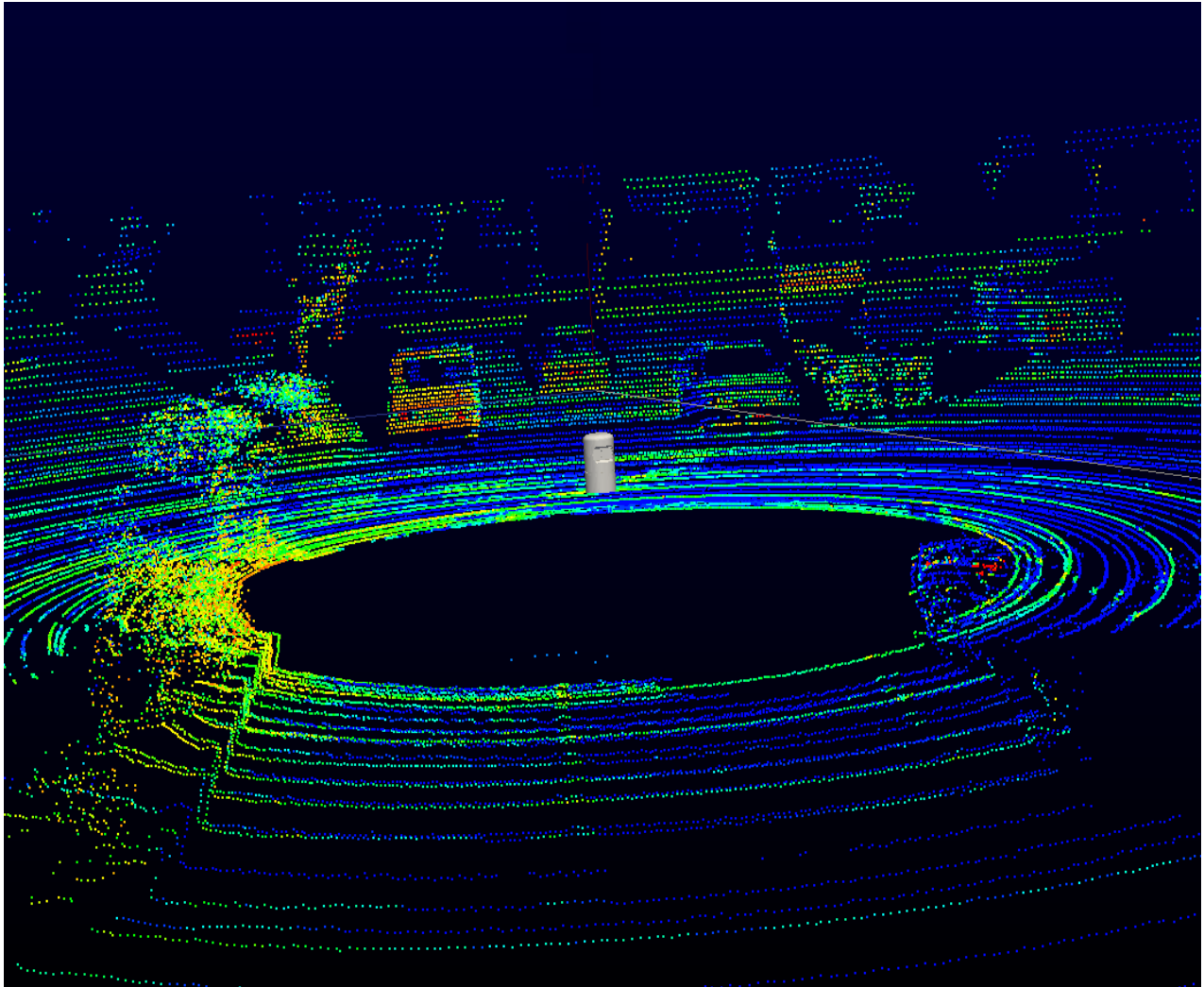


VeloView Developer Guide

Developer documentation for the VeloView application and libraries



Download: <https://www.paraview.org/VeloView>

Copyright: Copyright © 2017, Velodyne Lidar. All rights reserved

Version: 3.5.

Contents

I- Introduction	3
II- VeloView dependencies.....	3
A- PCAP	3
B- Boost.....	3
C- Qt	3
D- Python	3
E- PythonQt.....	3
F- Paraview (and VTK)	4
II- Configure and build instructions.....	4
A- Prebuild steps	4
A-1- Getting the source code	4
A-2- Superbuild Overview	4
A-3- Configure and build instructions	4
A-4 Windows build instructions	5
A-5- Ubuntu 16.04 build instructions.....	8

I- Introduction

VeloView is the software provided by Velodyne to visualize, analyze and record data from Velodyne-HDL sensors. The software performs real-time visualization and processing of live captured 3D LiDAR data from Velodyne's HDL sensors (HDL-64E, HDL-32E, VLP-32, VLP-16, Puck, Puck Lite, Puck HiRes). VeloView is able to playback pre-recorded data stored in .pcap files, and can record live stream as .pcap file. In the next sections of the manual it will be assumed that the reader is familiar with the Velodyne's sensors.

In a first part of the manual we will focus on the VeloView dependencies, what are their uses and why they are required. Then, we will explain how to build VeloView on Windows 10 and Ubuntu 16.04

II- VeloView dependencies

The VeloView application and libraries have several external library dependencies. As explained in the "Superbuild Overview" (see section ?), the dependencies will be downloaded and compiled automatically during the build step. See "Configure and build instructions".

A- PCAP

The required pcap version is 1.4. On Mac/Linux, only libpcap is required. On Windows, we use the Winpcap project which includes libpcap but also includes Windows specific drivers. Since the winpcap project only provides Visual Studio project files, which may be out dated, the superbuild does not attempt to compile winpcap. Instead, a git repository containing headers and precompiled .lib and .dll files is used. The repository url is <https://github.com/patmarion/winpcap>. Pcap is required to support saving captured packets to a file, and reading files containing saved packets.

B- Boost

The required boost version is 1.50. Boost is used for threading and synchronization, for network communication and to handle filesystem . The boost thread, asio and filesystem libraries are used.

C- Qt

The required Qt version is 4.8. Qt is a desktop widget library that is used to provide user interface elements like windows and menus across the supported platforms Windows, Mac, and Linux.

D- Python

The required Python version is 2.7. VeloView uses libpython to embed a Python interpreter in the VeloView application. The core VeloView features are implemented in C++ libraries, and the libraries are wrapped for Python using VTK's Python wrapping tools.

E- PythonQt

PythonQt is used from a specific git commit sha1, but a specific released version. PythonQt is used to build Qt applications using Python. PythonQt has support for wrapping types derived from Qt objects and VTK objects.

F- Paraview (and VTK)

The required VTK version is 6.0. The required ParaView version is 4.0. The ParaView repository includes VTK, so the superbuild only needs to checkout and build ParaView in order to satisfy both dependencies. A specific git commit sha1 is used instead of a specific released version. The commit sha1 is very similar to the version 4.0 release but it has a few commits from the ParaView master branch cherry-picked onto it. The commits added are those that resolve some issues with the Python console and add the PythonQtPlugin for ParaView. The PythonQtPlugin is a small plugin that initializes the PythonQt library and makes it available in the ParaView Python console.

II- Configure and build instructions

A- Prebuild steps

A-1- Getting the source code

The VeloView software is hosted in a git repository on github.com Use the following command line to checkout the source code:

```
git clone git://public.kitware.com/VeloView.git
```

A-2- Superbuild Overview

VeloView uses a cmake *superbuild* to download and compile third party projects that are dependencies of VeloView. The superbuild will give you the option to use system installations of third party projects instead of compiling them as a superbuild step. Some dependencies, on certain platforms, must be compiled by the superbuild, and there is no option to use a system version.

A-3- Configure and build instructions

The superbuild requires cmake version 2.8.8. The build will be performed in a separate, out-of-source build directory. Start a terminal in the parent directory that contains your veloview checkout. Create a new build directory and then use cmake to configure the superbuild:

```
mkdir build
cd build
cmake -DENABLE_veloview:BOOL=ON ../veloview/SuperBuild
```

This will configure the veloview superbuild with the VeloView application enabled, and all other options left at default. For most builds, the default options are satisfactory. Some users may wish to configure custom cmake options in order to select system versions of certain third party dependencies such as Qt, Boost, or Python. You should not use a system version of pcap, instead, let the superbuild checkout the correct version of pcap or winpcap.

You can use cmake, ccmake (a curses interface to cmake) or cmake-gui. You can set the CMAKE_BUILD_TYPE to Release, Debug, or RelWithDebInfo to set the build type. Most users will want to select Release. You can set the CMAKE_INSTALL_PREFIX to determine where the VeloView binaries are installed when you run make install. After cmake has generated the build files, just run make to run the superbuild:

```
make
```

On Mac and Linux computers, you can run parallel make with `*make -j*`. Parallel make is not supported on Windows because the Windows build uses NMake.

A-4 Windows build instructions

Tools required

- CMake > 2.8.8
- Git
- Visual Studio 2012 (VS12 = MSVC 11)

Download sources

Create a VeloView folder and clone the sources: `git clone git://public.kitware.com/VeloView.git`

Running CMake

Create a build folder next to the sources folder (see Fig1).

Name	Date modified	Type
 build	6/26/2018 12:10 PM	File folder
 VeloView-kwinternal	6/26/2018 12:09 PM	File folder

Fig1. Filesystem recommended

Open a “VS2012 x64 Cross Tools Command Prompt” and run `cmake-gui` from this command prompt (see Fig2)

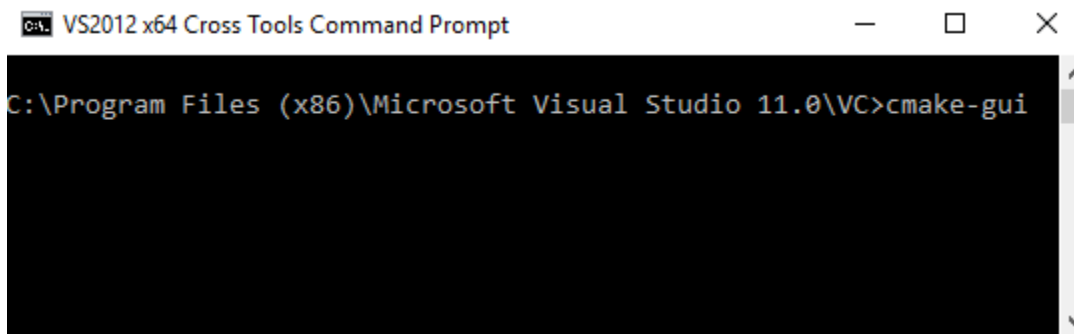


Fig2. Running cmake from VS2012 Cross Tools Prompt

It is important to launch `cmake-gui` from this specific command prompt so that some CMake variables are correctly set.

Once CMake is launched you must indicate the project source code and the build folder to create binaries (see Fig3). It is up to you to select a build folder, we recommend to create a build folder next to the sources folder (see Fig1). The source code folder is the VeloView Superbuild folder (see Fig3).

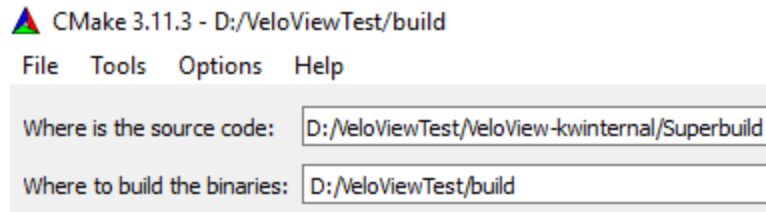


Fig3. Project path to source / build folder

Once the paths to the project code source and build folder are settled you can click on the configure button. A window will pop-up to select the compiler and project generator. Here you must choose Visual Studio 11 2012 (win64 or win32) since it is the only compiler that can build VeloView on windows for now (see Fig4).

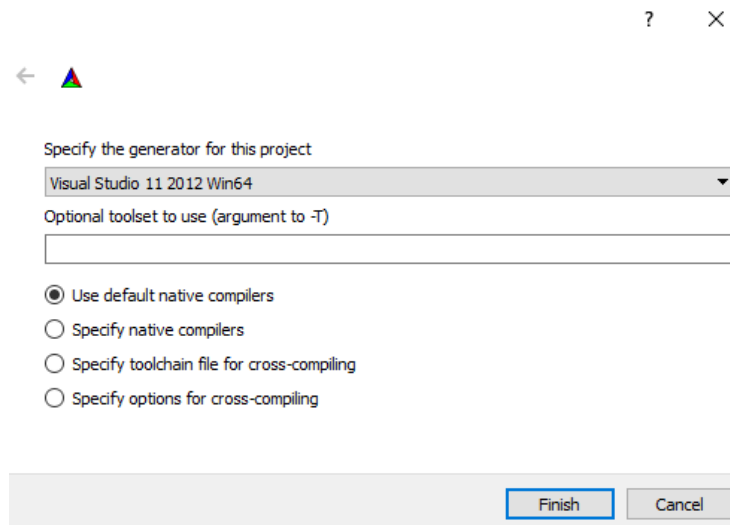


Fig4. Compiler selection

Then, you can click on the finish button. CMake will configure the VeloView project and show you the list of CMake variables. Here it is important to enable the VeloView project and all its dependencies (see Fig5). To do that check the variables: ENABLE_boost, ENABLE_eigen, ENABLE_liblas, ENABLE_paraview, ENABLE_pcap, ENABLE_pyhon, ENABLE_pythonqt, ENABLE_qt, ENABLE_veloview

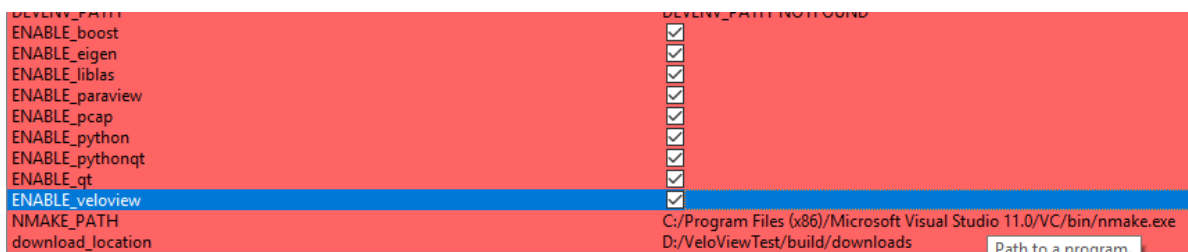


Fig5. Enable VeloView and its dependencies

Once all the dependencies and VeloView are enabled you can click on “configure” and then “Generate”.

Building VeloView

Once the project has been generated using CMake, a Visual Studio solution "VeloviewSuperbuild.sln" should be located in the build folder (see Fig6).




	VeloViewSuperBuild.sdf	6/26/2018 2:05 PM	SQL Server Comp...	448 KB
	VeloViewSuperBuild.sln	6/26/2018 2:02 PM	Microsoft Visual S...	14 KB
	ZERO_CHECK.vcxproj	6/26/2018 2:02 PM	VC++ Project	28 KB

Fig6. VeloViewSuperBuild.sln solution

Once the VeloViewSuperBuild project is loaded change the build configuration to Release (see Fig7). Veloview can't be built in Debug mode on Windows. To build a project in debug mode on windows it requires that all dependencies are provided in debug mode too. But, since we use a prebuild version of winpcap which is in Release mode VeloView cannot be built in debug mode.

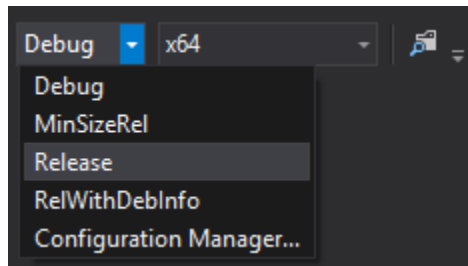


Fig7. Set Visual Studio Release mode

Before building VeloView and all its dependencies you must build eigen. To do so, right click on eigen project, select Project Only menu and click on Build Only eigen (see Fig8)

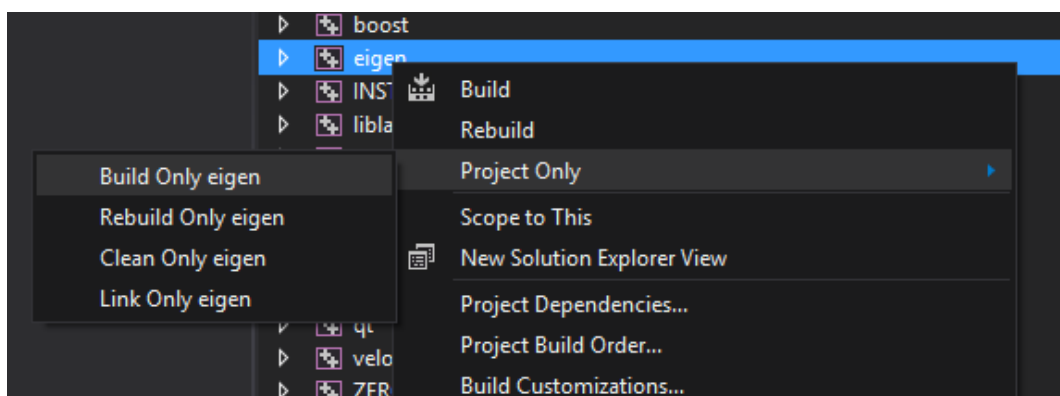


Fig8. Build Only Eigen

An error message "error : Generator: execution of make failed. Make command was: "" "Eigen.sln" "/build" "Release" "/project" "ALL_BUILD" D:\VeloViewTest\build\CUSTOMBUILD eigen" should appear.

Then, go on BuildFolder > eigen > src > eigen-build and load Eigen.sln solution. Once the eigen project is

loaded you will need to change the project configuration to Release (see Fig7). Finally right click on INSTALL project and click on Build (see Fig8).

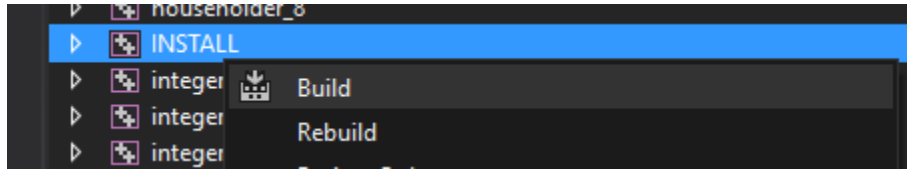


Fig8. Install eigen

To complete the VeloView building, you must go back in the VeloviewSuperBuild.sln project and build the ALL_BUILD project. The build will take ~2-3 hours.

Day to Day development

If you want to modify VeloView source code and to build only VeloView without rebuilding its dependencies you should load the VeloView.sln solution located in BuilFolder > veloview > src > veloview-build. Once again you will have to change the build configuration to Release mode. Once the modifications have been done you can build and install veloview by building the install project.

A-5- Ubuntu 16.04 build instructions

Tools required

- CMake > 2.8.8
- Git

Download sources

Create a VeloView folder and clone the sources: `git clone git://public.kitware.com/VeloView.git`

Running CMake

Create a build folder next to the sources folder (see Fig1). Launch cmake-gui. Once CMake is launched you must indicate the project source code and the build folder to create binaries (see Fig9). It is up to you to select a build folder, we recommend to create a build folder next to the sources folder (see Fig1). The source code folder is the VeloView Superbuild folder (see Fig9).

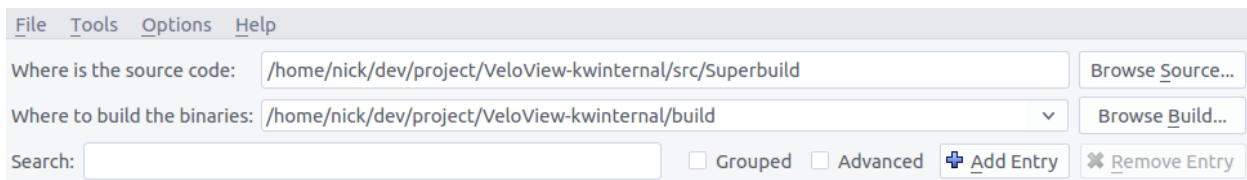


Fig9. Project path to source / build folder

Once the paths to the project code source and build folder are settled you can click on the configure button. A window will pop-up to select the compiler and and project generator. Here you must choose Unix Makefiles (see Fig10).

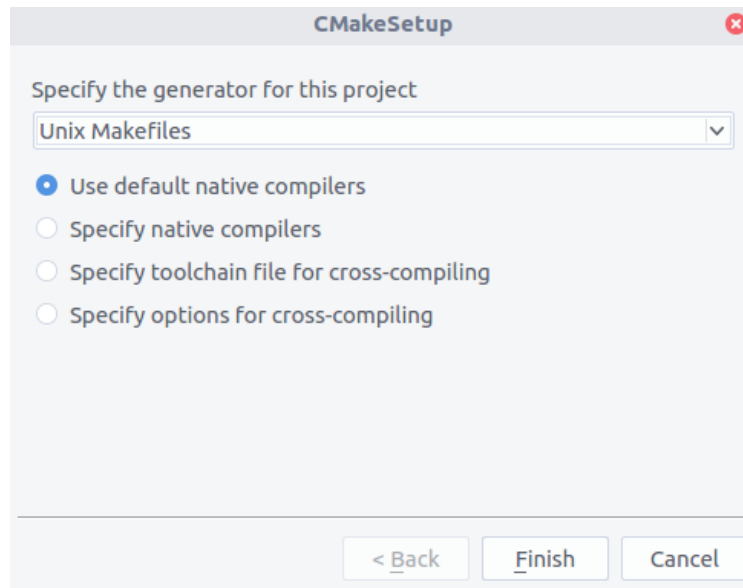


Fig 10. Compiler selection

Then, you can click on the finish button. CMake will configure the VeloView project and show you the list of CMake variables. Here it is important to enable the VeloView project and all its dependencies (see Fig11). To do that check the variables: `ENABLE_boost`, `ENABLE_eigen`, `ENABLE_libblas`, `ENABLE_paraview`, `ENABLE_pcap`, `ENABLE_pyhon`, `ENABLE_pythonqt`, `ENABLE_qt`, `ENABLE_veloview`



Fig11. Enable VeloView and its dependencies

Once the dependencies enabled click on Configure. An error message indicating that CMake cannot find `zlib` should pop. Click on `USE_SYSTEM_python` and click on Configure again and then generate.

Building VeloView

Go on the build folder, open a terminal a type "make -j8".