion client.

The instructions explain how to install libraries in the /install directory.

1.2.1 Install an external matrix library

BFL requires an external library for matrix computations. You can choose to use the eigen library, the Newmat library, or the Boost library.

Newmat matrix library You can install the Newmat library as a precompiled Ubuntu/Debian package, or compile it from source.

- To install the precompiled Newmat package run:

  ```
  sudo apt-get install libnewmat10-dev
  ```

- To install the Newmat library from source:

  1. Create a directory in which you will put the source of newmat:

     ```
     mkdir ~/sources/newmat/
     ```

2. Then download the latest version (version 11) of the Newmat library from the Newmat web page: `http://www.robertnz.net/download.html`
3. Move the downloaded file to your source folder:

   ```
   mv newmat11.tar.gz ~/sources/newmat
   ```

4. Go to the directory `~/sources/`

   ```
   cd ~/sources/newmat
   ```

5. Untar and unzip the file:

   ```
   tar -xzvf newmat11.tar.gz
   ```

6. Then go to `http://people.mech.kuleuven.be/~tdelaet/Makefile.Newmat` and save the file as Makefile in your newmat directory.
7. Also get the adapted include.h at `http://people.mech.kuleuven.be/~tdelaet/include.h.newmat11` and save the file as include.h. in your newmat directory.
8. Build the library:

   ```
   make
   ```

9. Finally, install the library. By default newmat gets installed in `/usr/local`. So change the *PREFIX* variable in `Makefile.Newmat` to `/install` or any other installation directory before running make install.

   ```
   [sudo] make install
   ```

Boost matrix library You can install the Boost library as a precompiled Ubuntu/Debian package, or install it from source.

- To install the precompiled Boost package run:

  ```
  sudo apt-get install libboost-dev
  ```

- To get the source of the Boost library from source, download it from `http://sourceforge.net/project/showfiles.php?group_id=7586`.

  1. Move the downloaded file to your source folder:

     ```
     mv boost_1_34_0.tar.gz ~/sources/
     ```

  2. Untar and unzip the file?

     ```
     tar -xzvf boost_1_34_0.tar.gz
     ```

  3. Go into the created directory:

     ```
     cd boost_1_34_o
     ```

  4. Configure the library:

     ```
     ./configure
     ```

  5. Make the package:

     ```
     make
     ```

  6. Install boost:

     ```
     sudo make install
     ```

1.2.2 Install an external random number generator library

BFL requires an external library for RNG computations. Currently only boost is supported.

Boost rng library You can install the Boost library as a precompiled Ubuntu/Debian package, or install it from source (follow the same instructions as described in section **??**). If you already installed Boost as the matrix library, you already have everything for the rng library.

1.2.3 Install BFL itself

Now we arrived to the installation of the BFL library itself. BFL is available from our subversion server, or as a tarbal.

1. First go to the `~/sources/` folder:

   ```
   cd ~/sources/
   ```

2. Then, to get a copy of BFL, use

   - For subversion access:

     ```
     svn co http://svn.mech.kuleuven.be/repos/orocos/trunk/bfl
     ```

   - To extract a BFL tarbal you got from the website:

     ```
     tar xvf orocos-bfl.tar.gz
     ```

3. Now create and go to a build directory in /sources/bfl

   ```
   mkdir ~/sources/bfl/build
   cd ~/sources/bfl/build
   ```

4. If you've installed one of the dependencies in a non-standard (where your compiler doesn't look by default) location, set the following environment variables to point to the appropriate directories (**Don't bluntly copy the statements below**)

   ```
   export CMAKE_INCLUDE_PATH=/path/to/installed/library/include
   export CMAKE_LIBRARY_PATH=/path/to/installed/library/lib
   ```

   See `http://www.cmake.org/Wiki/CMake_Useful_Variables#Environment_Variables` for more information

5. Run ccmake:

   ```
   ccmake ..
   ```

6. Then type 'c' to configure, and 'e' to exit the page that shows the configure output. Now you see the cmake configuration menu. In this menu you can change the following options:

   - BUILD_EXAMPLES: ON if building the examples is desired,
   - BUILD_TESTS: ON if building the cppunit tests is desired,
   - CMAKE_BUILD_TYPE: Debug or e.g. Release,
   - CMAKE_INSTALL_PREFIX: directory where to install bfl, e.g. `/install`,
   - GINAC_SUPPORT: OFF if you don't want to use GINAC support,
   - LIBRARY_TYPE: type of library build, shared or static,
   - MATRIX_LIB: matrix library used: boost, eigen or newmat,
   - RNG_LIB: random number generator used: boost

   Now repeat the 'c' configure and 'e' exit cycle, until you get no more errors. When this is the case, you'll have the 'g' generate option available. Press 'g' to generate the makefiles, and then quit cmake with 'q' quit.

7. Now all configuration is done, and you can build BFL:

   ```
   make
   ```

8. To check the functionality of BFL, use

   ```
   make check
   ```

9. To install BFL use

   ```
   sudo make install
   ```

In the `~/sources/sources/bfl/examples` directory, you find some example BFL filters. A good next step is to check out the BFL tutorial on `http://www.orocos.org/bfl`, for a step-by-step introduction in building your own filters in BFL.

Good luck!

2 Windows - Visual Studio installation

This section offers a step-by-step guide for BFL installation with Visual Studio on Windows.

1. Download the BFL source code from the website: `http://www.orocos.org/bfl/source`. Unzip this code.
2. Create a folder *build* and a folder *install* where you will respectively build and install bfl (what's in a name?).
3. Install boost
   - The boost Getting Started Guide gives excellent installation instructions for windows: `http://www.boost.org/more/getting_started/index.html`. In accordance with these instructions we advice to use the installer provided by Boost Consulting.
   - Change the install of boost such that the include files are located under: $c:\backslash boost\backslash$ Remark that the first part is arbirary, but the *include* part not; this will make the bfl installation easier.
4. Install CMake:
   - You can find cmake at `www.cmake.org`. Install it for windows.
5. Run CMake:
   - Go to the start menu to start the CMake-GUI.
   - Fill in the location of the source code e.g.:

     ```
     c:\Users\username\Desktop\orocos-bfl-0.6.0-pre1-src
     ```

   - Fill in the location of the build directory (which you previously created), e.g.:

     ```
     c:\Users\username\Desktop\orocos-bfl-0.6.0-pre1-src\build
     ```

   - Press the Configure botton. Select 'Visual Studio 8, 2005' as a generator. You well get a lot of 'errors' (remark this are actually not errors but messages). You can press the 'OK' botton to check all messages or just press the 'Cancel' botton to supress all further messages. Also mind the CPP-Unit error, which will prevent you from running the CPP-unit tests, but won't bother the examples nor any of your applications.
   - Fill in:
     - CMAKE_BUILD_TYPE: release
     - CMAKE_INSTALL_PREFIX: $c:\backslash Users\backslash username\backslash Desktop\backslash orocos-bfl-0.6.0-pre1-src\backslash build$
     - LIBRARY_TYPE: static
     - MATRIX_INSTALL: $c:\backslash boost\backslash$
     - MATRIX_LIB: boost
     - RNG_INSTALL: $c:\backslash boost\backslash$
     - RNG_LIB: boost
   - Press the Configure botton until you can press OK.
   - Press OK. CMake will shut down now, and BFL is ready to build. (If you get unexpected errors check the common problems section **??**).
6. Build BFL with Visual Studio
   - Start Visual Studio through the Start botton.
   - Open the project 'ALL_BUILD' located in the build directory.
   - Build the project (right mouse botton, build). (If you get unexpected errors check the common problems section).
   - Build the INSTALL project to install everything.

In the `~/sources/sources/bfl/examples` directory, you find some example BFL filters. A good next step is to check out the BFL tutorial on `http://www.orocos.org/bfl`, for a step-by-step introduction in building your own filters in BFL.

Good luck!

2.1 Common problems

2.1.1 Why does CMakeSetup with the message *LINK : fatal error LNK1104: cannot open file 'user32.lib'* while configuring a project?

The path to the SDK libs (user32.lib) must be added by the IDE when the project generator "Visual Studio 8 2005" is used, because cmake uses VCExpress.exe and on the fly generated project files to check for compiling (VCExpress.exe reads some config files for the compiler/linker options)

So add the sdk lib path at Tools→Options→Projects and Solutions→VC++ Directories→Library files So you have to install the platform SDK, (available on the Microsoft website `http://www.microsoft.com/downloads/details.aspx?FamilyId=A55B6B43-E24F-4EA3-A93E-40C0EC4F68E5&displaylang=en`) but normally you will already have done this. The installation will take some time.

2.1.2 CPP UNIT error

The CPPunit tests will not compile on windows due to the lack of the libcppunit library. (this will give an error message, you can ignore this message), the examples should work however. If you would like to get the cpp unit tests, you can check the platform specific build instructions on `http://cppunit.sourceforge.net/cppunit-wiki/`. Building the library is possible for e.g. Microsoft Visual Studio 2005 but not for Visual Studio 2005 Express.