Difficulty Level : Beginners

From 2D Images to 3D Surface Models

Processing of UAV data Using Open Source Tools

*__Detailed__ screenshots of processing Using : **VisualSfM + MeshLab**
*__Steps__ of Processing Using : **osmBundler, VisualSfM+CMPMVS**

PS Singh
Scientist/Engineer-SE
ss.puyam@nesac.gov.in

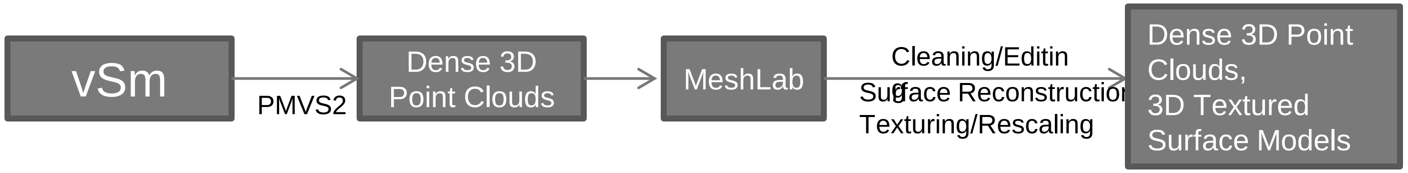
Some Open Source Tools for UAV Data Processing

Tools	Process Outputs	System Requirements
VisualSfM+Meshlab	Dense 3D Point Clouds,3D Textured Surface Models	Windows/Linux (64 bit)
Python osm-bundler	Dense 3D Point Clouds	Windows/Linux (32/64 bit)
VisualSfM+CMPMVS	OrthoMosaic, DEM, 3D Textured Surface Models	Windows/Linux (64 bit)
OpenDroneMap	OrthoMosaic, Georeferenced Point Clouds, 3D Textured Surface Models	Linux (Ubuntu LTS 14 N above)
SfMToolkit3	OrthoMosaic, Dense 3D Point Clouds, DSM	Windows(32/64 bit)

1

VisualSfM, Meshlab

Dense 3D Point Clouds, 3D Textured Surface Models



VisualSFM

VisualSFM finds unique features in each image, matches the features and constructs sparse 3D point cloud, The model is then refined into a dense point cloud.

VisualSFM gives two main outputs -
.out file (bundler format) which stores the calculated (solved) cameras' positions and a sparse point cloud representing the points in the scene that the software used to determine the camera positions.

.ply file which stores a denser cloud of points, each with position, color and normal (a vector perpendicular to the surface the point is on) data.

Meshlab

Meshlab works on 3D mesh or a point cloud input files

Meshlab produces textured, clean, high-res mesh.
Meshlab also automatically calculates UV maps (the basis for 3D texturing) and builds a texture for us by estimating which projector is most accurate on each spot of the model. The outputs of this step are:

.obj file of a mesh (with UVs) that can be easily interchanged with various 3D softwares
.png file of varying size representing the texture of the mesh

VisualFSM

Example Scene :

Landslide Scar, Umling, Ribhoi

90 Images

Few Points to get the best possible reconstruction

1. **Parallax**: MOVE AROUND!. Move in a 360 degree circle around an object, taking snaps every 10-20 degrees, as well as at several heights
2. **Overlap**: minimum 75% overlap, 40% sidelap
3. **Photograph all angles :** to get everything
4. **Scale**: reconstructing entire environments is totally possible, but be prepared to take a lot more photographs. Smaller objects require fewer images and the process will take less time
5. **Minimize subject motion**: The software expects that everything in the scene is stationary when it analyzes the scene. if objects move around, the assumption that features are in the same space between photographs will be false, and VisualSFM will build a distorted model

Step 1: Import the images

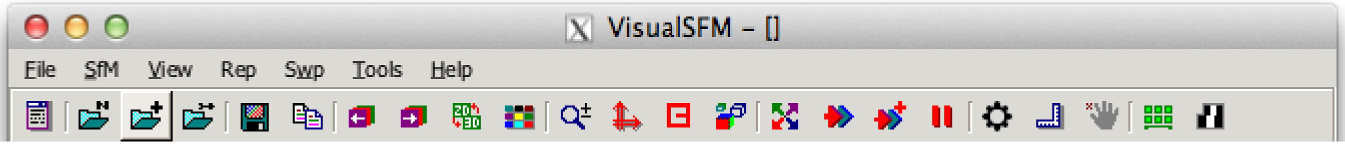
Note : **VisualSFM** will produce generally better results the higher-resolution your images are, **BUT**, after 3200px, the results get worse !

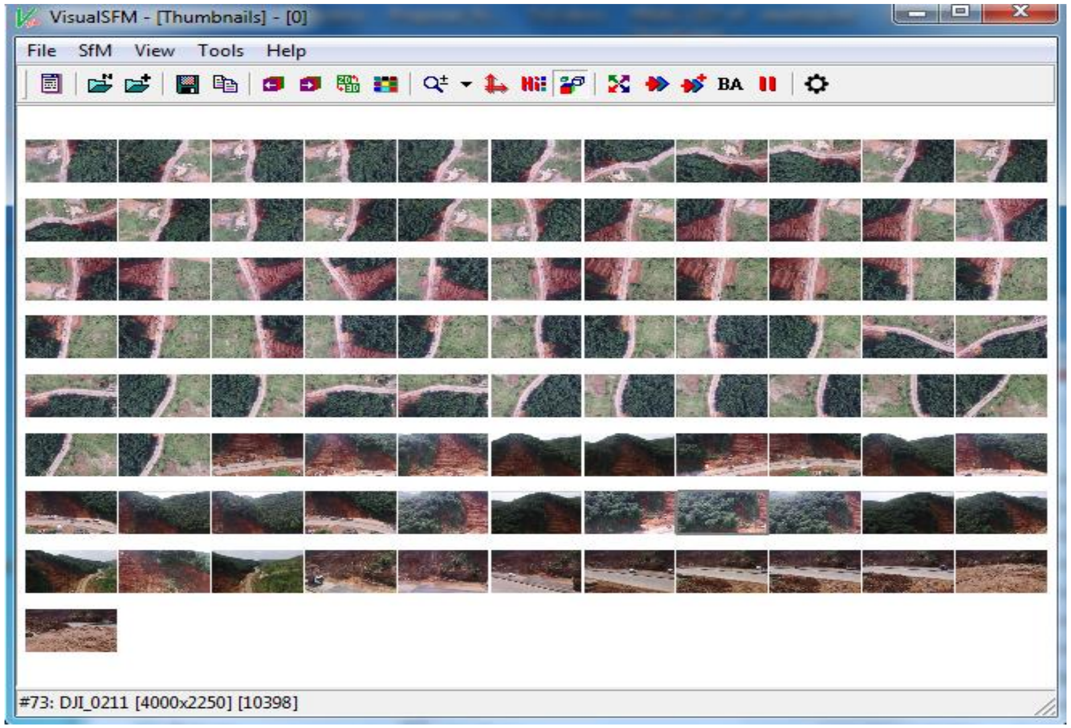
Before you import the images, make sure your images are no larger than 3200px tall or wide

Note : *Some experiments have been performed, that smaller images (<2k) actually improve feature detection and matching success. The process of matching features between images is done on the GPU. If you have high-end GPUs such as Tegra,...then you may try with better res imagery.*

If you do have to resize your images, make sure to retain any EXIF metadata logged by the camera as VisualSFM can use it in order to pick more accurate focal length and sensor size settings. VisualSFM installs a FOSS program called jhead for reading EXIF data from the command line

Step 2: click the second folder icon to import your image sequence

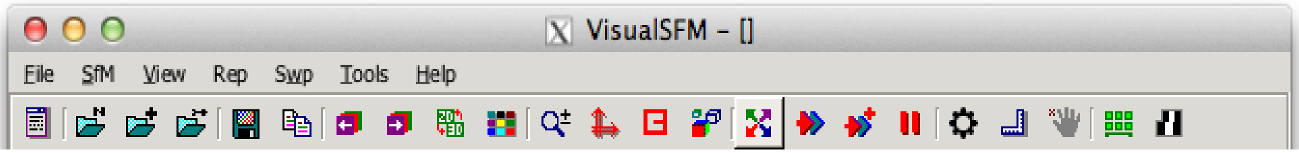




#73: DJI_0211 [4000x2250] [10398]

Note : Enable the task, it's in Tools –> Show TaskViewer and gives logs of what VisualSFM is doing

Step 2: SIFT and match : Click on icon that looks like 4 arrows in an X shape on the toolbar.



Detection :"SIFTing" is where unique features of each image are detected and logged

(This process can accelerate on machines with better GPU)

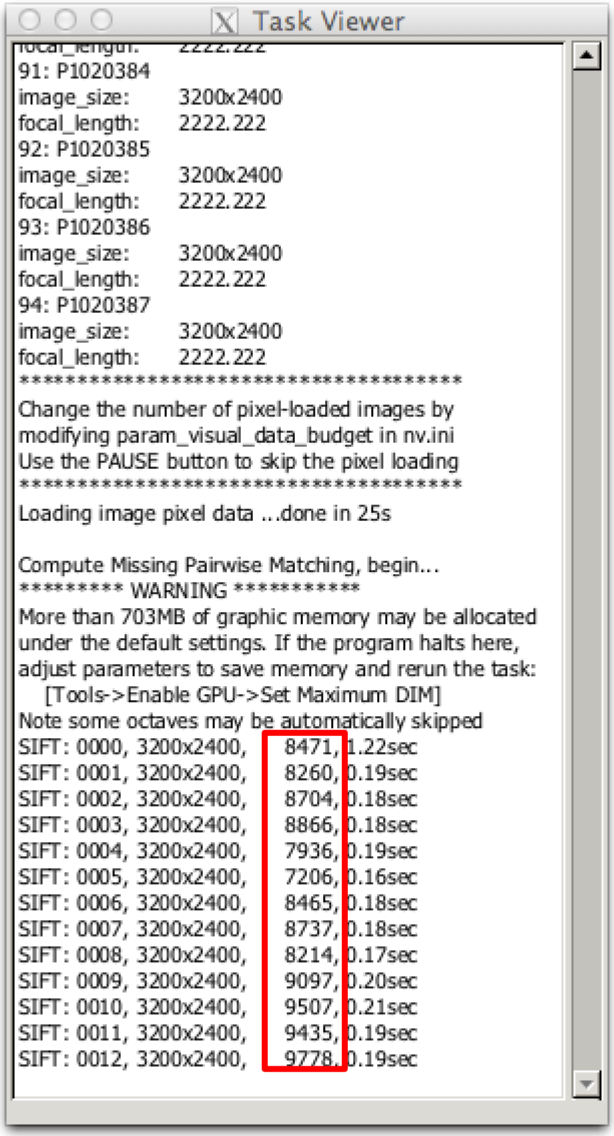
Matching : Features are matched between images

(amount of comparisons that have to be made is exponentially related to the number of images)

Neighbour comparison : comparison to only the n images on either side of each image taken with for a linear camera move.

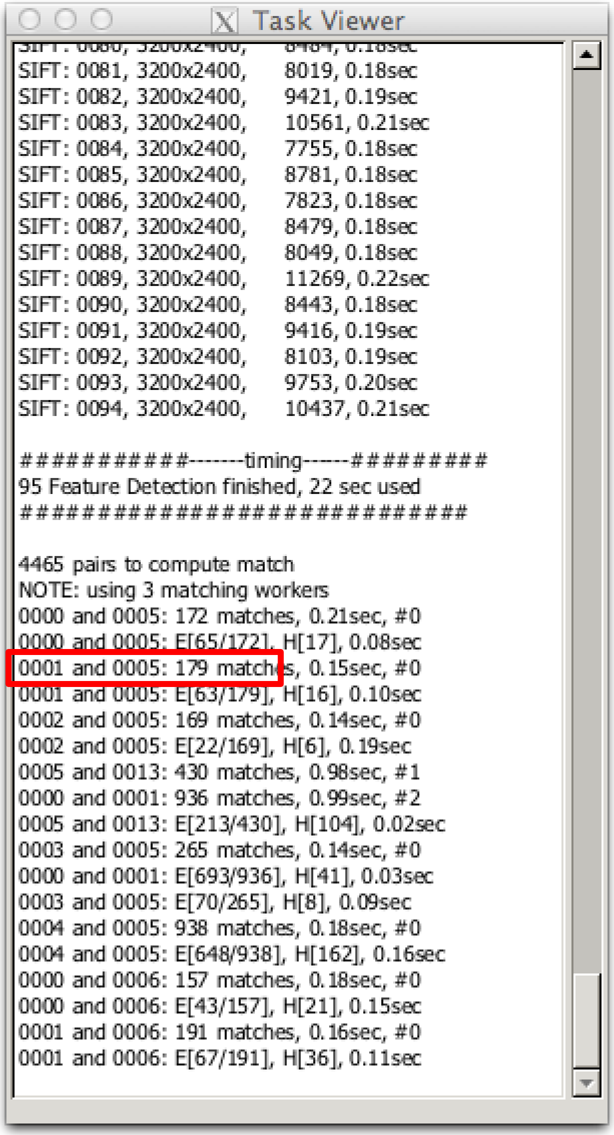
Note : In case the process is interupted or crashing out/hanged, process resumes where it left off the next time you try to SIFT and match those images

Taskviewer Logs



```
91: P1020384
image_size:      3200x2400
focal_length:    2222.222
92: P1020385
image_size:      3200x2400
focal_length:    2222.222
93: P1020386
image_size:      3200x2400
focal_length:    2222.222
94: P1020387
image_size:      3200x2400
focal_length:    2222.222
****************************************
Change the number of pixel-loaded images by
modifying param_visual_data_budget in nv.ini
Use the PAUSE button to skip the pixel loading
****************************************
Loading image pixel data ...done in 25s

Compute Missing Pairwise Matching, begin...
********* WARNING ***********
More than 703MB of graphic memory may be allocated
under the default settings. If the program halts here,
adjust parameters to save memory and rerun the task:
    [Tools->Enable GPU->Set Maximum DIM]
Note some octaves may be automatically skipped
SIFT: 0000, 3200x2400,     8471, 1.22sec
SIFT: 0001, 3200x2400,     8260, 0.19sec
SIFT: 0002, 3200x2400,     8704, 0.18sec
SIFT: 0003, 3200x2400,     8866, 0.18sec
SIFT: 0004, 3200x2400,     7936, 0.19sec
SIFT: 0005, 3200x2400,     7206, 0.16sec
SIFT: 0006, 3200x2400,     8465, 0.18sec
SIFT: 0007, 3200x2400,     8737, 0.18sec
SIFT: 0008, 3200x2400,     8214, 0.17sec
SIFT: 0009, 3200x2400,     9097, 0.20sec
SIFT: 0010, 3200x2400,     9507, 0.21sec
SIFT: 0011, 3200x2400,     9435, 0.19sec
SIFT: 0012, 3200x2400,     9778, 0.19sec
```

Feature Detection

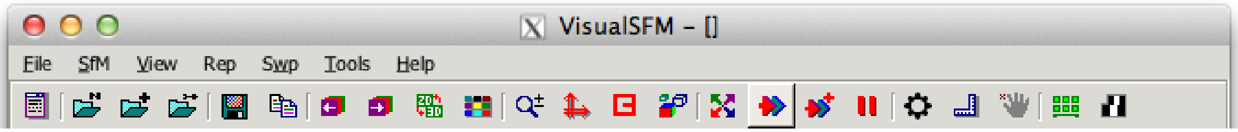


```
SIFT: 0081, 3200x2400,     8019, 0.18sec
SIFT: 0082, 3200x2400,     9421, 0.19sec
SIFT: 0083, 3200x2400,     10561, 0.21sec
SIFT: 0084, 3200x2400,     7755, 0.18sec
SIFT: 0085, 3200x2400,     8781, 0.18sec
SIFT: 0086, 3200x2400,     7823, 0.18sec
SIFT: 0087, 3200x2400,     8479, 0.18sec
SIFT: 0088, 3200x2400,     8049, 0.18sec
SIFT: 0089, 3200x2400,     11269, 0.22sec
SIFT: 0090, 3200x2400,     8443, 0.18sec
SIFT: 0091, 3200x2400,     9416, 0.19sec
SIFT: 0092, 3200x2400,     8103, 0.19sec
SIFT: 0093, 3200x2400,     9753, 0.20sec
SIFT: 0094, 3200x2400,     10437, 0.21sec

##########-------timing------#########
95 Feature Detection finished, 22 sec used
#############################

4465 pairs to compute match
NOTE: using 3 matching workers
0000 and 0005: 172 matches, 0.21sec, #0
0000 and 0005: E[65/172], H[17], 0.08sec
0001 and 0005: 179 matches, 0.15sec, #0
0001 and 0005: E[63/179], H[16], 0.10sec
0002 and 0005: 169 matches, 0.14sec, #0
0002 and 0005: E[22/169], H[6], 0.19sec
0005 and 0013: 430 matches, 0.98sec, #1
0000 and 0001: 936 matches, 0.99sec, #2
0005 and 0013: E[213/430], H[104], 0.02sec
0003 and 0005: 265 matches, 0.14sec, #0
0000 and 0001: E[693/936], H[41], 0.03sec
0003 and 0005: E[70/265], H[8], 0.09sec
0004 and 0005: 938 matches, 0.18sec, #0
0004 and 0005: E[648/938], H[162], 0.16sec
0000 and 0006: 157 matches, 0.18sec, #0
0000 and 0006: E[43/157], H[21], 0.15sec
0001 and 0006: 191 matches, 0.16sec, #0
0001 and 0006: E[67/191], H[36], 0.11sec
```

Feature Matching

Step 3: Sparse 3D reconstruction : Click on button with the two arrows on it (>>) to start this process.

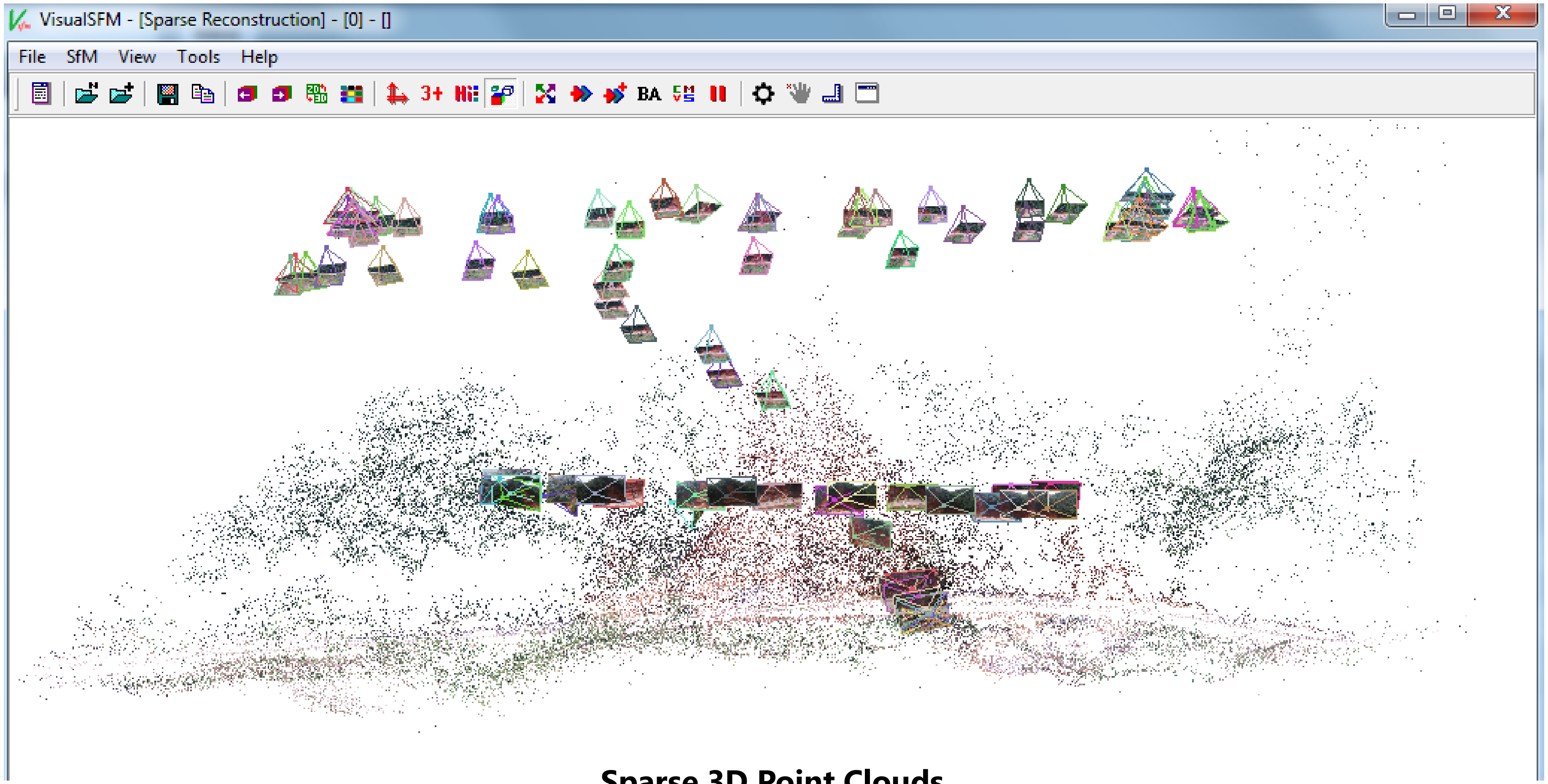
Once the overlapping features have been detected, you can begin a sparse 3D reconstruction



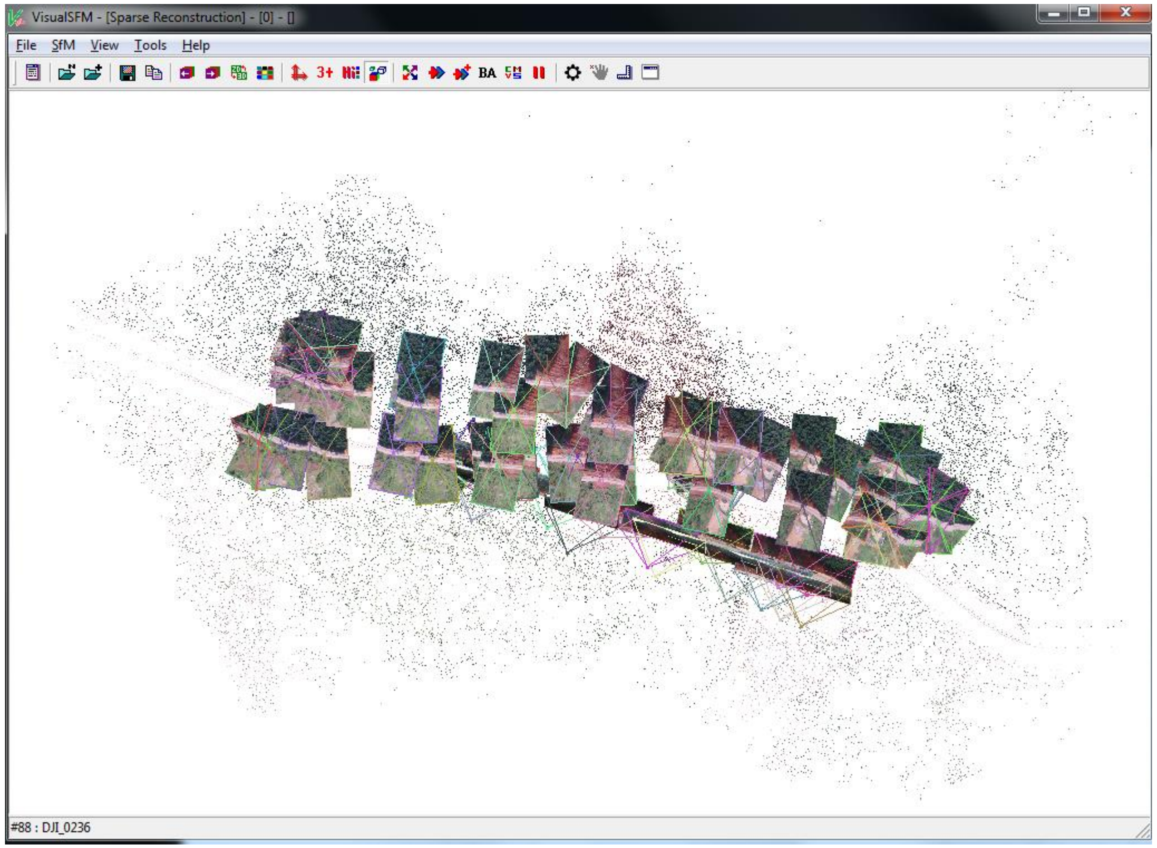
This process is interactive! : Pan, rotate and zoom through the 3D space as it tries to solve the feature overlaps into a 3D form!

See the position and rotation of the cameras that recorded the original images

ctrl+mousewheel up or down in order to scale the photos that are reconstructing the scene to get an idea of which photos are being places where



Sparse 3D Point Clouds



Adjusted the size of the projectors up a bit so you can how it's fitting each camera / image into the scene

This process is interactive in that you can influence the outcome of the solve by removing bad cameras

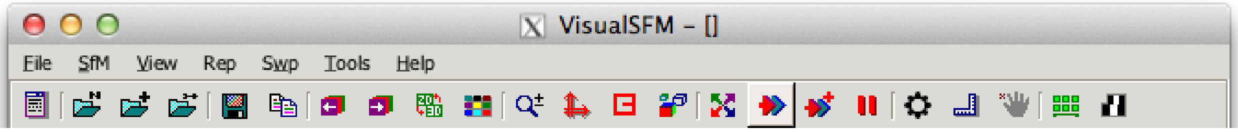
Removing bad cameras (Optional)

Method 1.

1.Turn off 3+
2.Click F2, select the bad 3D points
3.Click the 'hand' icon twice, it will delete the camera that sees the largest number of selected points
4.Go to step 1 if there are still bad 3D points
5.Click >>+ to grow the modified 3D model

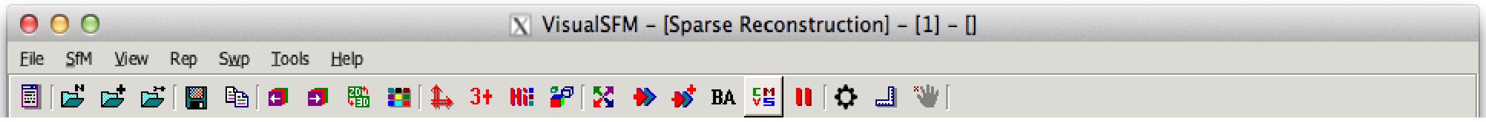
Method 2.

1. Click F1, draw a rectangle to select all the bad cameras (at most 250 cameras at a time)
2. Click the "hand" icon to remove all the bad cameras
3. Click >>+ to grow the modified 3D model.



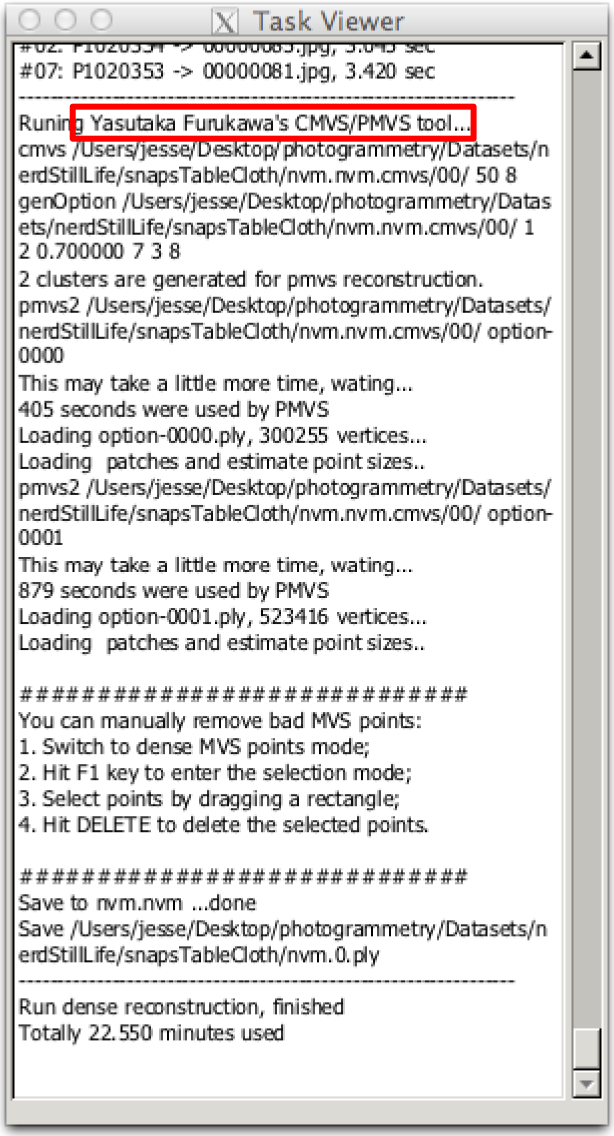
Step 4: Dense reconstruction : Click on the "cmvs" button, and save the cmvs folder structure

Takes Sparse Point cloud and computes a much denser version of the point cloud



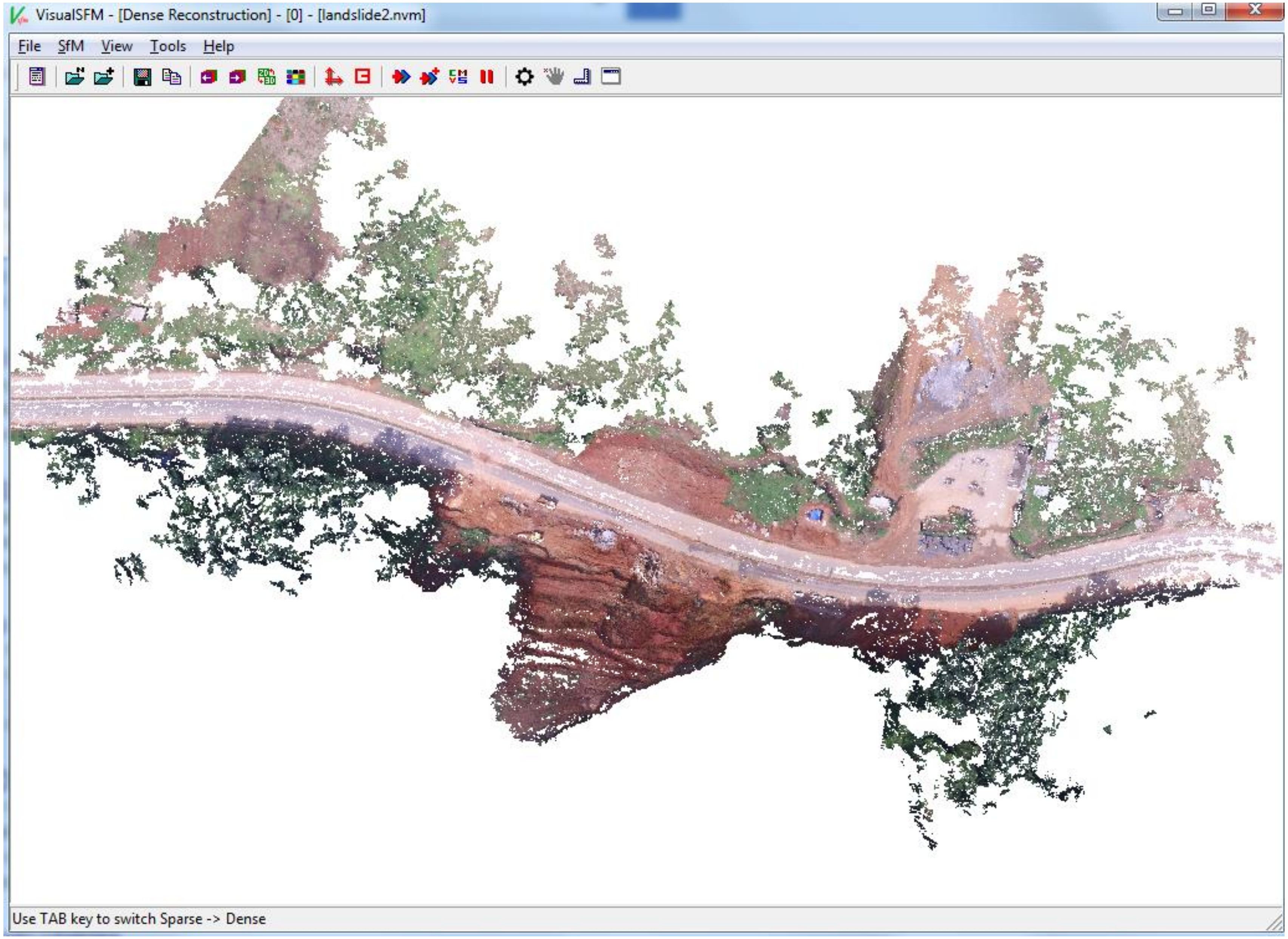
VisualSFM will export its internal sparse reconstruction to that folder in the format needed by cmvs / pmvs2 and that software is then run in order to populate that folder structure with the dense point cloud

Taskviewer Logs



```
#07: P1020353 -> 00000081.jpg, 3.420 sec
----------------------------------------------------------------
Runing Yasutaka Furukawa's CMVS/PMVS tool...
cmvs /Users/jesse/Desktop/photogrammetry/Datasets/n
erdStillLife/snapsTableCloth/nvm.nvm.cmvs/00/ 50 8
genOption /Users/jesse/Desktop/photogrammetry/Datas
ets/nerdStillLife/snapsTableCloth/nvm.nvm.cmvs/00/ 1
2 0.700000 7 3 8
2 clusters are generated for pmvs reconstruction.
pmvs2 /Users/jesse/Desktop/photogrammetry/Datasets/
nerdStillLife/snapsTableCloth/nvm.nvm.cmvs/00/ option-
0000
This may take a little more time, wating...
405 seconds were used by PMVS
Loading option-0000.ply, 300255 vertices...
Loading  patches and estimate point sizes..
pmvs2 /Users/jesse/Desktop/photogrammetry/Datasets/
nerdStillLife/snapsTableCloth/nvm.nvm.cmvs/00/ option-
0001
This may take a little more time, wating...
879 seconds were used by PMVS
Loading option-0001.ply, 523416 vertices...
Loading  patches and estimate point sizes..

############################
You can manually remove bad MVS points:
1. Switch to dense MVS points mode;
2. Hit F1 key to enter the selection mode;
3. Select points by dragging a rectangle;
4. Hit DELETE to delete the selected points.

############################
Save to nvm.nvm ...done
Save /Users/jesse/Desktop/photogrammetry/Datasets/n
erdStillLife/snapsTableCloth/nvm.0.ply
----------------------------------------------------------------
Run dense reconstruction, finished
Totally 22.550 minutes used
```

Once the dense reconstruction is complete, Click tab to see a visualization within VisualSFM of 3D Dense Point Cloud



Use TAB key to switch Sparse -> Dense

Click on ‘F’ to hide/show camera position

Note : you can also perform a dense reconstruction with cmp-mvs

VisualSFM Process Conclusion

1. Debugging – Remove bad points / cameras that have been added in the wrong position / orientation in order to improve
2. The SIFT and matching process is stored on disk

   .sift and .mat file for each image

Meshlab

“**MeshLab** is an advanced 3D mesh processing software system that is oriented to the management and processing of unstructured large meshes and provides a set of tools for editing, cleaning, healing, inspecting, rendering, and converting these kinds of meshes. MeshLab is free and open-source software, subject to the requirements of the GNU General Public License (GPL), version 2 or later, and is used as both a complete package and a library powering other software. It is well known in the more technical fields of 3D development and data handling “

Input : Meshlab operates on mesh and point-cloud data

Caution :

1. it doesn’t have undo button/action!

Changes are only made to a mesh if you “export” it.
If you want to go back, you can “reload” the current mesh you’re editing

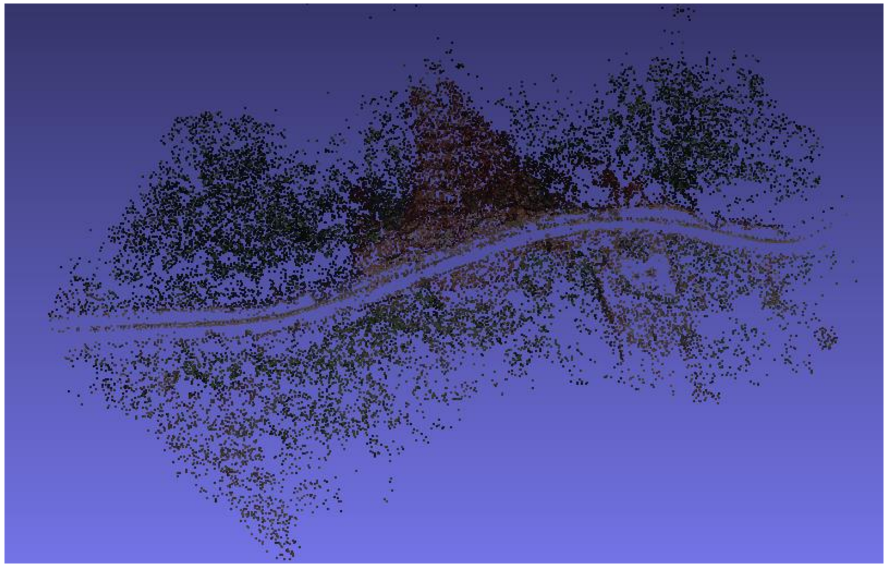
Note : Be careful and save your meshlab project AND export your mesh (“export as…” in order to save a new file) frequently

Step 1. Open the bundle.rd.out file
(to import the bundle.rd.out file to get the camera positions and rasters)

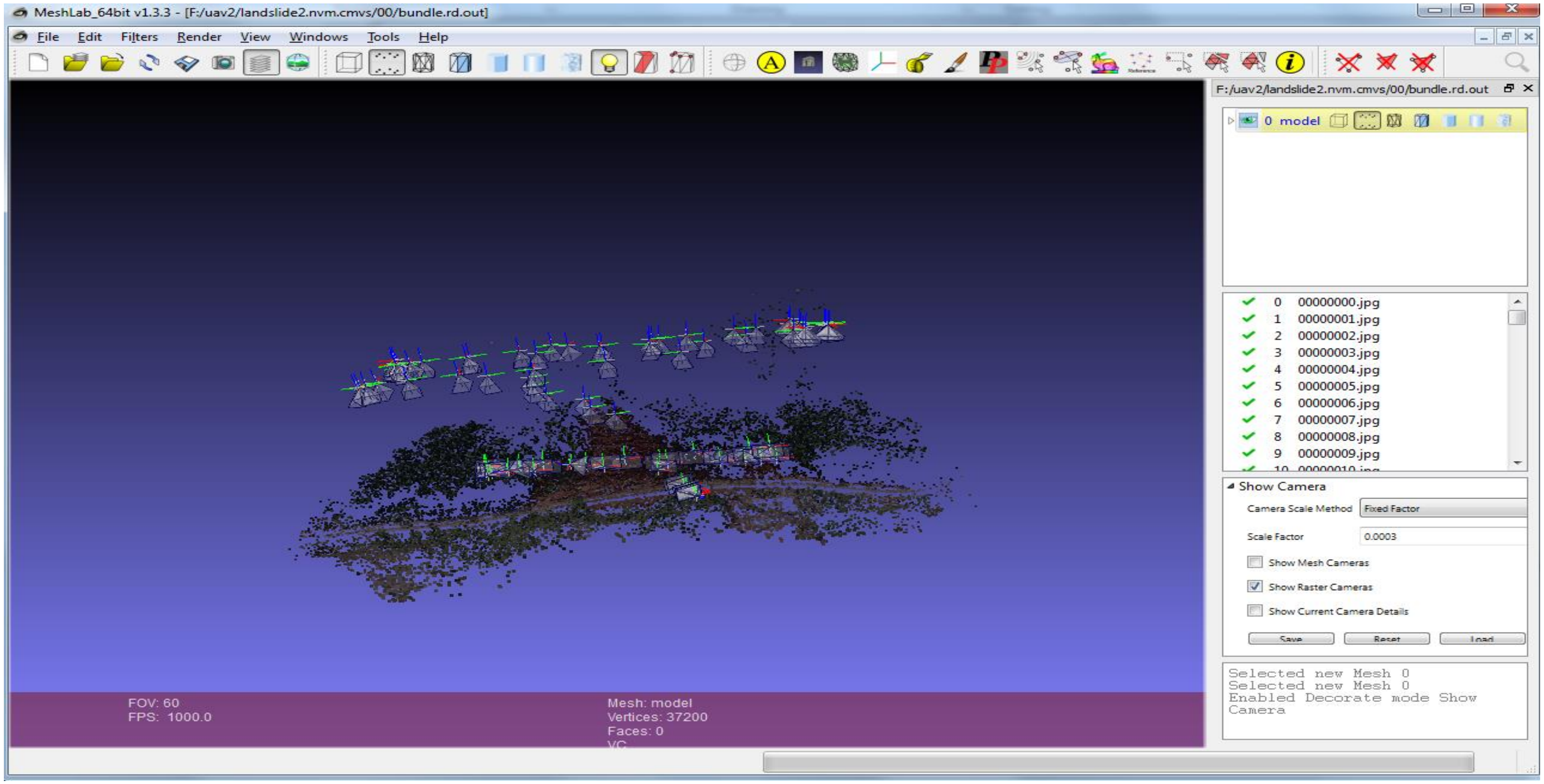
bundle.rd.out stores internal representation of the sparse reconstruction

Go to File –> Open Project, and navigate to your bundle.rd.out file within the nvm directory

Meshlab will ask you as soon as you select the bundle.rd.out file to select the camera file. That's in a file in the same directory called, "list.txt", so select that next



The above steps will import the cameras from your solve into the scene as well as the photos they're associated with (they're called raster layers in Meshlab). This is important for the texturing process later on


MeshLab_64bit v1.3.3 - [F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out]
File Edit Filters Render View Windows Tools Help
F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out
0 model

0	00000000.jpg
1	00000001.jpg
2	00000002.jpg
3	00000003.jpg
4	00000004.jpg
5	00000005.jpg
6	00000006.jpg
7	00000007.jpg
8	00000008.jpg
9	00000009.jpg

Show Camera
Camera Scale Method: Fixed Factor
Scale Factor: 0.0003
☐ Show Mesh Cameras
☑ Show Raster Cameras
☐ Show Current Camera Details
Save Reset Load

```
Selected new Mesh 0
Selected new Mesh 0
Enabled Decorate mode Show
Camera
```

FOV: 60
FPS: 1000.0

Mesh: model
Vertices: 37200
Faces: 0
VC

click on the layers icon on the toolbar to open up the layers palette. You should see your geometry layers on the top right, and the raster layers on the bottom right of the screen.

Go to Render –> Show Camera. This will display the camera frustums from all of your projectors in the viewer. The default scale of the camera visualization is huge compared to the size of our mesh, so click on the disclosure triangle next to “Show Camera” that has appeared on our layers sidebar:
change the scale factor to something like 0.001. Keep decreasing it until you can see both your mesh and the camera positions around your mesh.

Disable the camera visualization by unchecking Render –> Show Camera.

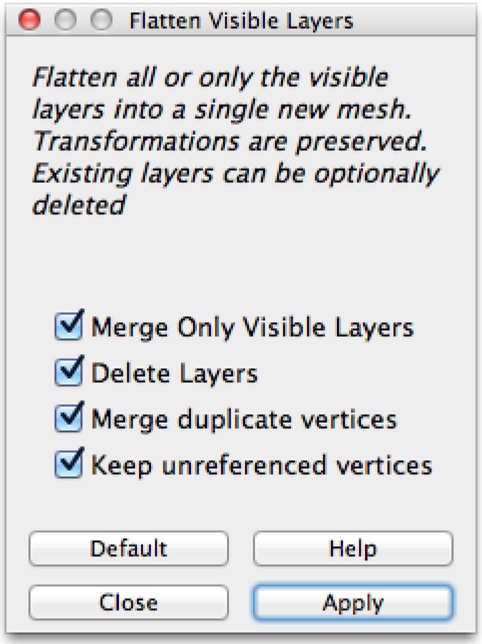
Step 2. Replace sparse cloud with the dense one

1. In the layers palate, right click on the mesh layer, and select "delete current mesh" in order to remove it entirely.
2. select File –> Import Mesh and load the dense mesh. It should be located in your NVM_FOLDER/00/models/option-0000.ply

Note : It's possible that VisualSFM (really pmvs2) generated multiple .ply files (called option-0000.ply, option-0001.ply etc). Because the dense mesher only works with 50 cameras at a time.
If you have multiple .ply files, you'll need to flatten them into one mesh before you can proceed.

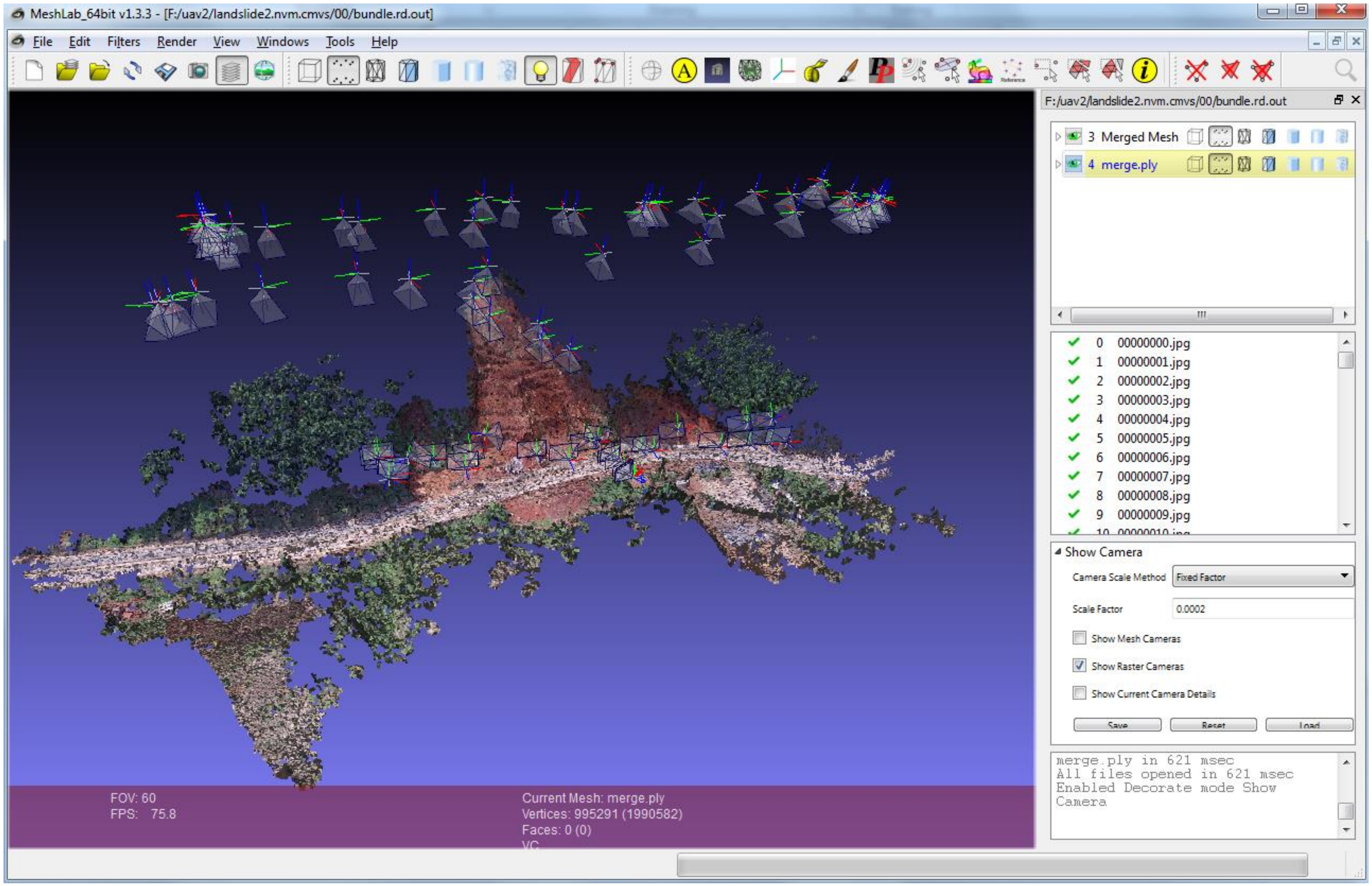
Simply right click on any of the mesh layers, and select "Flatter Visible Layers". Make sure to check "Keep unreferenced vertices":

Save as the merge ply file and import it afresh


Flatten Visible Layers

Flatten all or only the visible layers into a single new mesh. Transformations are preserved. Existing layers can be optionally deleted

- [x] Merge Only Visible Layers
- [x] Delete Layers
- [x] Merge duplicate vertices
- [x] Keep unreferenced vertices



Show Camera
Camera Scale Method: Fixed Factor
Scale Factor: 0.0002
☐ Show Mesh Cameras
☑ Show Raster Cameras
☐ Show Current Camera Details

```
merge.ply in 621 msec
All files opened in 621 msec
Enabled Decorate mode Show
Camera
```

FOV: 60
FPS: 75.8

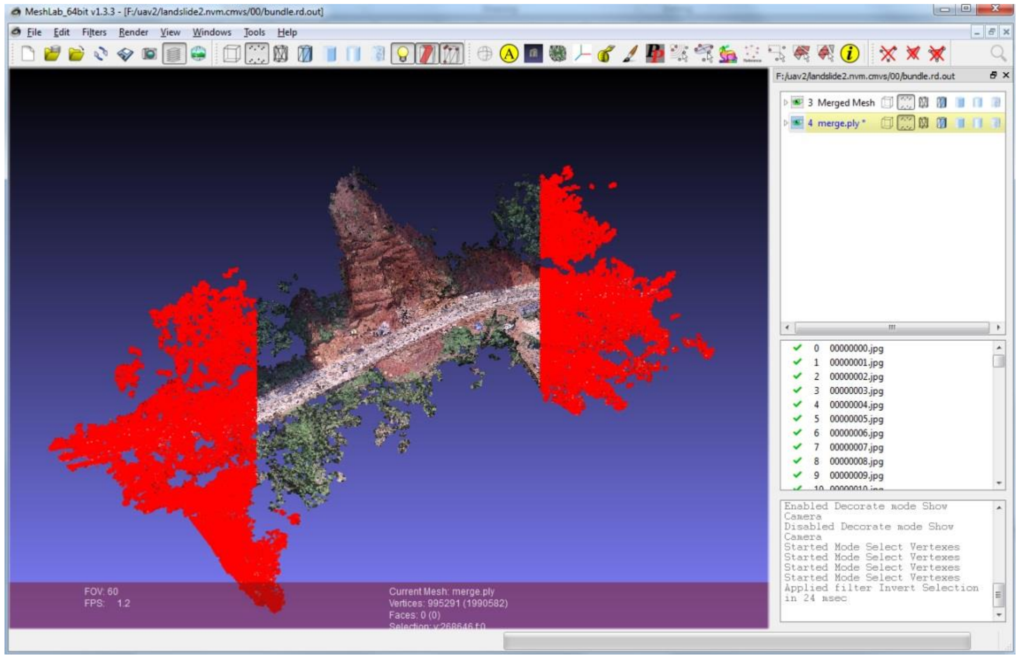
Current Mesh: merge.ply
Vertices: 995291 (1990582)
Faces: 0 (0)

Dense point cloud AND the camera positions / raster layers!

Step 3. Clean up dense 3d points

box selection tool : select regions of points that are bad

delete points button


MeshLab_64bit v1.3.3 - [F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out]
File Edit Filters Render View Windows Tools Help
F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out
3 Merged Mesh
4 merge.ply *

```
0  00000000.jpg
1  00000001.jpg
2  00000002.jpg
3  00000003.jpg
4  00000004.jpg
5  00000005.jpg
6  00000006.jpg
7  00000007.jpg
8  00000008.jpg
9  00000009.jpg
10 00000010.jpg
```

```
Enabled Decorate mode Show
Camera
Disabled Decorate mode Show
Camera
Started Mode Select Vertexes
Started Mode Select Vertexes
Started Mode Select Vertexes
Started Mode Select Vertexes
Applied filter Invert Selection
in 24 msec
```

```
FOV: 60
FPS:   1.2
```

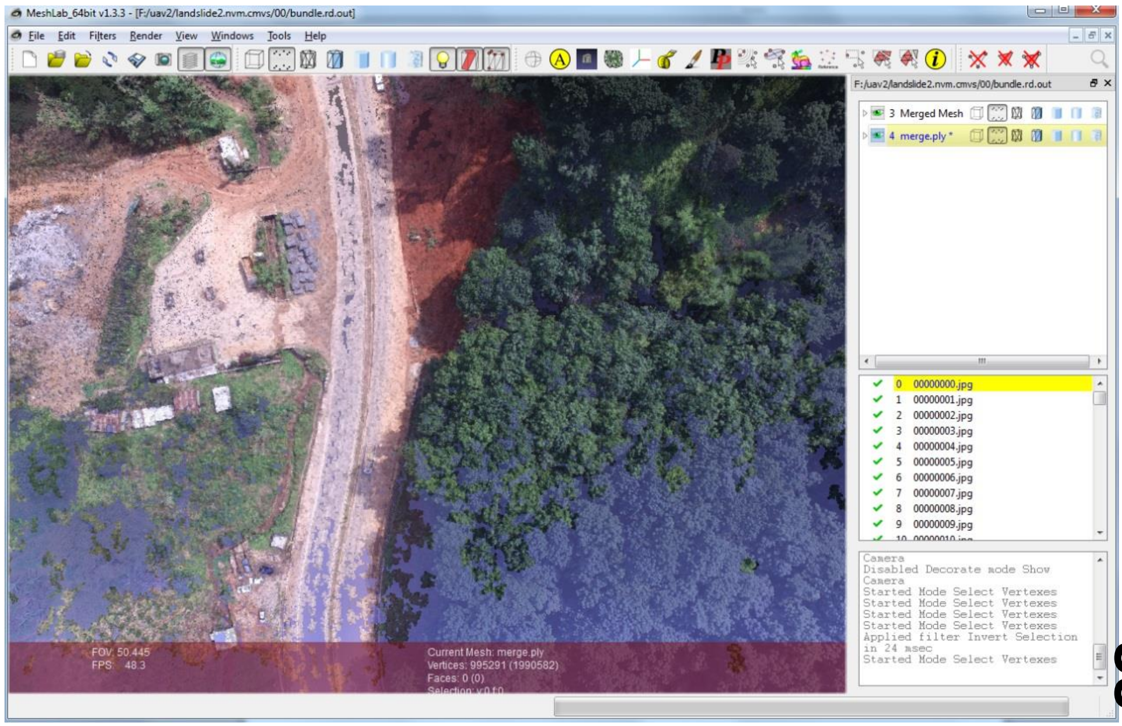
```
Current Mesh: merge.ply
Vertices: 995291 (1990582)
Faces: 0 (0)
Selection: v:268646 f:0
```

Step 4. Check the Cameras

Click on “Show current raster mode” button

The 3D camera will snap to the position where VisualSFM thought that image was taken from in 3d space! You can scroll your mousewheel to alter the transparency of your image and double check the alignment!

After, you’ve checked your cameras, disable the “Show current raster mode” by hitting the icon again.



```
FOV: 50.445
FPS:   48.3
```

```
Current Mesh: merge.ply
Vertices: 995291 (1990582)
Faces: 0 (0)
Selection: v:0 f:0
```

```
Camera
Disabled Decorate mode Show
Camera
Started Mode Select Vertexes
Started Mode Select Vertexes
Started Mode Select Vertexes
Started Mode Select Vertexes
Applied filter Invert Selection
in 24 msec
Started Mode Select Vertexes
```

Step 5. Meshing

Convert point cloud into an actual polygonal mesh

Several algorithms for doing this meshing step, but the best seems to the Poisson Surface Reconstruction algorithm by *Kazhdan, Bolitho, and Hoppe*

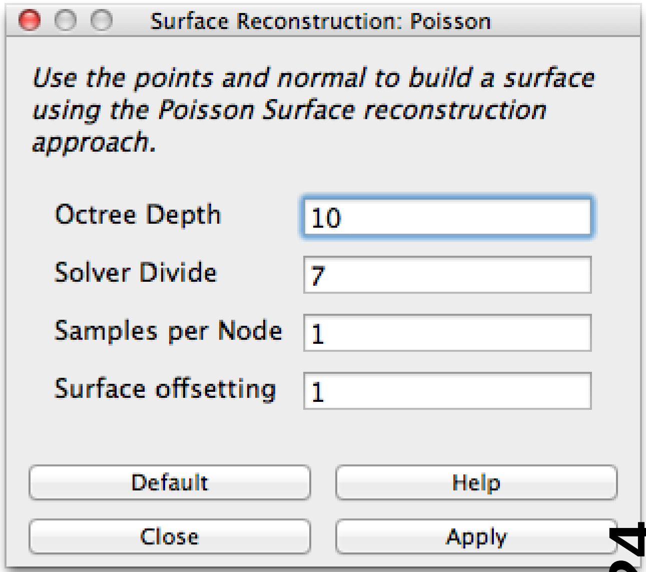
The algorithm is Open source : See the source code at *http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version6/*

Select Filter –> Points –> Surface Reconstruction: Poisson

Spend some time and experiment here,

Keep increasing the Octree Depth till you're getting meshes that have more or less the same level of detail.

Note : Each time you click on "apply" in the Poisson Surface Reconstruction filter, meshlab will create a new layer with your mesh in it. Instead of deleting the ones that are bad, keep them around and maybe even rename them with the settings you chose


Surface Reconstruction: Poisson

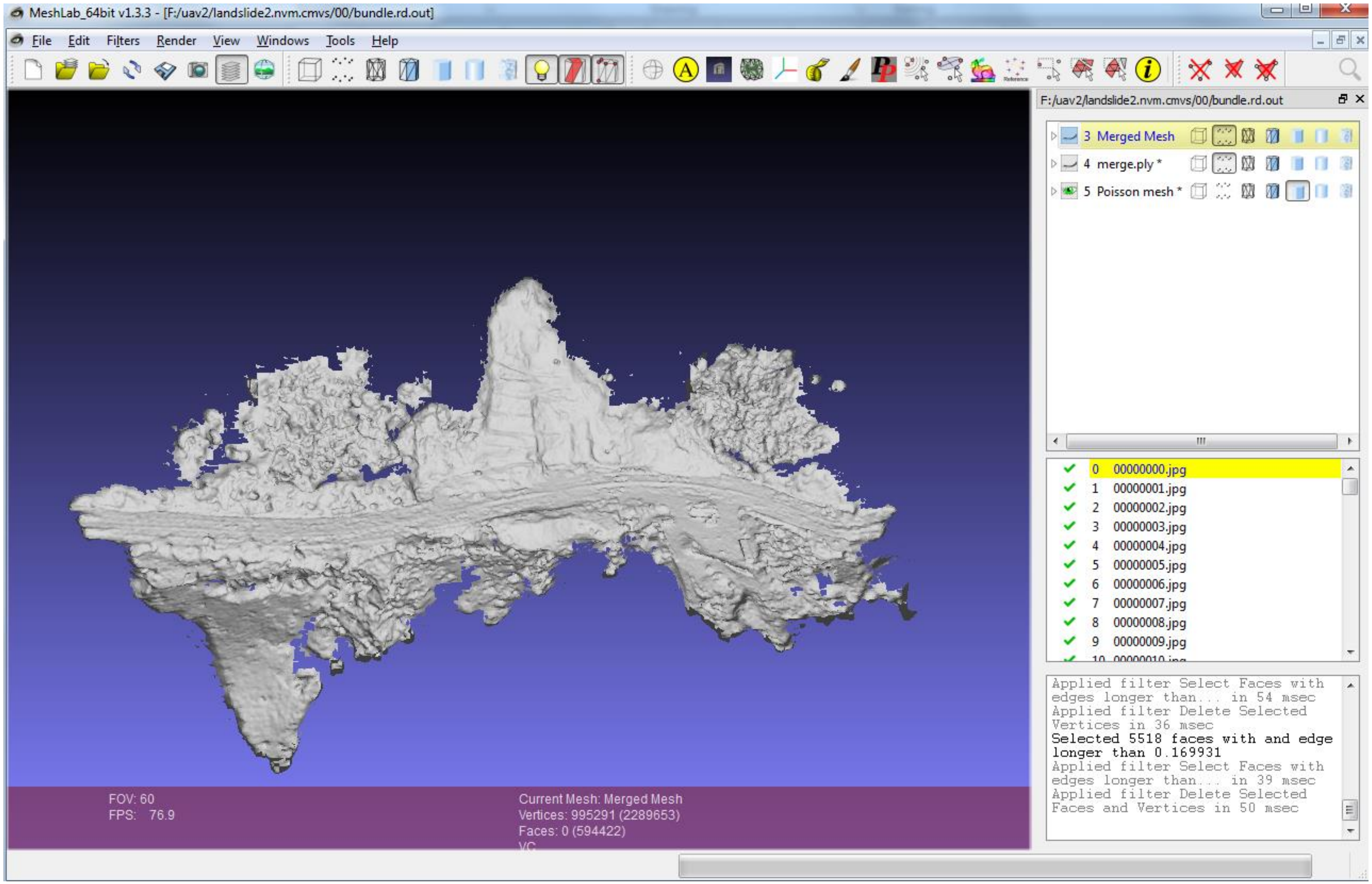
Use the points and normal to build a surface using the Poisson Surface reconstruction approach.

Octree Depth	10
Solver Divide	7
Samples per Node	1
Surface offsetting	1

Caution : Meshing

- Octree Depth seems to have the most control over the detail in the mesh. The larger that number gets, the slower things seem to go / more memory you need, so be careful
- The Solver Divide seems to help use less memory when the Octree depth increases
- "Samples per Node" setting seems to smooth the mesh out if it has a lot of errors in it (Leave it at 1-5 to start with and increase up to 20 to try to smooth your mesh out a bit if there are problems)

The Poisson Mesher always tries to return "watertight" meshes which means that often it will return a mesh-ball with your object /environment inside. In order to excavate your mesh, you'll need to spend some time deleting faces. You can try "Filters –> Selection –> Select faces with edges longer than"



MeshLab_64bit v1.3.3 - [F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out]

F:/uav2/landslide2.nvm.cmvs/00/bundle.rd.out

3 Merged Mesh
4 merge.ply *
5 Poisson mesh *

0	00000000.jpg
1	00000001.jpg
2	00000002.jpg
3	00000003.jpg
4	00000004.jpg
5	00000005.jpg
6	00000006.jpg
7	00000007.jpg
8	00000008.jpg
9	00000009.jpg
10	00000010.jpg

```
Applied filter Select Faces with
edges longer than... in 54 msec
Applied filter Delete Selected
Vertices in 36 msec
Selected 5518 faces with and edge
longer than 0.169931
Applied filter Select Faces with
edges longer than... in 39 msec
Applied filter Delete Selected
Faces and Vertices in 50 msec
```

```
FOV: 60
FPS:   76.9
```

```
Current Mesh: Merged Mesh
Vertices: 995291 (2289653)
Faces: 0 (594422)
VC
```

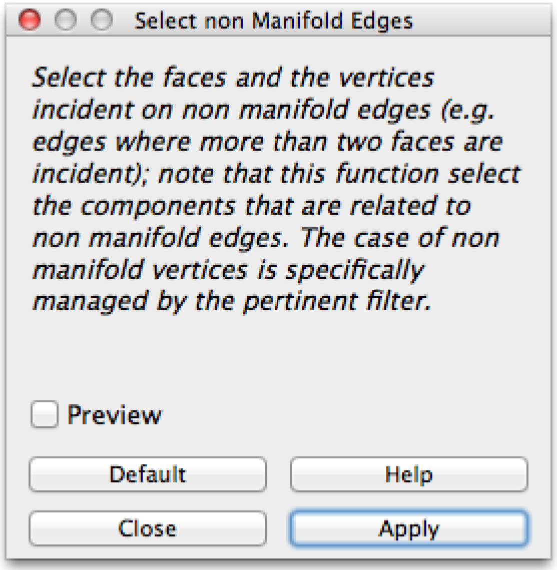
Save both mesh and project

Step 6. Fix manifold edges

After meshing, there are non-manifold edges which need to be removed before we can proceed.

Technically, non-manifold geometry is "*any edge shared by more than two faces.*" Non-technically, non-manifold geometry is bad geo that that certain algorithms can't deal with

Filters –> Selection –> Select Non-Manifold edges, hit Apply, then delete them with the delete points button.



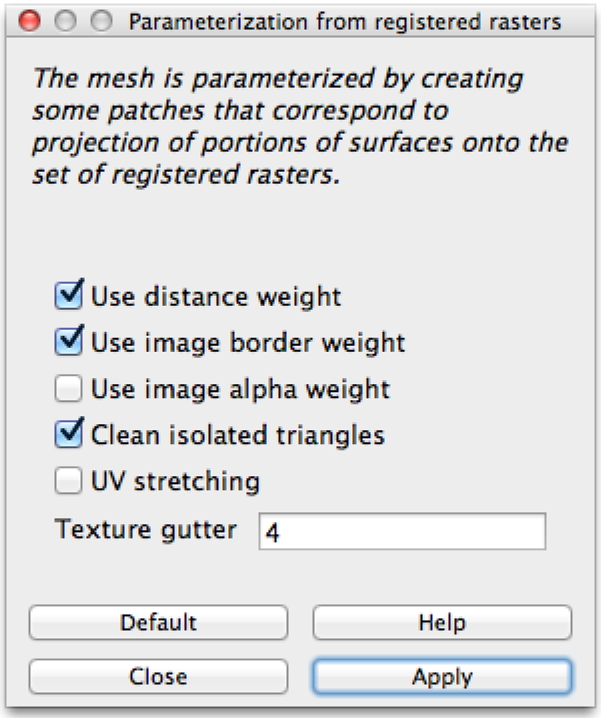
Select non Manifold Edges

Select the faces and the vertices incident on non manifold edges (e.g. edges where more than two faces are incident); note that this function select the components that are related to non manifold edges. The case of non manifold vertices is specifically managed by the pertinent filter.

Step 7. Parameterization

Automatically builds a UV map by figuring out which projector cameras have the best view of each face of the model

Filter –> Texture –> Parameterization from registered rasters, and match the following settings



Parameterization from registered rasters

The mesh is parameterized by creating some patches that correspond to projection of portions of surfaces onto the set of registered rasters.

- [✓] Use distance weight
- [✓] Use image border weight
- [ ] Use image alpha weight
- [✓] Clean isolated triangles
- [ ] UV stretching

Texture gutter: 4

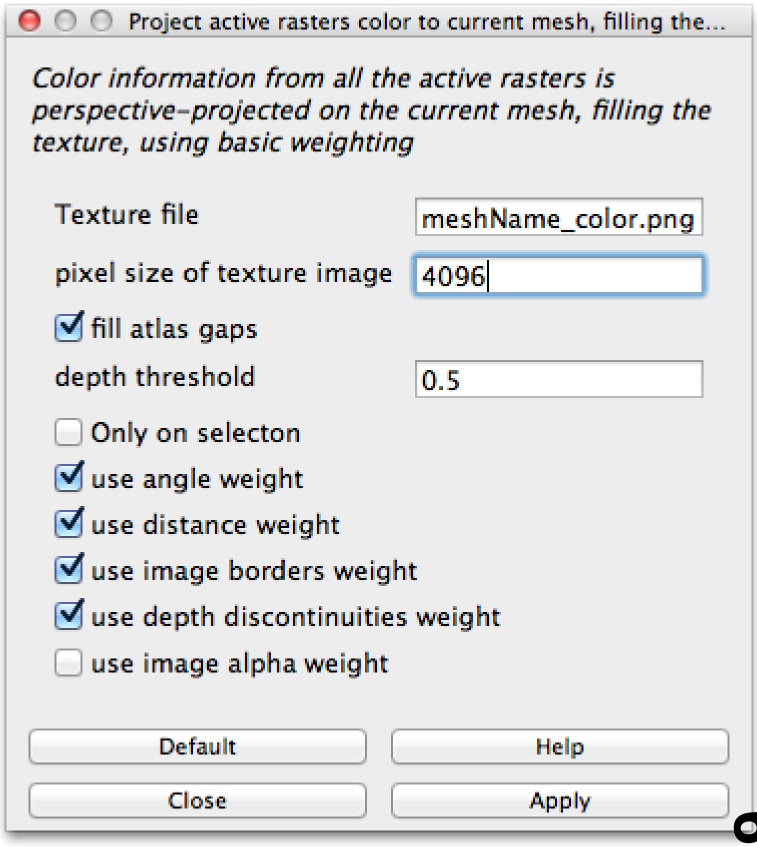
Save (both the project, and the mesh).

Step 8. Project textures

Project the texture from the projectors and make a texture map. Filter –> Texture –> Project active rasters color to current mesh, filling the texture, and match the following settings.

The texture filename you choose will be the name of the texture image that gets written out. The pixel size can be any power of 2 greater than, I'd say, 512. 512 / 1024 / 2048 / 4096 / 8192…the sky (really your computer) is the limit.

Shortcut: Parameterization + texturing from registered rasters



Project active rasters color to current mesh, filling the…

Color information from all the active rasters is perspective-projected on the current mesh, filling the texture, using basic weighting

Texture file: meshName_color.png

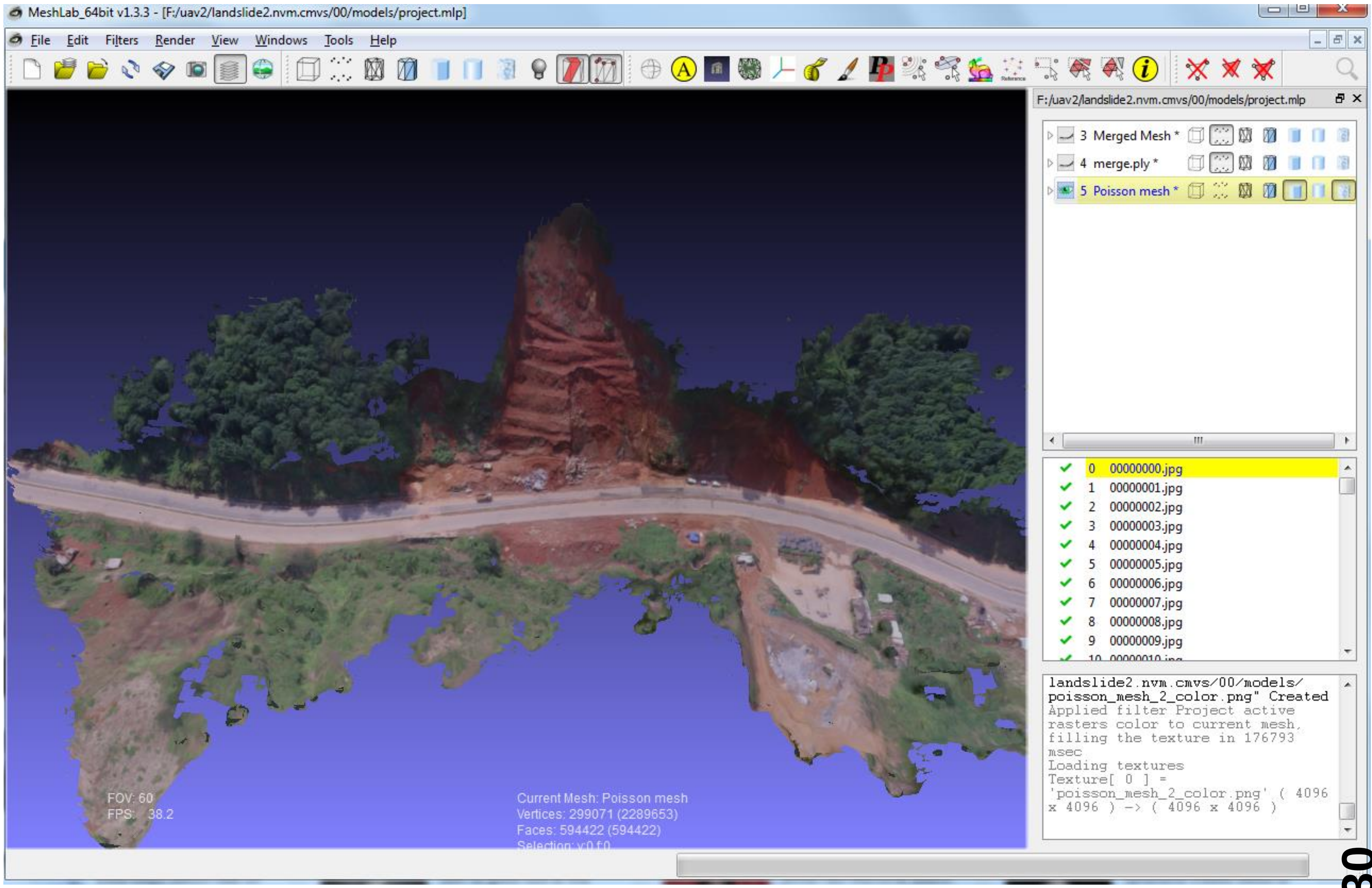
pixel size of texture image: 4096

- [x] fill atlas gaps

depth threshold: 0.5

- [ ] Only on selecton
- [x] use angle weight
- [x] use distance weight
- [x] use image borders weight
- [x] use depth discontinuities weight
- [ ] use image alpha weight

The Textured 3D Model!



Current Mesh: Poisson mesh
Vertices: 299071 (2289653)
Faces: 594422 (594422)
Selection: v:0 f:0

```
landslide2.nvm.cmvs/00/models/
poisson_mesh_2_color.png" Created
Applied filter Project active
rasters color to current mesh,
filling the texture in 176793
msec
Loading textures
Texture[ 0 ] =
'poisson_mesh_2_color.png' ( 4096
x 4096 ) -> ( 4096 x 4096 )
```

2

osmBundler

Dense 3D Point Clouds

osmBundler implements 3 algorithms

- Bundler –performs sparse 3D reconstruction
- PMVS (patch multi view stereo) takes the sparse reconstruction resulted from running Bundler and produces a dense 3D reconstruction.
- CMVS (clustered multi view stereo) which is the same as PMVS, only that is faster and should be used when you process a lot of images.

Installation

- Download and copy osm-bundler on any drive
- Download & Install python 2.7
- install Python Imaging Library

Setup and Running

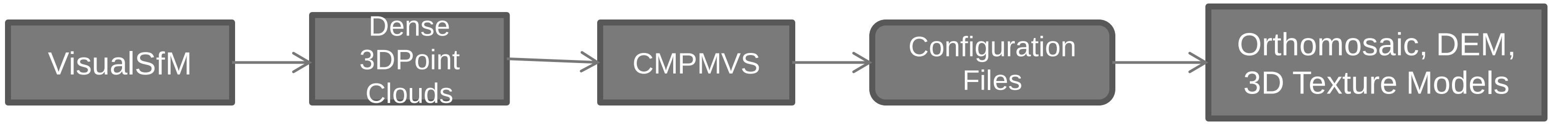
- Open Command Prompt as an Administrator
- *>>cd drive:\<Installation Path>\osm-bundler\osm-bundlerWin32/64*
- insert the camera model and the CCD width to Bundler's camera's database (For NESAC zenmuse, its 6.17 calculated from CMOS sensor size of ½.3")
  - *python RunBundler.py --photos=<full path where your images are located> --checkCameraDatabase*
- Keypoint detection/matching and generation of sparse 3d point clouds
  - *>>python RunBundler.py --photos="<full path where your images are located>"*
- Dense Point Cloud Generation
  - *>>python RunPMVS.py –bundlerOutputPath="<BundlerOutputPath>"*
- Open the Dense Point cloud .ply file in Meshlab (Observer the number of 3Dpoints/vertices)

3

VisualSfM+CMPMVS

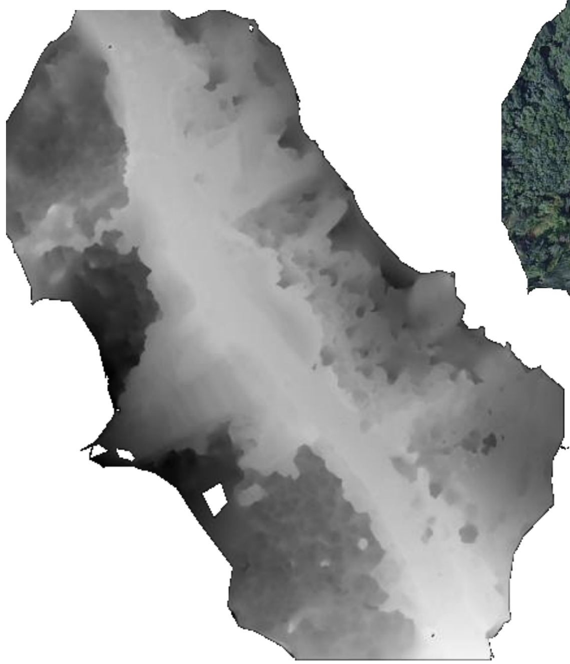
OrthoMosaic, DEM, 3D Textured Surface Models

Process Flow



Issues : It gives DEM data as **DEM.png** which 16-bit data and needs to be converted to 32 bit (float) .tiff where each pixels will represent the pixels value as altitude in meters

CMPMVS is contains computer vision algorithms and run using libraries from OpenCV


DEM

Orthophoto

VisualSfM+CMPMVS

OrthoMosaic, DEM, 3D Textured Surface Models

- Install and Setup visualSfM as usual
- Run VisualSfm till 3D sparse reconstruction
- In VisualSfM, save the dense 3D reconstruction output to folder of type
    - NVM.CMP
- Install CMPMVS version 0.6
    - Configure mvs.ini (output from VisualSfM) according to type of products that needs to be generated
    - Eg. For DEM, set `doComputeDEM=TRUE` ; For, Orthomosaic, set `doComputeOrthomosaic=TRUE`
    - Once .ini file has been configured
        - Open command prompt, Go to CMPMVS installation path
        - Execute >>cmpmvs "path to .ini"

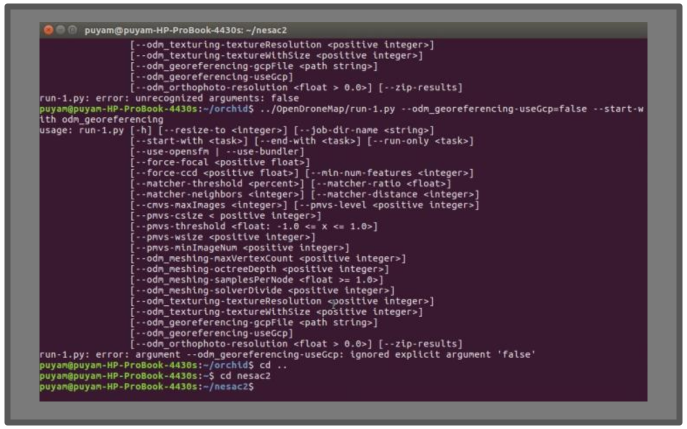
4

OpenDroneMap

Complete UAV data process pipelines in one open ‘box’!

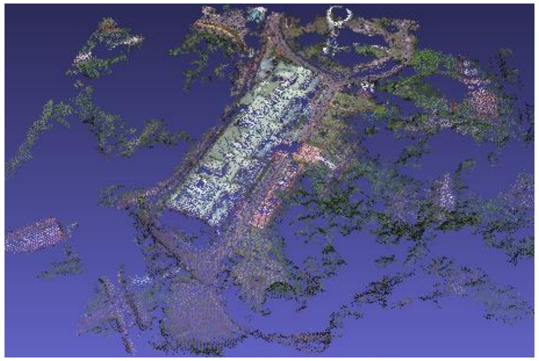
UAV Data Products from ODM

- Point Clouds
- Digital Surface Models
- Textured Digital Surface Models
- Orthorectified Imagery
- Classified Point Clouds
- Digital Elevation Models



```
puyam@puyam-HP-ProBook-4430s: ~/nesac2
                 [--odm_texturing-textureResolution <positive integer>]
                 [--odm_texturing-textureWithSize <positive integer>]
                 [--odm_georeferencing-gcpFile <path string>]
                 [--odm_georeferencing-useGcp]
                 [--odm_orthophoto-resolution <float > 0.0>] [--zip-results]
run-1.py: error: unrecognized arguments: false
puyam@puyam-HP-ProBook-4430s:~/orchid$ ../OpenDroneMap/run-1.py --odm_georeferencing-useGcp=false --start-w
ith odm_georeferencing
usage: run-1.py [-h] [--resize-to <integer>] [--job-dir-name <string>]
                 [--start-with <task>] [--end-with <task>] [--run-only <task>]
                 [--use-opensfm | --use-bundler]
                 [--force-focal <positive float>]
                 [--force-ccd <positive float>] [--min-num-features <integer>]
                 [--matcher-threshold <percent>] [--matcher-ratio <float>]
                 [--matcher-neighbors <integer>] [--matcher-distance <integer>]
                 [--cmvs-maxImages <integer>] [--pmvs-level <positive integer>]
                 [--pmvs-csize < positive integer>]
                 [--pmvs-threshold <float: -1.0 <= x <= 1.0>]
                 [--pmvs-wsize <positive integer>]
                 [--pmvs-minImageNum <positive integer>]
                 [--odm_meshing-maxVertexCount <positive integer>]
                 [--odm_meshing-octreeDepth <positive integer>]
                 [--odm_meshing-samplesPerNode <float >= 1.0>]
                 [--odm_meshing-solverDivide <positive integer>]
                 [--odm_texturing-textureResolution <positive integer>]
                 [--odm_texturing-textureWithSize <positive integer>]
                 [--odm_georeferencing-gcpFile <path string>]
                 [--odm_georeferencing-useGcp]
                 [--odm_orthophoto-resolution <float > 0.0>] [--zip-results]
run-1.py: error: argument --odm_georeferencing-useGcp: ignored explicit argument 'false'
puyam@puyam-HP-ProBook-4430s:~/orchid$ cd ..
puyam@puyam-HP-ProBook-4430s:~$ cd nesac2
puyam@puyam-HP-ProBook-4430s:~/nesac2$
```

Running ODM on Ubuntu



GeoRef Point Clouds

Orthophoto.tif