# 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 a complete 360˚ model is required, it is easier to generate a model of each of the halves of the object, and then use a tool such as [CloudCompare](http://www.danielgm.net/cc/) to merge the two meshes together than trying to generate a single mesh. If attempting to generate a single model at once, VisualSFM is the best tool to use, since it gives you a visual representation of the model to give an indication of how accurate the final model will be
* 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. This document will outline 3 methods, each with their own strengths and weaknesses. [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.

Generating the 3D point clouds on a local computer is very processor and memory 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.

[Arc3D](http://www.arc3d.be/) is a web service that allows images to be uploaded and processed on a remote server (Tingdahl & Van Gool, 2011; Vergauwen & Van Gool, 2006). The service sends an email on completion containing links to the generated point cloud and also to a generated model. It has the advantages that there is no processing performed on the local computer, removing the requirement for a high-powered computer and also that processing is asynchronous, allowing multiple models to be submitted for processing concurrently. Like VisualSFM, there is no interaction in the generation of the model, removing control of the reconstruction.

### Arc3D

The workflow for generating a model is as follows:

* Upload the photos to the [ARC3D](http://www.arc3d.be/) web service and wait for the model to be generated
* Import the model into [MeshLab](http://meshlab.sourceforge.net/)
* Convert the textures into vertex colours
* Clean the mesh
* Export the model as an OpenCTM model
* Upload the OpenCTM model to the web server and configure the new model on web site administration page

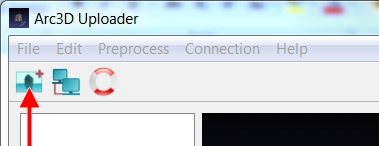


Figure 1. Load images into Arc3D

The details for each step are outlined below.

#### Upload Images

Start the Arc3D uploader and load the photos that you wish to process into the model (Figure 1). After the photos have been loaded into the application, upload them to the Arc3D web service to begin generation of the 3D model (Figure 2).

### 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.
* 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.
* To reproduce the colours onto the reconstructed surface, ensure the Poisson mesh layer is selected(highlighted in yellow) and click *Filters -> Colour Creation and Processing -> Project active rasters to current mesh*. Leave the defaults and select apply. It will take a few minutes to run. When it has completed, change the render mode to smooth to see the final reconstruction.
* Export mesh to .ctm
* Delete manifold edges
* Filter –> Texture –> Parameterization from registered raster
* Filter –> Texture –> Project active rasters color to current mesh, filling the texture
* Export obj
* Copy texture file to same location as .obj
* Modify .mtl -> map\_Kd trilobite\_color.png at bottom of file
* Import obj
* Filters > Remeshing, simplification and construction > Quadratic Edge Collapse Detection(with texture)
* Filters -> Texture -> Texture to Vertex Colour
* Export to ctm

### 3D Scanner

* Using artec studio 9.2
* Scan the object
  + Best to keep the object steady and move the scanner around the object
* In tools, run sharp fusion

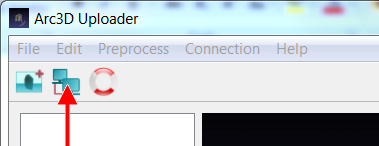


Figure 2. Upload images to Arc3D for processing

## Risks

The key risk in the processing workflow is the [ARC3D](http://www.arc3d.be/) web service. It is not an open source solution, so if the web service is taken offline there is no way to easily replace it.

## 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.