

UNIVERSITEIT  
VAN  
AMSTERDAM

IRL technical report IRL-UVA-24-02

## VisualSfM - Ubuntu installation

**Arnoud Visser**

IRL

Intelligent Robotics Lab

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Installation steps</b>	<b>1</b>
2.1	Getting Started . . . . .	1
2.2	Setup NVIDIA CUDA (Optional) . . . . .	1
2.3	Download the Necessary Software . . . . .	1
2.4	Install Dependency Packages . . . . .	1
2.5	Install Legacy Dependency Packages . . . . .	2
2.6	Install VisualSfM . . . . .	2
2.7	Build SiftGPU . . . . .	3
2.8	Build Multicore Bundle Adjustment (a.k.a. “pba”) . . . . .	3
2.9	Hack PMVS-2 . . . . .	3
2.10	Build Graclus 1.2 . . . . .	4
2.11	Hack CMVS . . . . .	4
<b>3</b>	<b>Running VisualSfM</b>	<b>5</b>
3.1	Add some images . . . . .	5
3.2	Match the images . . . . .	6
3.3	Sparse reconstruction . . . . .	6
3.4	Dense reconstruction . . . . .	6
<b>4</b>	<b>Conclusion</b>	<b>7</b>

**Intelligent Robotics Lab**  
Informatics Institute, Faculty of Science  
University of Amsterdam  
Science Park 900, 1098XH Amsterdam  
The Netherlands  
[www.intelligentroboticslab.nl](http://www.intelligentroboticslab.nl)

# 1 Introduction

In 2013 Changchang Wu, who worked on the Structure from Motion (SfM) part of the study 'Building Rome on a Cloudless Day' [1], created the graphical tool **VisualSfM**<sup>1</sup>, which can be used for 3D reconstruction. The VisualSfM graphical tool combines several algorithms, for feature detection [3], feature matching [6], bundle adjustment [5], sparse & dense reconstruction [4].

For Windows those tools are bundled up, for Ubuntu more installation work has to be done. An installation tutorial for Ubuntu is given<sup>2</sup>, yet this tutorial is based on the Long Term Support version of 2012 (Precise 12.04). This document gives the instructions needed to build it for modern versions (tested on Ubuntu 20.04 and 22.04).

## 2 Installation steps

To get the system working, six tools have to be built. This guide follows the steps from Scott Sawyer's tutorial, including the necessary modifications.

### 2.1 Getting Started

Scott was using working directly in his home-directory, `~/vsfm`, with several of the other tools installed in that directory. I preferred to work in `~/src/vsfm`, with the other tools build next to `~/src/vsfm`.

### 2.2 Setup NVIDIA CUDA (Optional)

I skipped this part, partly because not all laptops will have a NVIDIA GPU, partly because I am afraid of the legacy issues encountered when using CUDA-code from 2012. Our experience is that the program is fast enough to run on modern laptops without additional GPU.

### 2.3 Download the Necessary Software

These are 64bits versions of the six tools plus one support routine that are combined in the VisualSfM graphical user interface:

- **[VisualSfM:]** version v0.5.26
- **[SiftGPU:]** version v400
- **[Multicore Bundle Adjustment:]** version 1.0.5
- **[PMVS-2:]** fix0
- **[CLAPACK:]** version 3.2.1
- **[Graclus:]** v1.2
- **[CMVS:]** fix2

In the next sections we assume that you saved these downloads in your `~/Downloads/` directory.

### 2.4 Install Dependency Packages

Not all dependencies mentioned by Scott are still available. Instead, I used the following set:

```
$ sudo apt-get install libgtk2.0-dev libglew-dev glew-utils libdevil-dev libboost-all-dev libatlas-cpp-0.6-dev imagemagick libatlas-base-dev libcmminpack-dev libmetis-edf-dev libparmetis-dev freeglut3-dev libgsl-dev libcanberra-gtk-module dos2unix
```

---

<sup>1</sup><http://ccwu.me/vsfm/>

<sup>2</sup><http://www.10flow.com/2012/08/15/building-visualsfm-on-ubuntu-12-04-precise-pangolin-desktop-64-bit/>

## 2.5 Install Legacy Dependency Packages

The PVMS-2 tool is Fortran based, which is an old programming language, which needs some libraries from 2008. These libraries are no longer in the default Ubuntu archive, but can still be accessed via `old-releases.ubuntu.com`.

To get this to work, add to `/etc/apt/sources.list` the following lines.

```
deb [allow-insecure=yes] http://old-releases.ubuntu.com/ubuntu/ hardy main universe
deb [allow-insecure=yes] http://archive.ubuntu.com/ubuntu/ bionic main universe
```

To load this repository do `sudo apt-get update`

Next, you should install the old version of the compiler with `sudo apt-get install g++-4.1` and `sudo apt-get install g++-5`. To activate these compiler you have to check which g++ versions you have with the command `ls /usr/bin/g++-*`.

In my case I had four versions of the compiler, which you can register as alternative. The last integer in each of these commands indicates the priority for this alternative, I used the *year* – 2000 as value for this priority.

```
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.1 8
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 15
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-9 19
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-11 21
```

Now you can select the current gcc-compiler interactively for `sudo update-alternatives --config g++`. The same can be done for gcc:

```
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.1 8
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 15
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-9 19
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 21
```

Select gcc-4.1 and g++-4.1 with `sudo update-alternatives --config`:

```
$ sudo update-alternatives --config gcc
$ sudo update-alternatives --config g++
```

Now you can do `sudo apt-get install libgfortran1`

When `libgfortran1` is correctly installed, you can comment out the two new `ubuntu.com` old-archive lines from your `/etc/apt/sources.list`, to prevent problems later.

Switch back again to the default gcc-11 and g++-11 with `sudo update-alternatives --config`.

## 2.6 Install VisualSfM

Building VisualSfM from source, so we will use a prebuild version. Luckily Alberto Mardegan has a prebuild version:

```
$ cd ~/src
$ unzip ~/Downloads/VisualSfM_linux_64bit.zip
```

This installs the framework with all its subdirectories in `/src/vsfm`, only the executable is missing.

```
$ snap install visualsfm-mardy
$ cp -p /snap/visualsfm-mardy/current/bin/VisualSfM ~/src/vsfm/bin
```

We cannot use the snap-version directly, because it depends on an older `libgs1.so` library.

## 2.7 Build SiftGPU

In this guide we have chosen to circumvent CUDA installation issues, we build SiftGPU without CUDA support. CUDA support is per default on, so the makefile has to be adjusted.

```
$ cd ~/src
$ unzip ~/Downloads/SiftGPU.zip
```

Open the makefile and set `siftgpu_enable_cuda=0`. Although you did already `sudo apt-get install libdevil-dev` in Section 2.4, you can also set `siftgpu_disable_devil = 1`.

Now everything should be ready for a build.

```
$ cd ~/src/SiftGPU
$ make
$ cp bin/libsiftgpu.so ../vsfm/bin
```

## 2.8 Build Multicore Bundle Adjustment (a.k.a. “pba”)

Again we build the version without CUDA support. To build this tool we need to make some minor adjustments in the header files.

```
$ cd ~/src
$ unzip ~/Downloads/pba_v1.0.5.zip
```

Scott suggested to add a line `#include <stdlib.h>` to two files: `SparseBundleCU.h` and `pba.h`, which both can be found in the directory `pba/src/pba`. During our testing it also worked without this addition. You can make both files better readable with the command `dos2unix`.

```
$ cd ~/src/pba
$ make -f makefile_no_gpu
$ cp bin/libpba_no_gpu.so ../vsfm/bin/libpba.so
```

## 2.9 Hack PMVS-2

Scott recommends a hack to temporary save `mylapack.o` before building from source, but this fails because this file was compiled with an outdated option. Instead we will also build `clapack` from source.

To do that, we need the definitions of `clapack`.

```
$ cd ~/src
$ tar -xvf ~/Downloads/pmvs-2-fix0.tar.gz
$ cd ~/src/pmvs-2/program/base/numeric/
$ tar -xvf ~/Downloads/clapack.tgz
```

```
$ cd ~/src/pmvs-2/program/main/
$ mkdir include
$ cd ~/src/pmvs-2/program/main/include
$ ln -s ~/src/pmvs-2/program/base/numeric/CLAPACK-3.2.1/INCLUDE clapack
$ cd ~/src/pmvs-2/program/main/
```

Now edit in the main-directory the Makefile, and add the new include-directory to the compiler with the line `YOURINCLUDEPATH = -I include`. Now you should be able to build the executable `pmvs2`.

```
$ make clean
$ make depend
$ make
$ cp pmvs2 ~/src/vsfm/bin
```

## 2.10 Build Graclus 1.2

The Graclus is per default configured for 32-bits computers, so only a small modification is needed to build this tool.

```
$ cd ~/src
$ tar -xvf ~/Downloads/graclus1.2.tar.gz
```

Edit `Makefile.in` and set `-DNUMBITS=64`. That is enough to build this tool.

```
$ cd ~/src/graclus1.2
$ make
$ cp graclus ../vsfm/bin
```

## 2.11 Hack CMVS

To build CMVS, we need some of the previous build libraries.

```
$ cd ~/src
$ tar -xvf ~/Downloads/cmvs-fix2.tar.gz
```

You need to modify two source files. Add two lines (`#include <vector>` and `#include <numeric>`) to `~/src/cmvs/program/base/cmvs/bundle.cc`. Add one line (`#include <stdlib.h>`) to `~/src/cmvs/program/main/genOption.cc`.

Then modify `~/src/cmvs/program/main/Makefile` with the following lines 10-17 (don't forget to comment out line 10, 13, 16):

```
#Your INCLUDE path (e.g., -I/usr/include)
YOUR_INCLUDE_PATH = -I/home/yourname/src/pmvs-2/program/main/include

#Your metis directory (contains header files under graclus1.2/metisLib/)
YOUR_INCLUDE_METIS_PATH = -I/home/yourname/src/graclus1.2/metisLib

#Your LDLIBRARY path (e.g., -L/usr/lib)
YOUR_LDLIB_PATH = -L/home/yourname/src/graclus1.2
```

This tool will not compile with the default compiler, so you should change your compiler to `gcc-5.5` and `g++-5.5` with:

```
sudo update-alternatives --config gcc
sudo update-alternatives --config g++
```

Now you can start the build

```
$ cd ~/src/cmvs/program/main
$ make
$ cp cmvs ~/src/vsfm/bin
$ cp pmvs2 ~/src/vsfm/bin
$ cp genOption ~/src/vsfm/bin
```

Don't forget to switch back to the default `gcc-11` and `g++-11` again with `sudo update-alternatives --config`.

### 3 Running VisualSFM

Finally, add the executable VisualSFM to your path by adding the following lines to your `~/.bashrc` file.

```
export PATH=$PATH:/home/yourname/src/vsfm/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/home/yourname/src/vsfm/bin
```

Activate these extensions with the command `source ~/.bashrc`. Now you should be able to start the graphical tool with the command `VisualSFM`, or run without graphics with a command like `VisualSFM sfm+pmvs ./images ./result.nvm`, as indicated in the documentation<sup>3</sup>.

As an example, you can download a small dataset of 119 images recorded with the front camera of a RAE robot driving less than a meter in a small maze.

```
$ cd /home/yourname/src/vsfm/
$ mkdir results
$ mkdir -p data/small_maze
$ cd data/small_maze
$ wget https://staff.fnwi.uva.nl/a.visser/education/VAR/2024/small_maze.tgz
$ tar -zxvf small_maze.tgz
```

Now you can perform the four steps indicated in the VisualSFM documentation:



**Figure 1:** The VisualSFM toolbar

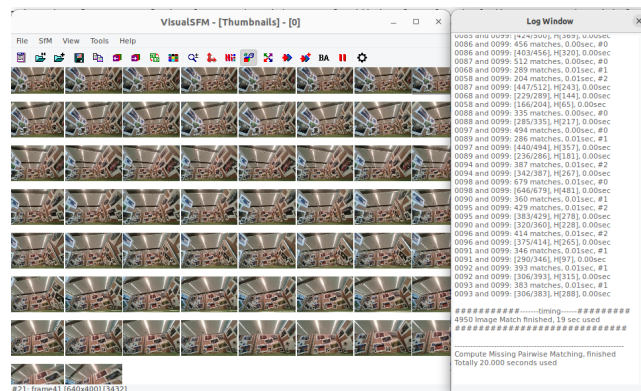
#### 3.1 Add some images

Here you can add the 119 images you unpacked in the `/home/yourname/src/vsfm/data/small_maze` directory. The nine-colored-block button in the toolbar allows you to see the thumbnails.

<sup>3</sup><http://ccwu.me/vsfm/doc.html>

### 3.2 Match the images

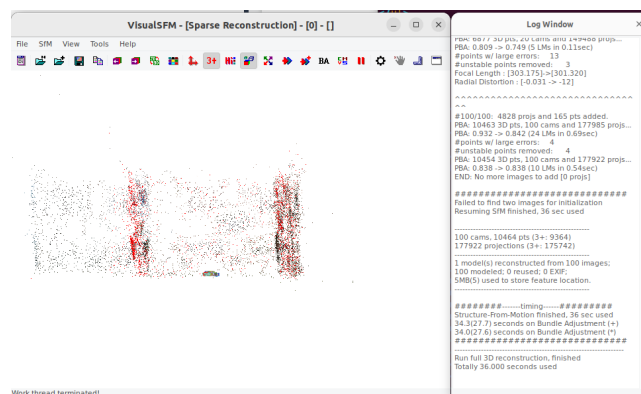
When you select this button, the library `libsiftgpu.so` you built in Section 2.7 will be used:



**Figure 2:** Matching the SIFT image features

### 3.3 Sparse reconstruction

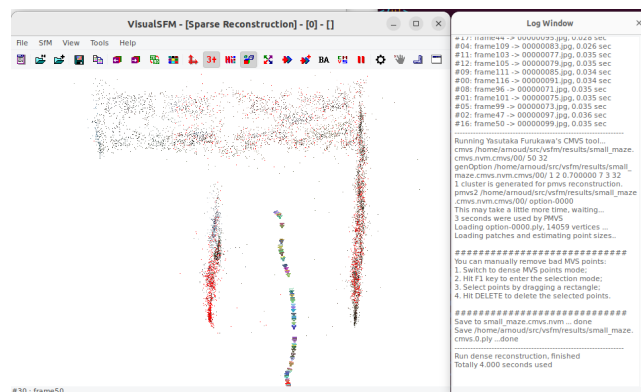
When you select this button, the library `libpba.so` you built in Section 2.8 will be used:



**Figure 3:** Sparse Reconstruction

### 3.4 Dense reconstruction

When you select this button, the tool `cmvs` you built in Section 2.11 will be used:



**Figure 4:** Dense Reconstruction

You can combine those steps in a single command with VisualSfM `sfm+pmvs /home/yourname/src/vsfm/data/small_maze /home/yourname/src/vsfm/results/small_maze.cmvs.nvm`.



---

## 4 Conclusion

When you are able to make a dense reconstruction for a small example on your local Ubuntu machine, you could try larger datasets. This will require more computation time. Scott Sawyer wrote his tutorial while working on a cluster of 64 compute cores [2]. So, future work could be an update of this Ubuntu installation guide for a cluster including GPU-support.

## References

- [1] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik and M. Pollefeys, “Building Rome on a Cloudless Day”, in “Computer Vision – ECCV 2010”, pp. 368–381, Springer, 2010, doi:10.1007/978-3-642-15561-1\_27.
- [2] S. M. Sawyer, K. Ni and N. T. Bliss, “Cluster-based 3D reconstruction of aerial video”, in “2012 IEEE Conference on High Performance Extreme Computing”, pp. 1–6, 2012, doi:10.1109/HPEC.2012.6408681.
- [3] C. Wu, *Geometry-driven Feature Detection*, Ph.D. thesis, University of North Carolina at Chapel Hill, May 2011, doi:10.17615/a8q4-8h80.
- [4] C. Wu, “Towards Linear-Time Incremental Structure from Motion”, in “International Conference on 3D Vision – 3DV 2013”, pp. 127–134, 2013, doi:10.1109/3DV.2013.25.
- [5] C. Wu, S. Agarwal, B. Curless and S. M. Seitz, “Multicore bundle adjustment”, in “IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2011”, pp. 3057–3064, 2011, doi:10.1109/CVPR.2011.5995552.
- [6] C. Wu, B. Clipp, X. Li, J.-M. Frahm and M. Pollefeys, “3D model matching with Viewpoint-Invariant Patches (VIP)”, in “IEEE Conference on Computer Vision and Pattern Recognition – CVPR 2008”, 2008, doi:10.1109/CVPR.2008.4587501.

## IRL/IAS reports

This report is in the series of IRL technical reports, which is a continuation of the original IAS technical reports. The IRL series editor is Arnoud Visser ([A.Visser@uva.nl](mailto:A.Visser@uva.nl)) The IAS series editor was Bas Terwijn ([B.Terwijn@uva.nl](mailto:B.Terwijn@uva.nl)). Within this series the following titles appeared:

Qi Zhang and A. Visser *Automatic Control, Calibration and Recording for the FrodoBots* Technical Report IRL-UVA-24-01, Informatics Institute, University of Amsterdam, The Netherlands, September 2024.

A. Visser *A Guide to the RoboCup Virtual Rescue Worlds* Technical Report IRL-UVA-16-01, Informatics Institute, University of Amsterdam, The Netherlands, May 2016.

A. Visser *UvA Rescue Technical Report: A description of the methods and algorithms implemented in the UvA Rescue code release* Technical Report IAS-UVA-12-02, Informatics Institute, University of Amsterdam, The Netherlands, December 2012.

All technical reports are available for download at the IRL website: <https://www.intelligentroboticslab.nl/reports-and-theses/>.