

Stella-vslam installation manual for WSL

Miłosz Wojciechowski

August 8, 2023
v0.1

Contents

1	Environment preparation	1
1.1	Install WSL on your windows 10 system	1
1.2	Prepare Ubuntu system	1
1.3	Install CMake	4
1.4	Install vcpkg	4
2	Install libraries required by stella-vslam	7
2.1	Eigen	7
2.2	g2o	7
2.3	Yaml-cpp	7
2.4	OpenCV	8
2.5	Pangolin	8
2.6	FBoW	8
3	Stella-vslam installation	11
3.1	Build basic library	11
3.2	Build with support for PangolinViewer	12
3.3	Build examples	13

Chapter 1

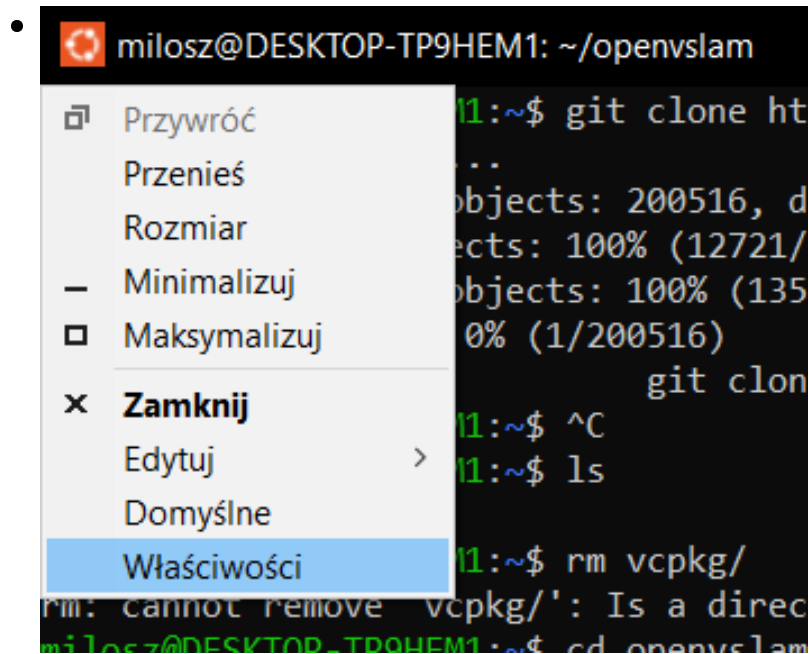
Environment preparation

1.1 Install WSL on your windows 10 system

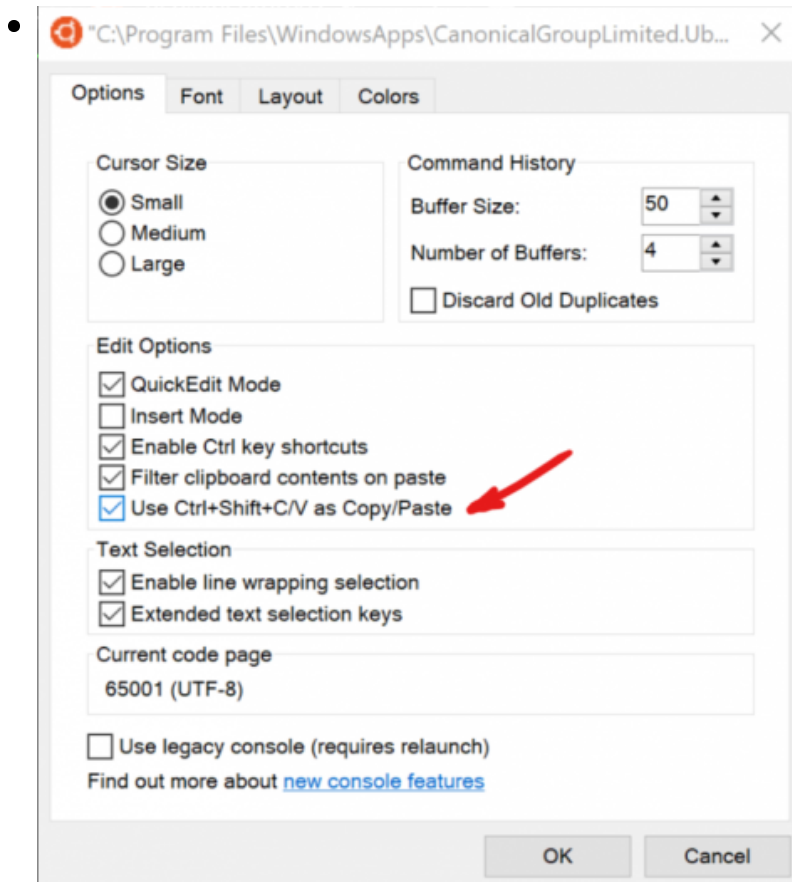
- Open windows command prompt by typing cmd in start menu
- Write `wsl --install` (by default it will install the Ubuntu distribution of Linux, if you want to change that visit [this site](#))
- Just follow instructions displaying on the console, computer restart might be needed
- If you run into an issue during install process check [this site](#)

1.2 Prepare Ubuntu system

- type your username and password (remember it, since you will need this to work on this system)
- To turn copy/paste on just click on the Ubuntu icon in the left upper corner of the console



Go to Properties.



Tick "Use Ctrl+Shift+C/V as Copy/Paste" option

- Now search and install system updates by using commands:

```
sudo apt update -y
```

```
sudo apt upgrade -y --no-install-recommends
```

```
sudo apt install -y build-essential pkg-config git wget
```

```
sudo apt install -y sqlite3 libsqlite3-dev
```

```
sudo apt-get install binutils-dev
```

```
sudo apt-get install libdwarf-dev
```

- If you want to search WSL files with Windows File Explorer just type following path in your File Explorer:

```
\\wsl$
```

1.3 Install CMake

- For safety uninstall CMake version if it was installed by default on Ubuntu:

```
sudo apt purge --auto-remove cmake
```

- Install CMake:

```
sudo apt-get install cmake
```

1.4 Install vcpkg

- First install zip and tar package:

```
sudo apt-get install curl zip unzip tar
```

- Download vcpkg from GitHub:

```
wget -O vcpkg.tar.gz https://github.com/microsoft/vcpkg/archive/master.tar.gz
```

- Create a new directory for vcpkg and unpack downloaded file there:

```
sudo mkdir /opt/vcpkg
```

```
sudo tar xf vcpkg.tar.gz --strip-components=1 -C /opt/vcpkg
```

- Now build vcpkg:

```
sudo /opt/vcpkg/bootstrap-vcpkg.sh
```

- Make a symbolic link to the vcpkg command in “/usr/local/bin”:

```
sudo ln -s /opt/vcpkg/vcpkg /usr/local/bin/vcpkg
```

- If you want to check the version of installed vcpkg use the following command:

```
vcpkg version
```

- Lastly remove file downloaded from GitHub:

```
rm -rf vcpkg.tar.gz
```


Chapter 2

Install libraries required by stella-vslam

2.1 Eigen

- To install Eigen use the following command:

```
sudo vcpkg install eigen3:x64-linux
```

2.2 g2o

- First install Fortran, to do this use the command:

```
sudo apt-get install gfortran
```

- Now install g2o, x64-linux is added to be sure we are downloading 64-bit version of the library. g2o has many dependencies so installation may take a while:

```
sudo vcpkg install g2o:x64-linux
```

2.3 Yaml-cpp

- To install yaml-cpp use:

```
sudo vcpkg install yaml-cpp:x64-linux
```

2.4 OpenCV

- First install bison utility:

```
sudo apt-get update  
sudo apt-get install bison
```

- Now install libgtk2.0-dev:

```
sudo apt-get install libgtk2.0-dev
```

- Install OpenCV3:

```
sudo vcpkg install opencv3:x64-linux
```

2.5 Pangolin

- Install nasm:

```
sudo apt-get install nasm
```

- Install Glew:

```
sudo apt-get install libglew-dev
```

- Install Pangolin:

```
sudo vcpkg install pangolin:x64-linux
```

2.6 FBoW

- Use the following commands to download and build the custom version of the library:

```
cd /tmp  
git clone https://github.com/stella-cv/FBoW.git  
cd FBoW
```

```
mkdir build && cd build
cmake \
  -DCMAKE_BUILD_TYPE=Release \
  -DCMAKE_INSTALL_PREFIX=/usr/local \
  -S ../ -DBUILD_TESTS=ON -DBUILD_UTILS=ON \
  -DCMAKE_TOOLCHAIN_FILE=/opt/vcpkg/scripts/buildsystems/vcpkg.cmake
make -j4 && sudo make install
```


Chapter 3

Stella-vslam installation

3.1 Build basic library

- `mkdir -p ~/lib`
- `cd ~/lib`
- `git clone --recursive https://github.com/stella-cv/stella_vslam.git`
- `mkdir build && cd build`
- `cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo -S ../stella_vslam \`
`-DCMAKE_TOOLCHAIN_FILE=/opt/vcpkg/scripts/buildsystems/vcpkg.cmake`

- **SKIP TO MAKE -J4 IF CMAKE COMMAND WAS SUCCESSFUL**

If previous command gives you error that some library installed via vcpkg wasn't detected perform the following step:

Open CMakeLists.txt in `~/lib/stella_vslam` directory and paste the following line before `find_package()`

```
include(/opt/vcpkg/scripts/buildsystems/vcpkg.cmake)
```

```
64 # CMakeLists.txt
65 # C4181: DLL export, C4144: floating-integer, C4309: double-float, C4267: size_t->any, C4127: constant condition
66 add_compile_options(/W $If(<CONFIG:Debug>;/MT:$If(<CONFIG:Release>;/MT /source-charset:utf-8 /execution-charset:utf-8 /w5251 /ws2466 /wd3050 /w
67 add_compile_options("/Zc:wchar_t;/Zc:stdC++;/Zc:stdC++lib")
68 else()
69     add_compile_options(-Wall -Wextra)
70     add_compile_options(
71         "$<<CONFIG:Debug>;-Og"
72         "$<<CONFIG:Release>;-Os"
73         "$<<CONFIG:RelWithDebInfo>;-O3"
74     )
75 endif()
76
77 set(BUILD_WITH_MARCH_NATIVE OFF CACHE BOOL "Enable architecture-aware optimization")
78 if((BUILD_WITH_MARCH_NATIVE AND NOT Msvc))
79     add_compile_options(-march=native -xarch=native)
80     message(STATUS "Architecture-aware optimization: ENABLED")
81 else()
82     message(STATUS "Architecture-aware optimization: DISABLED")
83 endif()
84
85 # ---- Find dependencies ----
86 include($ENV{CPM}/scripts/buildsystems/cpm.cmake)
87
88 # Threads
89 find_package(Threads REQUIRED)
90
91 # OpenMP
92 find_package(OpenMP REQUIRED)
93 if(NOT TARGET OpenMP::OpenMP_CXX)
94     add_library(OpenMP::OpenMP_CXX IMPORTED INTERFACE)
95     set_property(TARGET OpenMP::OpenMP_CXX
96         PROPERTY INTERFACE_LINK_LIBRARIES $If(OpenMP_CXX_FLAGS) Threads:Threads)
97 endif()
98
99 # Eigen
100 find_package(Eigen3 3.3 REQUIRED)
101
102 # yaml-cpp
103 find_package(yaml-cpp REQUIRED)
```

It should look like this.

Then run command:

```
cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo -S ../stella_vslam
```

- make -j4
- sudo make install

3.2 Build with support for PangolinViewer

- `cd ~/lib`
- `git clone -b 0.0.1 --recursive https://github.com/stella-cv/pangolin_viewer.git`
- `mkdir -p pangolin_viewer/build`
- `cd pangolin_viewer/build`
- `cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo -S ../ \`
`-DCMAKE_TOOLCHAIN_FILE=/opt/vcpkg/scripts/buildsystems/vcpkg.cmake`
- **SKIP TO MAKE -J IF CMAKE COMMAND WAS SUCCESSFUL**

If previous command gives you error that some library installed via vcpkg wasn't detected perform the following step:

Open CMakeLists.txt in lib/pangolin_viewer directory and paste the following line before find_package()

```
include(/opt/vcpkg/scripts/buildsystems/vcpkg.cmake)
```

Then run command:

```
cmake -DCMAKE_BUILD_TYPE=RelWithDebInfo -S ../
```

- `make -j`
- `sudo make install`

3.3 Build examples

- Install Backward-cpp library:

```
sudo apt-get install libbackward-cpp-dev
```

- `cd ~/lib`
- `git clone -b 0.0.1 --recursive https://github.com/stella-cv/stella_vslam_examples.git`
- `mkdir -p stella_vslam_examples/build`
- `cd stella_vslam_examples/build`
- `cmake \`
`-DCMAKE_BUILD_TYPE=RelWithDebInfo \`
`-DUSE_STACK_TRACE_LOGGER=ON \`
`-S ../ -DCMAKE_TOOLCHAIN_FILE=/opt/vcpkg/scripts/buildsystems/vcpkg.cmake`

- **SKIP TO MAKE -J IF CMAKE COMMAND WAS SUCCESSFUL**

If previous command gives you error that some library installed via vcpkg wasn't detected perform the following step:

Open CMakeLists.txt in `lib/stella_vslam_examples` directory and paste the following line before `find_package()`

```
include(/opt/vcpkg/scripts/buildsystems/vcpkg.cmake)
```

Then run command:

```
cmake \
-D CMAKE_BUILD_TYPE=RelWithDebInfo \
-D USE_STACK_TRACE_LOGGER=ON \
-S ../
```

- `make -j`
- After building check if everything was successfully built:

```
./run_kitti_slam -h
```