Free Photogrammetry on Mac OS: From Photos to 3D Models

by joecooning

Photogrammetry is the use of images/photography to measure distances between objects (thanks Webster). But for modern purposes, it is often used to make a 3D model of somethings from the real world without needing a 3D Scanner.

There is plenty of software out there that you can use for photogrammetry, including some great free options, but I noticed that many (such as Meshroom) didn't have Mac builds available. Or they would require a graphics card with CUDA support (not typical with things like a Macbook). So that meant some digging around.

I finally stumbled on this excellent article: https://peterfalkingham.com/2017/12/17/free-photog...

This lead to a follow-up build script: https://peterfalkingham.com/2018/04/01/colmap-open...

It took me a while to get it working, but once I got it going, I was pretty pleased with the results I started getting. So, I'm going to break down the steps a little more, especially for you Mac users out there.

Step 1: Get COLMAP

COLMAP (https://colmap.github.io/install.html) is a nice, little free tool for getting started with photogrammetry. I tried it on its own at first, but some of the later steps required CUDA. So I had to start looking again. That is why we will use other software for the later steps.

Downloading COLMAP is easy. You can follow the instructions here: https://colmap.github.io/install.html

Or you can look at the latest release on their github page: https://github.com/colmap/colmap/releases and download the latest COLMAP-dev-mac-no-cuda.zip

Once you dowload the zip, unzip it and stick the COLMAP app into your Applications folder.

Step 2: Get and Build OpenMVS

The next program that I use to help finish building the 3d models is OpenMVS (http://cdcseacave.github.io/openMVS/). You are going to have to build this on your machine, so I will attempt to make this as painless as possible.

The link to the instructions for getting and building OpenMVS on a Mac is here: https://github.com/cdcseacave/openMVS/wiki/Buildin...

but I did have to modify slightly. Here is what I did:

- Download Xcode from the App store
    - Open Xcode and agree to the license

- Install Homebrew if you don't already have it: https://treehouse.github.io/installation-guides/m...
- Install GIT: https://git-scm.com/download/mac
- Install CMake: https://cmake.org/install/
- Open up a terminal and execute the following script. Make sure you do it from somewhere where you want openMVS to live (I have a 'Projects' folder under my main user):

```
#Install dependencies
 brew update
brew install boost eigen opencv cgal ceres-solver
main_path=`pwd`

#VCGLib (Required)
git clone https://github.com/cdcseacave/VCG.git vcglib

#Getting the OpenMVS sources:
git clone https://github.com/cdcseacave/openMVS.git

#Build OpenMVS
mkdir openMVS_build && cd openMVS_build
cmake . ../openMVS -DCMAKE_BUILD_TYPE=Release -DVCG_ROOT="$main_path/vcglib" -G "Xcode"

 xcodebuild -configuration Release
```

Step 3: Create a Photogrammetry Script

I then created a script based on the one from here: https://peterfalkingham.com/2018/04/01/colmap-open...

Here is what I ended up with (pay attention to the notes in the script, since it requires you to set some locations):

Photogrammetry.sh

```
 # These parameters are specific to computer

# Store current Directory:
currDir=$PWD

 # get folder name as variable
myfolder=${PWD##*/}

 # Set colmap directory (change this to where you've downloaded colmap, replace 'dev' with version number if necessary):
colDir=/Applications/COLMAP.app/Contents/MacOS/colmap

 # Set openMVS directory (change this to the 'bin/Release' folder where you've downloaded and built openMVS)
oMVS=/Users/joecooning/Projects/openMVS_build/bin/Release

 # Set Working Directory (I create a temporary workspace folder in my 'Projects' directory to process data in)
workDir=/Users/joecooning/Projects/3dscans/workspace/$myfolder/

 mkdir $workDir
cp *.jpg $workDir/
cp *.JPG $workDir/
cd $workDir

 $colDir feature_extractor --database_path database.db --image_path .
$colDir exhaustive_matcher --database_path database.db
mkdir sparse
$colDir mapper --database_path database.db --image_path . --output_path sparse
$colDir model_converter --input_path sparse/0 --output_path model.nvm --output_type NVM
$oMVS/InterfaceVisualSFM model.nvm
$oMVS/DensifyPointCloud model.mvs
$oMVS/ReconstructMesh model_dense.mvs
$oMVS/RefineMesh --resolution-level 1 model_dense_mesh.mvs
$oMVS/TextureMesh --export-type obj -o $myfolder.obj model_dense_mesh_refine.mvs

 mkdir $currDir/model/
cp *.obj $currDir/model/
cp *.mtl $currDir/model/
cp *Kd.jpg $currDir/model/

 cd $currDir
```

Step 4: Run the Script

Now that you have the script, you can take photos of an object that you want to make a 3d model of. There are other articles and videos out there that can give you some great advice on how to best take photos for the purpose of photogrammetry (such as this one: https://www.tested.com/art/makers/460142-art-photo...).

But those photos into a folder, the copy the script you made into the folder.

From you terminal, go to the folder where the photos and script are and run:

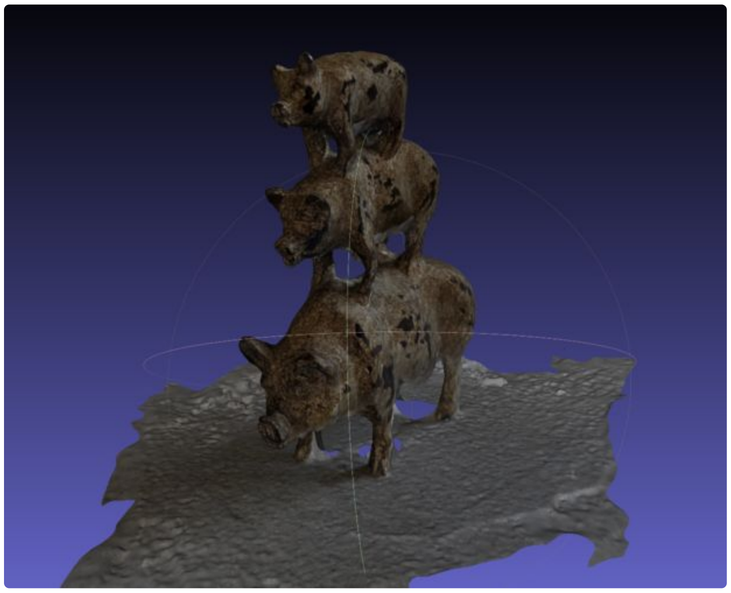
```
sh Photogrammetry.sh
```

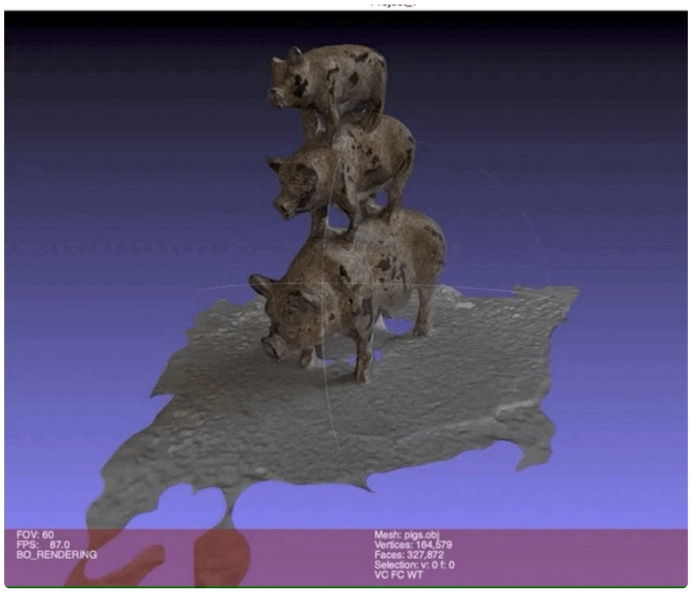
The script will then do the rest of the work for you. Please note that this can take quite a bit of time to process (especially if you are using a lot of high-res photos). I would suggest trying some smaller photo sets first. Some simple, sample photo sets can be found here: (http://www.regard3d.org/index.php/demo-models)

Step 5: Check Out You Model

Once the script finishes running, you should have an OBJ file exported into a model file where you had the photos. If you go to your workspace folder, you can find intermediate PLY models that you can check out in programs such as MeshLab.

Hope you enjoy and let me know if it worked for you. I'm also open to suggestions for improvements in the process.





```
FOV: 60
FPS:   87.0
BO_RENDERING
```

```
Mesh: pigs.obj
Vertices: 164,579
Faces: 327,872
Selection: v: 0 f: 0
VC FC WT
```