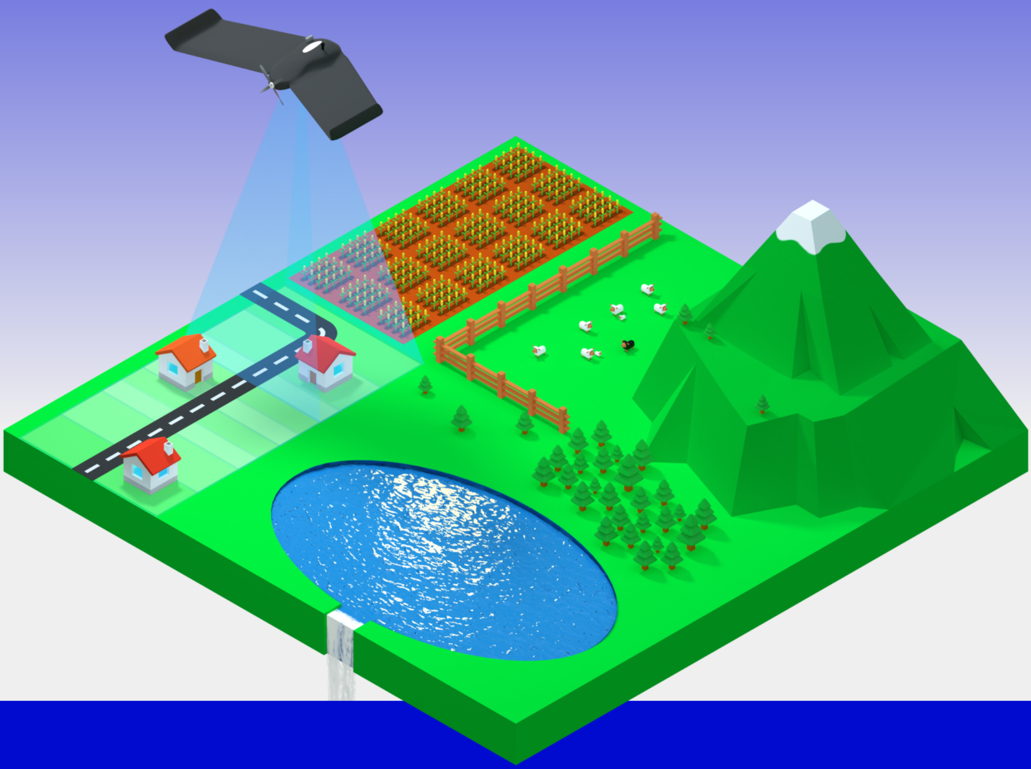
A Practical Guide to Drone Mapping Using Free and Open Source Software



OpenDroneMap

The Missing Guide

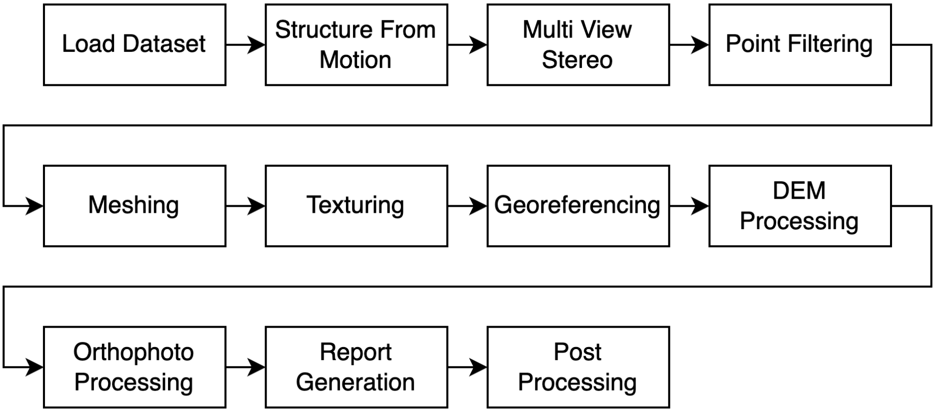
Second Edition

Piero Toffanin

Contents

4. The Processing Pipeline

Going from images to 3D models and orthophotos is a process best visualized as a series of incremental steps. Each step relies on the work of previous steps.

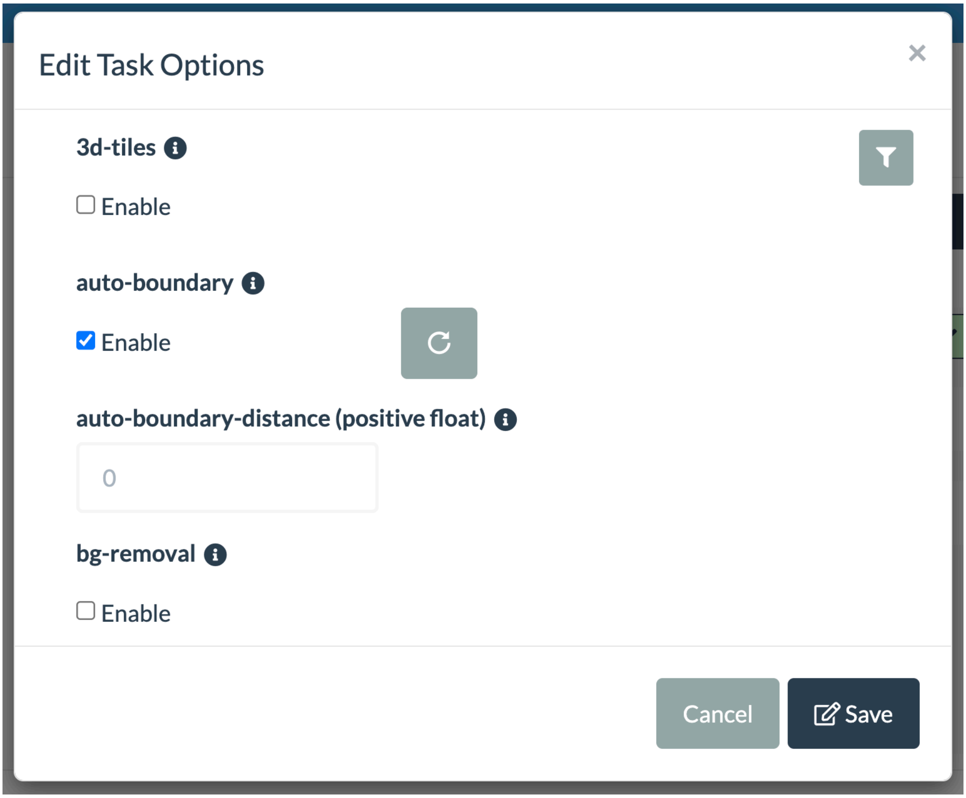


ODM's processing pipeline

In this chapter we will explore an overview of the pipeline. We will not cover too many details, as each step's behavior can be tweaked by changing the task options. We will discuss in depth of how task options affect the inner workings of the pipeline in the next chapter.

5. Task Options in Depth

There are several steps involved in the data processing pipeline. Each step has several adjustable settings that influence the output. The software exposes a subset of these available knobs through various options. When creating a task, a user can choose to tweak one or more options to change the behavior of the pipeline.



Options as shown in WebODM when creating a task

If the list seems overwhelming, just remember that this is a subset of all possible options that could be available from the various steps of the data pipeline. Hidden features and processing capabilities could be hiding in the source code of ODM, in the form of an option not yet exposed! The software exposes only those options that have shown the biggest impact on results, or those necessary to handle different workflows. But many, many more options, under the hood, remain unexposed in order to keep their number somewhat manageable.

Tuning options is more art than science. That's mostly because the best options for certain datasets do not automatically transfer to others. As a general rule, one should start with the defaults, which work fairly well for most datasets and apply tweaks as needed.

This chapter is about understanding in detail what each option does. By the end of the chapter you'll be able to quickly improve your results, explain why certain models turn out the way they do and know what to tweak if the results don't turn out the way you want.

A few of these options might be missing from WebODM and might be available only from ODM. This is because sometimes the option does not make sense in the context of the graphic interface workflow, or it's simply not supported.

When there is some math to explain, I write the formulas using Python because it's easier than math notation and can be typed on a computer. You can copy/paste the code on a website such as online-python.com and follow along even if you don't know Python.

Feel free to jump around and use this chapter as a reference. As the software gets better, some of these options might disappear from future versions and new ones might be introduced. The list below is taken from the software as of June 10th 2023. In alphabetical order:

3d-tiles

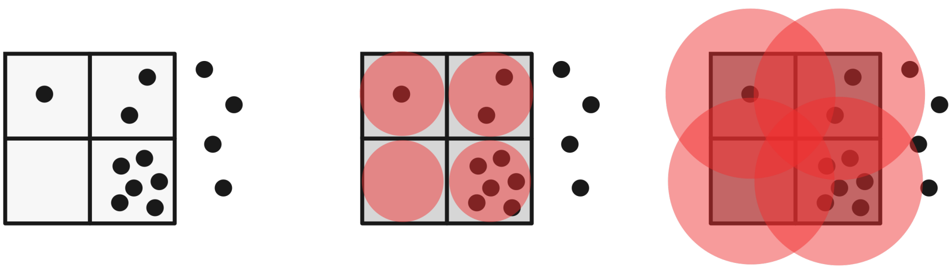
OGC 3D Tiles[1] are a format specification for visualization and interaction with 3D geospatial content. These files can be displayed with software such as the open source virtual globe engine Cesium[2]. ODM has support for generating point clouds

[1] OGC 3D Tiles: ogc.org/standard/3dtiles/
[2] Cesium: github.com/CesiumGS/cesium

Euclidean map results are stored in the *odm_dem* directory.

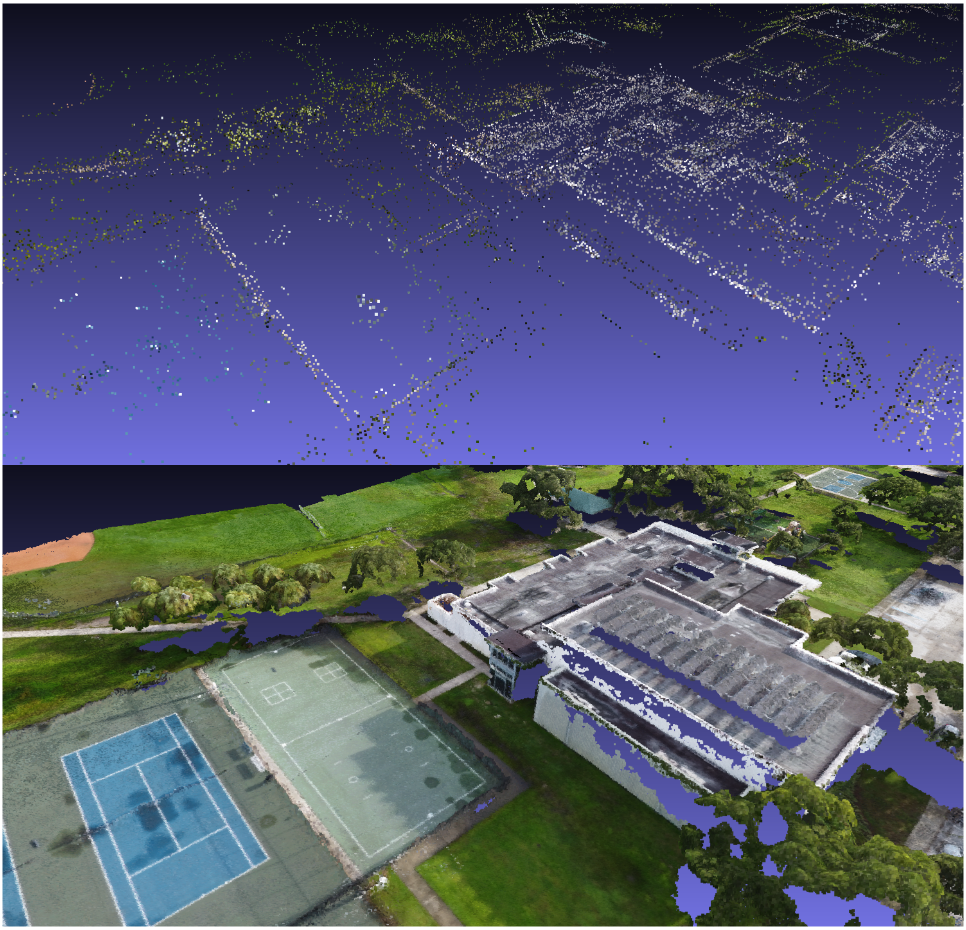
dem-gapfill-steps

The process of going from point cloud to DEMs is not as straightforward as it may seem. Since DEMs are *rasters* (images), they have *cells* (pixels). Each cell, should have a value. Depending on the resolution of the raster, certain cells may have zero, one or more points that fall into it. Every cell needs a value, even if no points fall directly into it, otherwise there will be empty areas (gaps) in the DEM! One way to overcome this is to use a radius around each cell. Every point that falls within the radius is considered part of the cell.



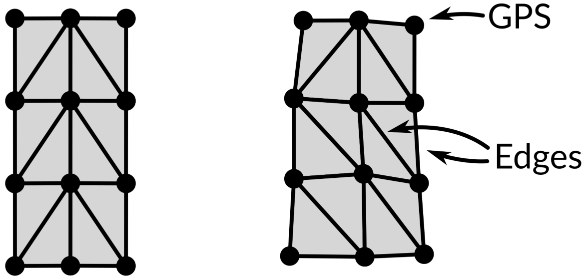
Pixels and points (left), radius of 0.5 (middle) and radius of 1 (right)

But how big should the radius be? If too small, as in the 0.5 radius example above, some cells might remain empty. If too big, there will be too much smoothing and accuracy will suffer. Since different point clouds have varying degrees of density, one solution is to compute multiple DEMs with different radiuses and stack them.



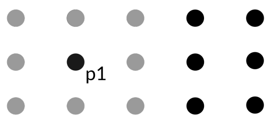
Sparse (top) vs. dense (bottom) point cloud outputs

Both point clouds can be used to generate a mesh. However, it's better to have more points, as meshes can be created with more details. In the dense point cloud screenshot above, the building on the right side of the scene is well defined, but it's poorly represented in the sparse point cloud. Buildings are especially difficult to model without a dense point cloud, so this option tends to yield poor results in



Initial graph (left) and graph with randomly moved positions and new edges (right). Every edge indicates an image pair

ODM also supports a different method to perform preemptive matching by considering only the nearest neighbors of each image instead of using a graph. It is enabled by setting this option. The illustration below shows the result of setting this option to 8:



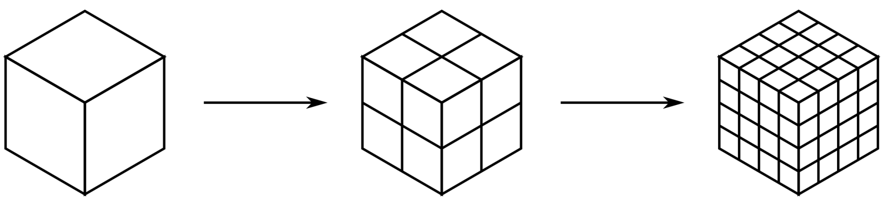
Dots represent approximate image locations, extracted from EXIF tags. When the matcher-neighbors is set to 8, only the 8 nearest neighbors (highlighted in gray) are considered for matching with image p1

This option can sometimes be beneficial for speeding up processing by reducing the number of matching pairs. If no GPS information is available, this option is disabled and all image pairs are considered, unless matcher-order is specified.

mesh-octree-depth

When it comes to generating 3D models, this is probably the most important option. It specifies a key variable for the Screened Poisson Reconstruction[21] algorithm, which is responsible for generating a mesh from the point cloud. The details of the algorithm are fascinating, but probably outside the scope of this book.

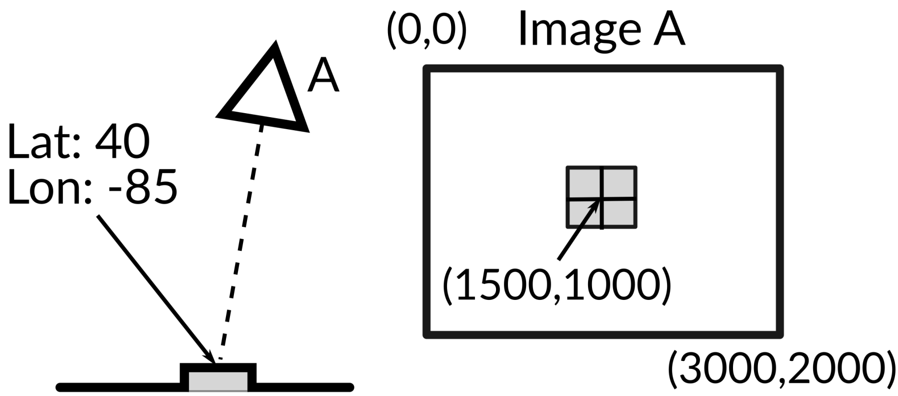
To understand how this option affects the output, it helps to visually understand the concept of an octree. First, octree means *eight-tree* (okta is *eight* in Greek). Why eight? Because at each level (or *depth*) of the tree, each box (or *node or branch*) of the tree is divided in eight parts. At the first level there's only one branch. At the second level there's 8. At the third there's 64 and so forth.



An octree with depth 1, 2 and 3

Lower depths in an octree allow finer details to be captured.

[21] Screened Poisson Reconstruction: watertight surfaces from oriented point sets. cs.jhu.edu/~misha/MyPapers/ToG13.pdf



A GCP marker is photographed by camera A to produce Image A. In a second step, the pixel location of the marker (1500,1000) from Image A can be manually tagged with its real world coordinates (latitude 40, longitude -85)

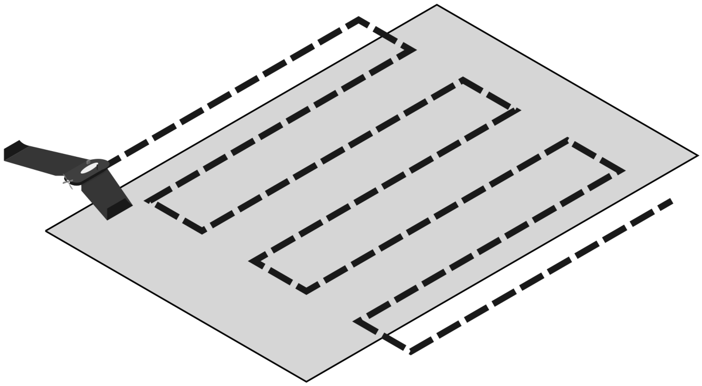
Using ground control points can increase the georeferencing accuracy of a reconstruction, since measurements of static (non-moving) objects using a high precision GPS are often better than those obtained from the GPS of moving UAVs.

The ideal number of ground control points ranges between 5 to 8, placed evenly across the area to be flown. Adding more than 8 ground control points does not necessarily result in increased accuracy.

If the same marker is visible from multiple images, it should be tagged multiple times for each image. Ideally each marker should be tagged at least 3 times. Another way to think of it is to capture each marker on at least 3 images. This is so that the marker's location can be triangulated during computation.

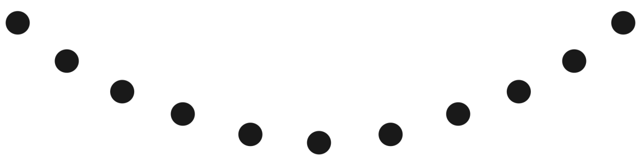
Ground control points can be used by providing an additional text file along with the input images. The file follows a simple format:

The first line indicates the spatial reference system (SRS) of the world coordinates. There are no restrictions on the type of SRS you can use. Internally the program will



Typical flight path from mission planning software. Not great for self-calibration

This doesn't mean a person should never fly this pattern. It just means that when flying this pattern, people need to be aware that the internal camera parameters' estimates will not be as good. Inaccurate parameters lead to an improper camera lens model, which over large areas typically results in a *doming* effect.



Point cloud exhibiting doming. The terrain appears arched instead of straight

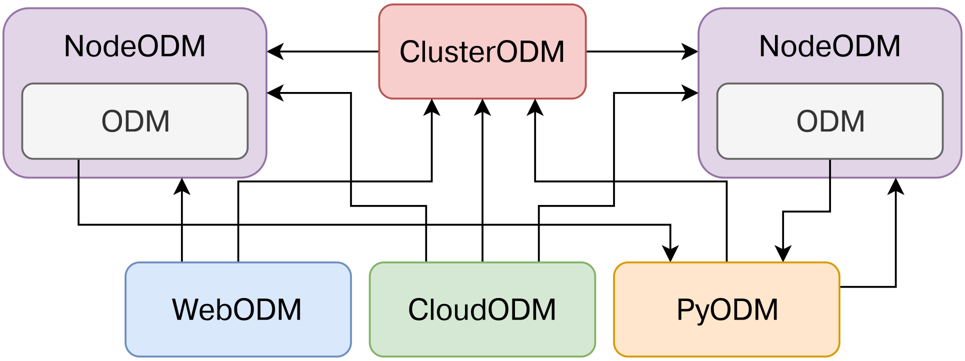
Doming is best cured by following best practices while collecting aerial imagery: flying at different elevations (maximize scale variation) and varying angles.

Unfortunately the luxury of capturing perfect images is not always available. Perhaps a dataset has already been captured and there's no opportunity for a retake,

18. The NodeODM API

ODM is a processing engine and WebODM is a friendly user interface. NodeODM[1] was historically built to allow WebODM to communicate with ODM over a network. Today NodeODM has expanded its role and is the glue that binds together many of OpenDroneMap projects. Each project understands the API that NodeODM defines. When we say *NodeODM* we are referring to the reference implementation of the NodeODM API available at github.com/OpenDroneMap/NodeODM.

At its core, the API defines ways to easily create new tasks, manage such tasks (cancel, delete, restart), download results and query status information.



Many OpenDroneMap projects use the NodeODM API to communicate with each other

[1]Node is a reference to Node.js, the language NodeODM is written in

Glossary

2.5D Model: A model where elevation is simply *extruded* from the ground plane and thus is not a true 3D model.

Artifacts: undesired alterations generated as the result of a digital process.

API: Application Programming Interface. A set of functions allowing the creation of applications that access the features or data of another application.

Bundle Adjustment: a refinement step during the Structure From Motion process that improves the location of cameras, the 3D points of the scene and the camera parameters.

CloudODM: A command line tool to process aerial imagery in the cloud.

ClusterODM: A NodeODM API compatible autoscalable load balancer and task tracker for connecting multiple NodeODM nodes under a single network address.

CRS: Coordinate Reference System. A CRS is a coordinate-based system used to locate geographical entities.

CSV: Comma Separated Value is a textual file format where fields are typically separated by commas or some other character such as a space or a tab.

cURL: a software providing a library and command-line tool for transferring data using many protocols.

DEM: Digital Elevation Model (either a DSM or a DTM).