RITK Installation Guide for Windows

January 27, 2012

The Range Imaging Toolkit (RITK) is a software framework for the development of range imaging (RI) applications. The toolkit is developed at the Pattern Recognition Lab, University Erlangen-Nuremberg, Germany.

Please send your feedback to this manual to ritk@i5.informatik.uni-erlangen.de

Contents

1 Setting up the Environment

1.1 Applications

Download and **install** the following applications that are required to install the libraries needed for the RITK (32 or 64-bit):

Name	Link	Comment
Visual Studio 2010 (*VS2010*) Express[1][2]	`http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-cpp-express`	Download and install Visual 2010 C++ Express (hereafter denoted as *VS2010*)
CMake	`http://www.cmake.org`	Download newest version

You can also use *Visual Studio 2008* if you already have it installed on your system. Any older version is not guaranteed to work with RITK.

Note that if you want to build the 64-bit version of RITK you will need the 64-bit version of Visual C++'s compiler `cl.exe` *which is* not *included in the Express edition. Either you install the Professional or Ultimate versions, or download a version of the Windows SDK that includes* `cl.exe` *from Microsoft's website.*

1.2 Libraries

Download and **install** the following applications or sources that are required to install the libraries needed for the RITK (32 or 64 bit):

Name	Link	Comment
Qt	`http://get.qt.nokia.com/qt/source/qt-everywhere-opensource-src-4.8.0.zip`	Compile newest version yourself or download pre-compiled binaries
CUDA 4.0	`http://developer.nvidia.com/cuda-toolkit-40`	Download and install only the *Toolkit*
Insight Toolkit (ITK)	`http://itk.org/ITK/resources/software.html`	Source Files Version 3.20 (will *not* work with ITK 4.0.0)
Visualization ToolKit (VTK)	`http://vtk.org/VTK/resources/software.html`	Source Files Version 5.8.0 or later

These libraries are fundamental to RITK and thus have to be installed. Therefore we have included instructions on how to compile and install them in the following sections. But first note that for some of the plug-ins to compile, you will furthermore need the libraries listed in the following table.

[1] You will need an emulation program to mount the VS2010 *.iso files. Use e.g. deamon tools

[2] via Microsoft's MSDNAA program you may also acquire the Professional or Ultimate versions if you are a student or university member

Name	Link	Comment
MS Kinect SDK[3]	`http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/download.aspx`	
OpenNI[3]	`https://github.com/avin2/SensorKinect`	Follow the installation guide here
libnoise	`http://libnoise.cvs.sourceforge.net/libnoise/`	download and compile the newest CVS sources

1.2.1 Qt Installation (4.8.0)

If you are compiling the 32-bit version of RITK probably the fastest and safest way to acquire Qt is by installing the appropriate binaries offered by Nokia. Just be sure that the binaries match your environment (e.g. in our case *Microsoft Visual Studio 2010*), otherwise RITK won't work.

There are a few reasons why you would want to compile Qt yourself: Firstly, Nokia does not provide 64-bit binaries, so you have to do it yourself. Secondly, you want more control over what Modules are built and installed; for example, you might want Qt to have less dependencies or use less disk space. Either way, follow these instructions to build Qt:

1. Unpack the source archive to a custom destination, let us say `<QT_DIR_PATH>`

2a. If you want to compile the 32-bit version of RITK, open the *VS2010 Command Prompt* from the Tools Menu or Windows Start Menu

2b. If you want to compile the 64-bit version of RITK, open the *VS2010 x64 Win64 Command Prompt* from the Windows Start Menu

3. Add `<QT_DIR_PATH>\bin` to the `PATH` variable via

```
set PATH=%PATH%;<QT_DIR_PATH>\bin
```

4. Go to `<QT_DIR_PATH>`, type

```
configure -platform win32-msvc2010
```

(this is for both builds; 64-bit version will build fine) and follow the instructions to configure Qt.

This will build a full Qt system which might take several hours. If you want to shorten the compile time and just want to use Qt for RITK, consider adding the following arguments to the `configure` command:

```
-nomake examples -nomake demos
```

for omitting all examples and demos; furthermore

```
-no-webkit -no-script -no-scripttools -no-qt3support
-no-phonon -no-phonon-backend -no-ltcg -no-multimedia
```

for omitting certain parts of Qt that are not needed by RITK.

[3]Please note that you have to decide either for installing OpenNI or MS Kinect SDK. In general, both will not work side by side in general.

5. Type `nmake` to build the desired version of Qt (this may take several hours, depending on your configuration in the previous step)

   If you have a multi-core CPU you can speed up the build process by using Nokia's `jom` instead of `nmake`.

6. To facilitate the following compile processes, you may add the new environment variable `QTDIR` to your system, which should point to <QT_DIR_PATH>(you can use an editor like Rapid Environment Editor or just go to System and Security in your Control Panel and then

   System → Advanced system settings → Advanced → Environment Variables...

   if you have Windows 7).

7. It is also advisable to include <QT_DIR_PATH>\bin permanently in your `PATH` variable; otherwise you will have to copy all needed Qt DLLs into your RITK binary directory.

1.2.2 CUDA Installation (4.0)

Install the CUDA Toolkit 4.0 with the setup wizard by using its default settings.

1.2.3 ITK Installation (3.20.1)

ITK does not provide any setup wizard but has to be compiled. The first step is to unzip the archive into a subfolder in your library folder (e.g. `.../libs/InsightToolkit-3.20.1/`). In this folder you should now find a file called CMakeLists.txt. This file is now used by CMake to create a *VS2010* Project:

1. Open *CMake* (`cmake-gui`).
2. Set the source path to the ITK root directory (`InsightToolkit-3.20.1/`).
3. Set the directory where the binaries will be built (usually a subfolder of the source folder, as we are using *VS2010* we call this folder `VC-10`).
4. Click *Configure*. If *CMake* asks whether to create the specified folder click *Yes*.
5. Now a window will pop up where you have to specify the generator. Select *Visual Studio 10* for the 32-bit or *Visual Studio 10 Win 64* for the 64-bit version and click *Finish*. CMake will now configure the project. This could take a while.
6. After configuring, CMake will show you some options (name and value) that depend on the actual project. For compiling ITK make sure that the option

   `BUILD_SHARED_LIBS` is enabled

   `BUILD_TESTING` and `BUILD_EXAMPLES` is disabled (for faster compilation).

7. Again, click *Configure*.
8. Click *Generate* (This will create a *VS2010*Solution `.sln` file).

Now we are ready to compile ITK.

In the binary folder (e.g. `.../libs/InsightToolkit-3.20.1/VC-10`) you should find a file `ITK.sln`, this is the *VS2010* Solution file. A double-click will open the solution. Build the project in `RELEASE` and/or `DEBUG` configuration. This may take a while.

1.2.4 VTK Installation (5.8.0)

The steps to build VTK are in general the same as for ITK: Extract files to a subfolder in your library folder, open `CMakeLists.txt` with *CMake* (make sure to update the source and binary paths), setting some options, generating a Visual Studio solution file and, finally, building the solution `VTK.sln` in `RELEASE` and/or `DEBUG` (debug built of Qt must be available) mode.
The following options have to be set:

1. Make sure that you enable `BUILD_SHARED_LIBS`.
2. Disable `BUILD_TESTING`, and `BUILD_EXAMPLE`, then click *Configure* (you may want to enable them but this takes really long to compile).
3. Switch to the *Grouped* view and expand the VTK section, enable `VTK_USE_PARALLEL` and `VTK_USE_QT`, then click on *Configure*.
4. Make sure that `QT_QMAKE_EXECUTABLE` under *QT* points to the correct Qt installation (`<QT_DIR_PATH>\bin\qmake.exe` if you followed our compilation guide)
5. Click on *Configure* and then *Generate* to create `VTK.sln` in your binary folder.

Building VTK is now the same as for ITK.

2 Setting up the RITK

In this section, we will first build the RITK application and then the packaged plug-ins. RITK heavily utilizes the plug-in architecture, so be sure to compile and install them.

2.1 Compiling the RITK

Now on to the main part:

1. Download the sources from `http://www5.cs.fau.de/ritk`.
2. Unzip the files to a custom destination, let us say `<RITK_SOURCE_DIR>`.
3. Open *CMake* (`cmake-gui`) and select

   `<RITK_SOURCE_DIR>/SurfaceImaging/RITK/v10/RITK` for the source directory

   `<RITK_SOURCE_DIR>/SurfaceImaging/RITK/v10/RITK/VC-10` for the build directory.
4. Now click *Configure* and set `CMAKE_INSTALL_PREFIX` to your needs, let us say `<RITK_BIN_DIR>` (default: `C:/Program Files/RITK` resp. `C:/Program Files (x86)/RITK`).
5. Again, click *Configure* and finally *Generate*.
6. Build RITK with *VS2010* (`DEBUG` if Qt, ITK and VTK debug builds are available, and `RELEASE`) in analogy to ITK/VTK .
7. In *VS2010*, right-click on the project "`INSTALL`" in the solution browser and choose "Project Only/Build only INSTALL..." to install RITK (you might need administrator privileges for this – right-click on *VS2010* in the Start Menu and choose "Run as administrator")
8. Copy all required dll files into the folder `<RITK_BIN_DIR>/bin/debug (release)`

   Copy all VTK DLLs:
   `.../libs/VTK-5.6.1/VC-10/bin/debug (release)`

   Copy the ITK DLL:
   `.../libs/InsightToolkit-3.20.1/VC-10/bin/debug (release)/ITKCommon.dll`

Now the RITK is ready to use, the application (debug/release versions) can be found in:

- Debug: `<RITK_BIN_DIR>/bin/debug/ritkd.exe`
- Release `<RITK_BIN_DIR>/bin/release/ritk.exe`

If there are any problems with starting RITK (e.g. error messages where some entry point is missing), please copy the DLLs for QT, too – in analogy to ITK/VTK. If problems persist, try using *DependencyWalker* to identify missing dependencies.

2.2 Setting up the Packaged Plug-ins

Compiling and installing the plug-ins follows a very similar procedure.

1. Open *CMake* and choose `RITKPlugins` as source directory, as well as an appropriate build directory.
2. Click on *Configure* and afterwards set the correct library directories as requested by *CMake*. (Hint: you can omit plug-ins from the build process by checking ”Advanced” and ”Grouped” and removing the mark from the plug-in’s name under RITK)
3. Also make sure that `RITK_INSTALL_DIR` points to the correct directory.
4. Finally, click *Configure* and *Generate*.
5. Open the generated solution and build it as usual.
6. Now build only the `INSTALL` project to install the plug-ins.

2.3 Testing the RITK

You can now test RITK for correct operation. Start up `ritk.exe` in your installation directory and press ’Ctrl+M’ to open the plug-in manager. If there is one missing, check with *DependencyWalker* whether all needed DLLs (e.g. `libnoise.dll`) are within reach (in a directory in the path or the directory `ritk.exe` is in).

If all plug-ins load fine, try to click on the body of the Visualization plug-in and press ’c’. A configuration bar should pop up; if it does, everything works. You might try to load data into the RISimulator plug-in to test for correct visualization. If nothing shows up, press ’r’ in order to reset the camera. Note that for the hot keys to work, the plug-in needs to be in focus.