# 3D Project

## Introduction

A website has been developed to display 3D models of objects which can be rotated and zoomed by users. The website uses a python backend running on the [Django](https://www.djangoproject.com/) web framework and uses [WebGL](http://www.khronos.org/webgl/) to display the 3D models.

## Aim

The purpose of this document is to detail the processes involved in generating a model and uploading it to the website so that it can be viewed.

## Generating a Model

#### Take photos of the object to be modelled

Generation of a point cloud is most successful with photos that are taken accurately and methodically. The following points are advised when taking photos:

* The object should be positioned in an area where you can move around it to take the photos
* The object should remain stationary while taking photos, and you should physically move around it
* The surface the object is placed on should be textured. The texture of the surface is also used for matching during point cloud generation.
* Model reconstruction is more successful with many photos with a lot of overlap between them. In a single loop around the object, 20-30 photos works well.
* Using a tripod, 3 loops around the object works well. The first at approximately eye level to the object, the second at approximately 30˚ to the object, and the last at a high angle to capture the top. If the object has areas with complex geometrical detail, or sharp edges, it is advisable to take extra photos around that area to ensure that the model can be accurately constructed
* The EXIF information must be retained in the photos, as it is used in the mesh generation

### 3D Point Cloud Generation

There are numerous methods to generate a 3D point cloud from the photos. The tools that were assessed during this project were [Arc3D](http://www.arc3d.be/), [123D Catch](http://www.123dapp.com/catch), [Python Photogrammetry Toolbox](http://www.arc-team.homelinux.com/arcteam/ppt.php), and [VisualSFM](http://ccwu.me//vsfm/). Arc3D and 123D Catch are both web based solutions, where you upload your photos to a webservice that asynchronously processes the model and emails you the result. The Python Photogrammetry Toolbox and VisualSFM are both run from your local workstation.

VisualSFM was chosen as the best tool for this purpose due to it producing the most reliable and accurate models. [VisualSFM](http://ccwu.me//vsfm/) is a visual tool that will construct a point cloud on a local computer. It gives a visual representation of the reconstructed point cloud, and also reconstructs the position of the cameras. It allows control over which cameras are included in the generation of the model by deleting cameras that are producing erroneous data. It is free to use for non-commercial purposes, but closed source. It is also possible to script the generation of models, however that removes the ability to review the accuracy of the model while generating the point cloud.

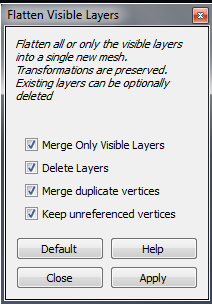
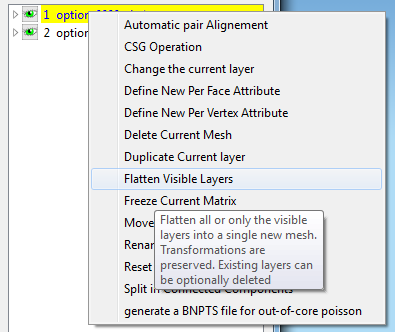
The [Python Photogrammetry Toolbox](http://www.arc-team.homelinux.com/arcteam/ppt.php) also runs on a local computer and runs a lot of the same processes that VisualSFM does, but is open source. It does not provide a visual representation of the model as it is being generated. It has the benefit of being able to be used over remote desktop to run models.

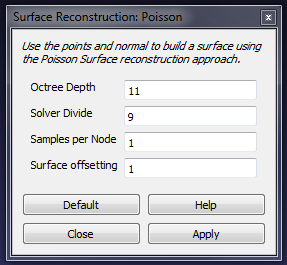
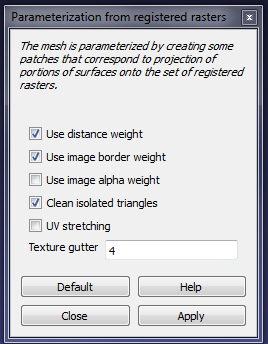
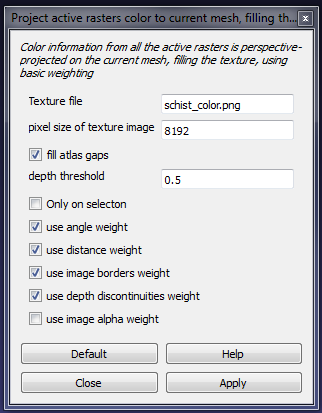
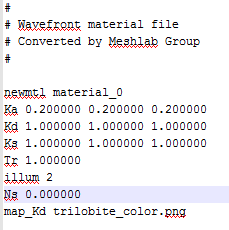
Arc3D (Tingdahl & Van Gool, 2011; Vergauwen & Van Gool, 2006) and 123D Catch provide no user feedback during the production of the point cloud and when there are errors generating the model provide very limited feedback about why the error occurred. They also suffer from being remotely hosted and closed source. If the organisations hosting the applications decide to take them offline, there is a vital part of the workflow missing suddenly.

Generating the 3D point clouds on a local computer is very processor, memory, and graphics processing intensive. The computer used while generating the initial models is a Intel Xeon W3670 (3.2GHz) with 12GB of Ram and a NVIDIA Quadro FX 1800 video card. Depending on the complexity of the model and number of photographs, model generating can take over 4 hours with the CPU maxed at 100% and the entire RAM used. If a workstation was being purchased for the purpose of producing these models, it is recommended to purchase a high end computer with a large amount of RAM. Once the RAM is used, the computer has to start using the hard drive to store data, which is vastly slower than RAM and increases the model processing time significantly.

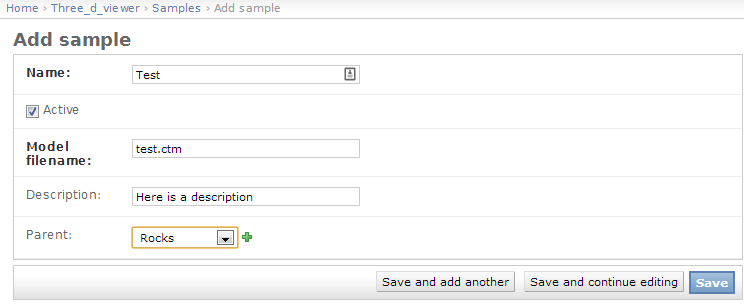
### Meshlab

* Open Project -> Load in the bundle.rd.out file created by VisualFSM, then load the list.txt file.
* This opens the sparse point cloud. To double check the location of the cameras click *Render -> Show Camera.* Click *View -> Show Layer Dialog* to edit the camera scale if necessary to view correctly. Once happy with the camera reconstruction, turn off the cameras, then right click the spare layer and select *Delete Current Mesh*.
* Import the mesh in, *File -> Import Mesh.* The mesh is in the models directory, named option-0000.ply, option-0001.ply etc. If there are multiple meshes, import all of them and flatten them into a single mesh by right clicking within the layer dialog and clicking *Flatten Visible Layers.*



* Clean up the mesh by using select verticies and delete verticies to remove unneeded parts images here. It is a good idea to rotate the image and change the zoom level to check for parts that need to be removed as from some points of view the model can look good, when there are still erroneous faces present. This part of the processing can take considerable time, depending on the quality of the model that VisualSFM produces.
* On models that have taken a lot of time to clean up, it is advisable to export the model as .ply. Meshlab crashes occasionally so that will save having to re-clean it. In the event of a crash, re-load the project files as in step 1 to re-import the camera positions, and then load the .ply mesh you exported instead of the mesh that was created by VisualSFM.
* Currently, what is visible is a point cloud representing the object. A surface for the object needs to be constructed. Click *Filters -> Point Set -> Surface Reconstruction: Poisson.* Change *Octree Depth* to 11, and *Solver Divide* to 9 and apply the filter. It will create a new layer containing the reconstructed surface. 
* Clean up the resulting model by deleting any non-manifold edges. *Filter -> Cleaning and Repairing -> Select non Manifold Edges*, then click apply. If any verticies are selected, delete them with the same delete verticies button used above. If this step is not completed, the next step will not be able to be compelted.
* Generate a UV map of the model. *Filter –> Texture –> Parameterization from registered raster*, using the options shown in figure xxx. 
* Generate a texture of the mesh. *Filter –> Texture –> Project active rasters color to current mesh, filling the texture*, using the following options(choose an appropriate filename for the texture file). 
* Export the model in .obj format. The export produces a .obj file and a .mtl file. Copy the texture file to same location as the exported .obj file(the texture file will be in \00\models\ from where the VisualSFM model is). Meshlab does not include the texture file in the .mtl file, so it must be modified. Open it in a text editor and insert “map\_Kd trilobite\_color.png” at bottom of file(with the correct name for your texture, and without the quotes). 
* Import the .obj model now that it is associated with the texture file.
* The model needs to be simplified to be able to be used efficiently with WegGL. Filters > Remeshing, simplification and construction > Quadratic Edge Collapse Detection(with texture) – 0.25 percent
* Filters -> Texture -> Texture to Vertex Colour
* Export the final model to .ctm format. The CTM format is heavily compressed while maintaining good detail, making it ideal for web based delivery.

## Uploading the Model to the Web Page

* Upload the .ctm model to the /media directory of the 3D Viewer webpage.
* Go to the admin page. Eg. If it is running on your localhost: <http://127.0.0.1:8000/admin/> and enter the admin credentials
* Click on the add icon next to samples, and fill in the details of the model.
* Click Save when complete.

## Acknowlegements

A lot of the details for the workflow came from [We Did Stuff](http://wedidstuff.heavyimage.com/), [Geospatial Modelling and Visualization](http://gmv.cast.uark.edu/modeling/converting-a-3d-model-to-openctm-in-meshlab-for-webgl/) and [Shapeways](http://www.shapeways.com/tutorials/polygon_reduction_with_meshlab).

## References

Cole, Keenan. 2012. Converting a 3D Model to OpenCTM In Meshlab for WebGL.CAST Technical Publications Series. Number 11015. http://gmv.cast.uark.edu/modeling/converting-a-3d-model-to-openctm-in-meshlab-for-webgl/. [Date accessed: 23 August 2013].

David Tingdahl and Luc Van Gool, "A Public System for Image Based 3D Model Generation", Computer Vision/Computer Graphics Collaboration Techniques 5th International Conference, MIRAGE 2011.

Maarten Vergauwen and Luc Van Gool, "Web-Based 3D Reconstruction Service", Machine Vision Applications, 17, pp. 411-426, 2006.