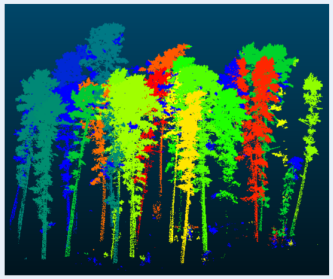
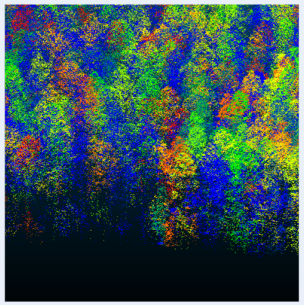
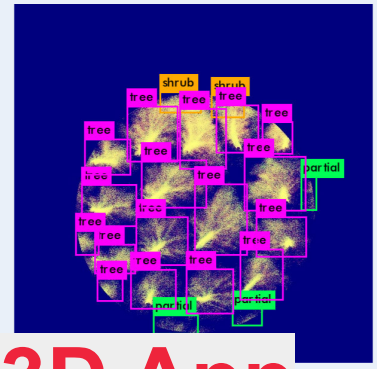
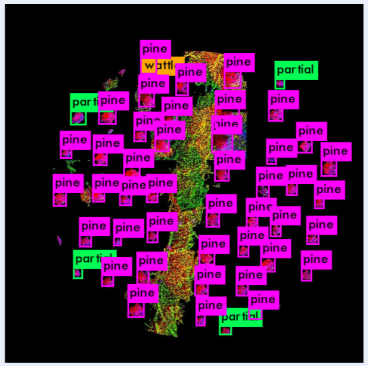
ACFR
AUSTRALIAN CENTRE
FOR FIELD ROBOTICS

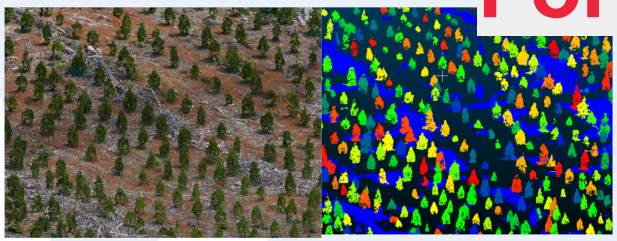


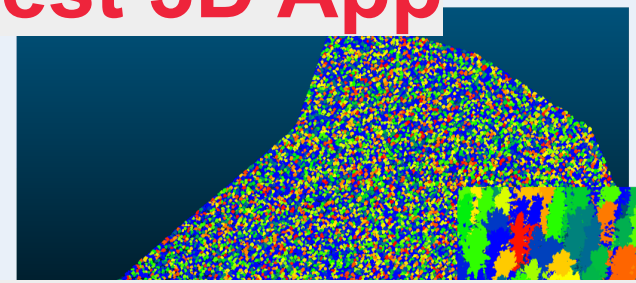


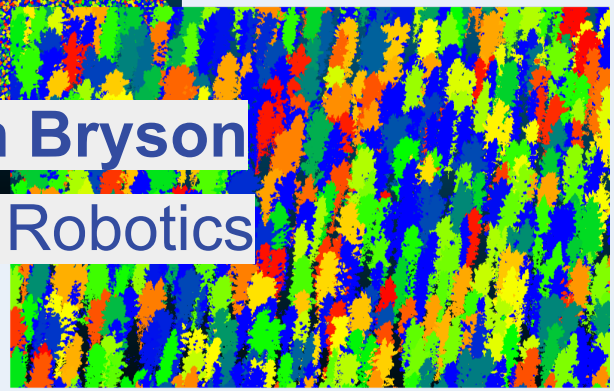




Forest 3D App



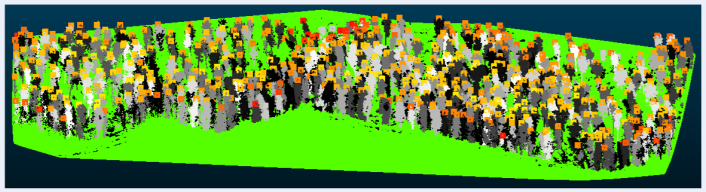




Lloyd Windrim and Mitch Bryson

Australian Centre for Field Robotics

September 2020



What is it?

- Characterise trees in 3D pointcloud data for forest inventory purposes
- A set of python libraries, pre-trained models and example scripts
  - backend: machine learning
  - frontend: simple to use API
- Highly modular libraries – breaks the process down into high-level steps (e.g. reading data, removing ground, detecting trees, exporting inventory)
- beta version: individual tree detection/delineation (tree counts, stem plots, tree heights). Full version will add stem segmentation and stem reconstruction (dbh, taper, diameter profile, branching, etc.)

What is it?

- Characterise trees in 3D pointcloud data for forest inventory purposes
- A set of python libraries, pre-trained models and example scripts
  - backend: machine learning
  - frontend: simple to use API
- Highly modular libraries – breaks the process down into high-level steps (e.g. reading data, removing ground, detecting trees, exporting inventory)
- beta version: individual tree detection/delineation (tree counts, stem plots, tree heights). Full version will add stem segmentation and stem reconstruction (dbh, taper, diameter profile, branching, etc.)

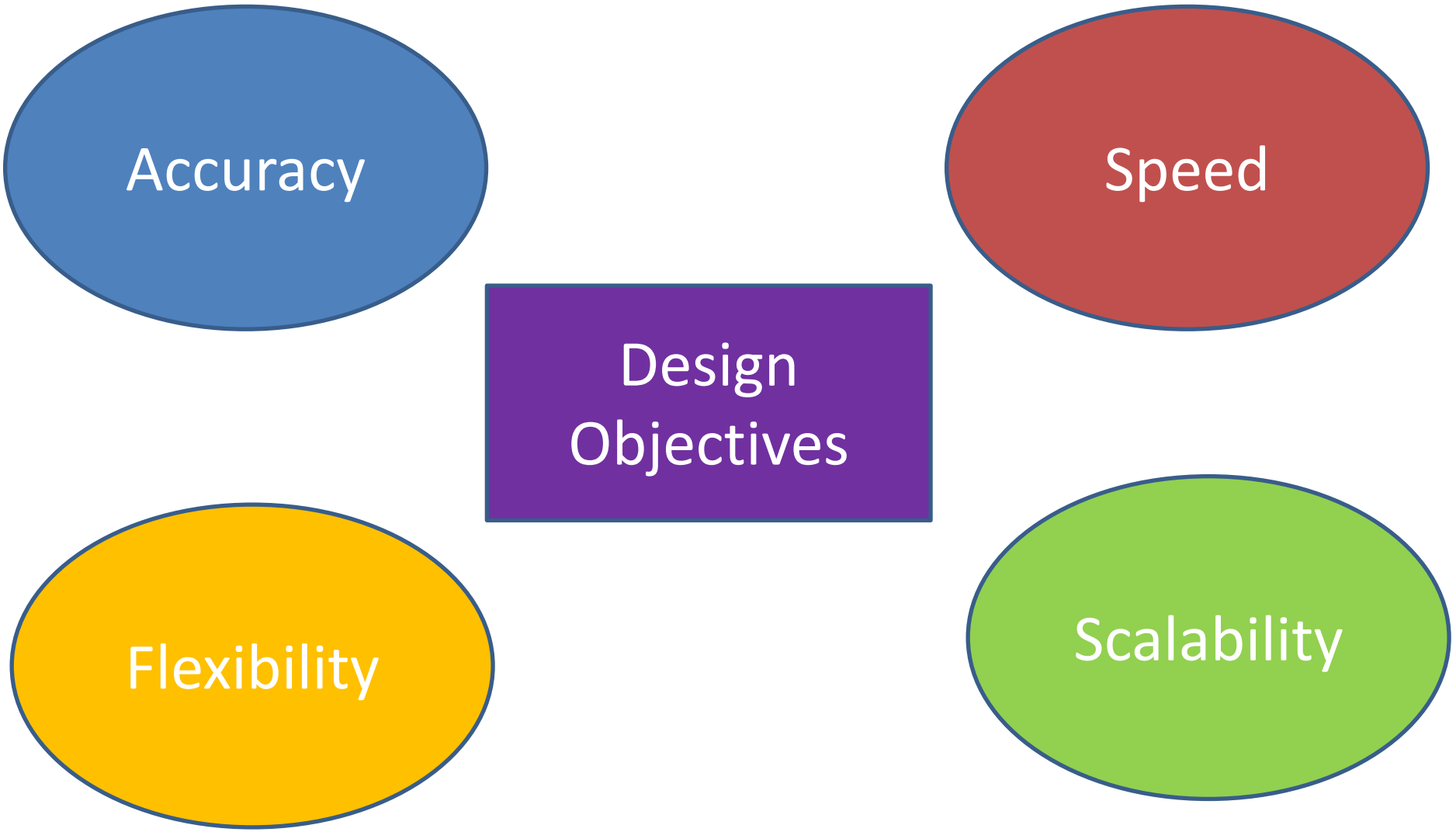
What is it?

- Characterise trees in 3D pointcloud data for forest inventory purposes
- A set of python libraries, pre-trained models and example scripts
  - backend: machine learning
  - frontend: simple to use API
- Highly modular libraries – breaks the process down into high-level steps (e.g. reading data, removing ground, detecting trees, exporting inventory)
- beta version: individual tree detection/delineation (tree counts, stem plots, tree heights). Full version will add stem segmentation and stem reconstruction (dbh, taper, diameter profile, branching, etc.)

What is it?

- Characterise trees in 3D pointcloud data for forest inventory purposes
- A set of python libraries, pre-trained models and example scripts
  - backend: machine learning
  - frontend: simple to use API
- Highly modular libraries – breaks the process down into high-level steps (e.g. reading data, removing ground, detecting trees, exporting inventory)
- beta version: individual tree detection/delineation (tree counts, stem plots, tree heights). Full version will add stem segmentation and stem reconstruction (dbh, taper, diameter profile, branching, etc.)

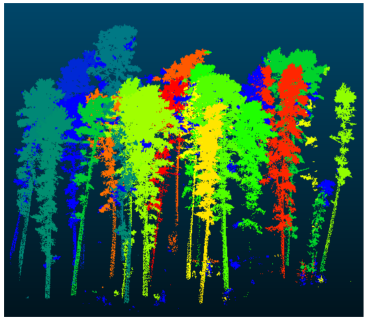
Software Design



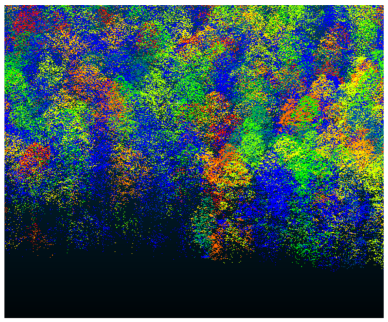
Software Design

- **Accuracy** – state-of-the-art machine learning methods and software tools
- **Speed** – runs on a CPU in reasonable time
- **Flexibility** – easy to reconfigure the software for different pointcloud types (low res or high res ALS, hovermap, photogrammetry)
- **Scalability** – suitable for a range of scan sizes and densities, single plots to whole compartments

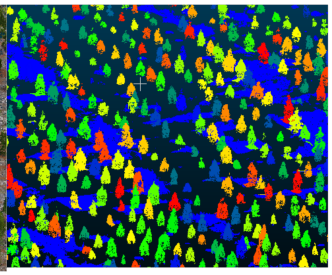
Software Design



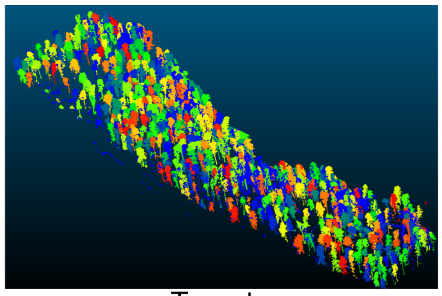
high res ALS



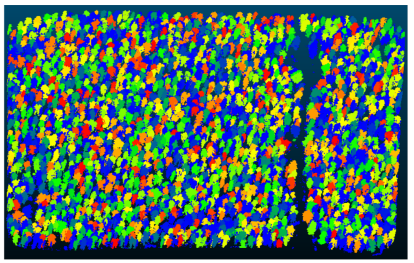
low res ALS



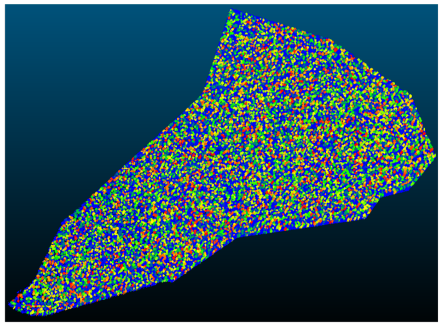
photogrammetry (seedlings)



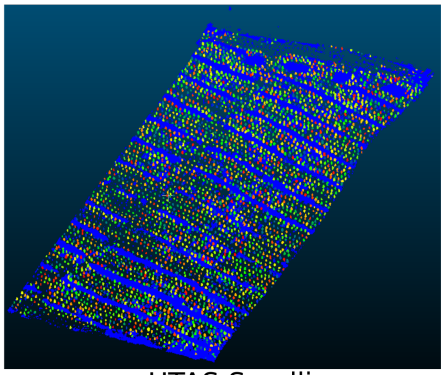
Tumut



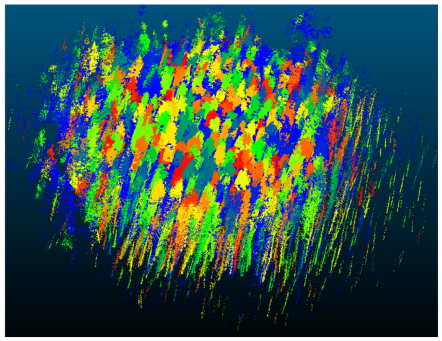
FCNSW



HVP



UTAS Seedling



Rotorua hovermap

How annoying will this be to install?

- Not very!
- Grab the folder from cloudstor/ACFR
- Install python if you haven't already got it
- Install the dependencies for the app e.g:
  - pip install –r requirements.txt
- Done.

- This will let you use the pre-trained models (more on them later)
- If you want to train your own model, you will need to install something called *darknet*

How annoying will this be to install?

- Not very!
- Grab the folder from cloudstor/ACFR
- Install python if you haven't already got it
- Install the dependencies for the app e.g:
  - pip install –r requirements.txt
- Done.

- This will let you use the pre-trained models (more on them later)
- If you want to train your own model, you will need to install something called *darknet*

Do I need a GPU to use this?

- **Nope!** Not if you are happy to use my pre-trained models

- If you want to train your own models, it is recommended you use a GPU
- There are cheaper (but not as convenient) alternatives to owning a computer with a GPU e.g. google collab, AWS, Azure, other cloud services

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

Libraries: what can they do?

- **lidar_IO:** reading and writing pointcloud data into and out of python. Supports a range of formats (e.g. las, ply, asc, csv)
- **processLidar:** useful classes for representing pointcloud data (can voxelise, rasterise, normalize, rotate, etc.)
- **ground_removal**: removing ground points, exporting the ground as a mesh, reading a ground mesh into python
- **treeDetector:** tree detection/delineation, creating raster images from pointclouds
- **inventory_tools:** computing individual tree attributes
- **detection_tools:** functions useful for detection in general

How do you use it

- Just import the forest 3D libraries you want to use into your python script
- There are lots of ready-made example scripts for a tree delineation pipeline:
  - Tumut (Vux-1 ALS)
  - Rotorua (hovermap)
  - UTAS seedlings (photogrammetry)
  - HVP (low res ALS)

How do you use it– example script

Import forest 3d libraries

```
from forest3D import lidar_IO,ground_removal,treeDetector,inventory_tools,utilities
```

Specify an output directory

```
output_dir = '../../outputs'
```

Read pointcloud into python

```
path = '/path/to/pointcloud/tumut.las'
xyz_data = lidar_IO.readFromLas(path, fields = ['x','y','z'])
```

Remove ground points and export as mesh

```
xyz_data_gr = ground_removal.removeGround(xyz_data,offset=[0,0,0],thresh=2.0,proc_path=output_dir)
ground_pts = ground_removal.load_ground_surface(os.path.join(output_dir,'_ground_surface.ply'))
```

Delineate trees

```
detector_addr = '../../models/detection/tumut1'
with open(os.path.join(detector_addr, 'raster_config.json')) as json_file:
    config_dict = json.load(json_file)
rasterTreeDetector = treeDetector.RasterDetector(**config_dict )
labels = rasterTreeDetector.sliding_window(detector_addr,xyz_data_gr,ground_pts=ground_pts,
                                           windowSize = [100,100],stepSize = 80)
```

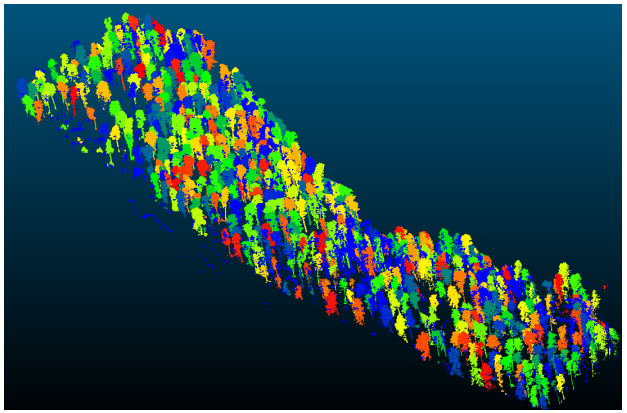
How to use it – example script

Export delineated pointcloud

```
lidar_IO.writePly_labelled(os.path.join(output_dir,'detection.ply'),xyz_data_gr,labels,offset)
```

Compute inventory

```
tree_tops = inventory_tools.get_tree_tops(xyz_data_gr,labels)
heights = inventory_tools.get_tree_heights(tree_tops[:,:3],ground_pts)
inventory = np.hstack((tree_tops,heights[:,np.newaxis]))
utilities.write_csv(os.path.join(output_dir,'inventory.csv'),inventory,header='x,y,z,id,height')
```



inventory_tumut

x	y	z	id	height
571107.849733887	6054842.46695361	554.5799833984	1	43.2697722168
571290.194733887	6054646.61995361	553.8942333984	2	36.5160595703
571338.180983887	6054697.23220361	558.0487333984	3	39.2270170898
571128.768983887	6054760.60395361	555.9419833984	4	38.0502597656
571139.301233887	6054782.28320361	559.4314833984	5	45.8612929688
571230.985483887	6054735.58895361	555.8612333984	6	41.7937285156
571437.103483887	6054657.28495361	559.1669833984	7	33.1992709961

Pre-trained models

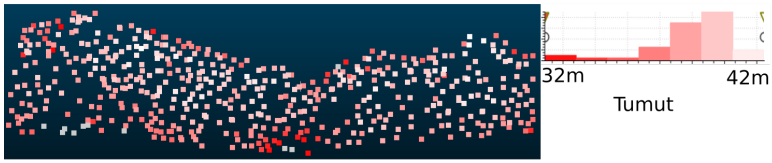
- Shop around and pick the pre-trained model most appropriate for your dataset

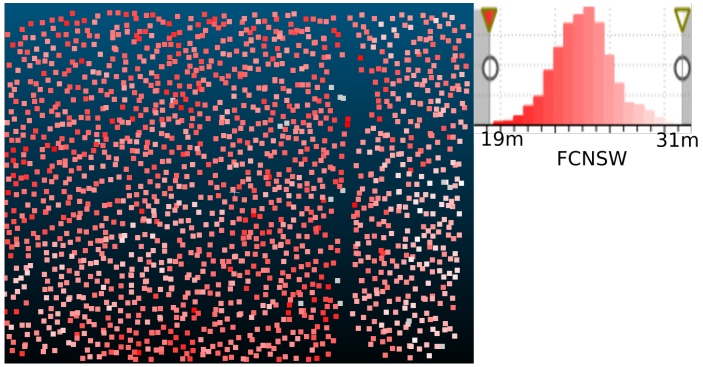
Model name	Dataset used for training	Location	Species/Age	Sensor	Description
tumut1	V1_Scanner1_161011_220153_crop001.las	Tumut NSW	Mature Pine	VUX-1 ALS (helicopter)	High res pointcloud. Only trained on x,y,z data (no return intensity).
tumut2	V1_Scanner1_161011_220153_crop001.las	Tumut NSW	Mature Pine	VUX-1 ALS (helicopter)	High res pointcloud. Trained on x,y,z and return intensity data.
hvp	saltwater_31B_1_2.las	HVP VIC	Mature Pine	? ALS ?	Low res ALS. Only trained on x,y,z data (no return intensity).
utas	transect_large_vis_dense_down5cm.las	Tasmania	Pine Seedling	Drone Photogrammetry	Dense photogrammetric pointcloud. Trained on x,y,z with 2 principle components of red,green,blue for each point.
hovermap	Interpine_02_Output_laz1_2_down5cm.las	Rotorua, NZL	Mature Pine	Hovermap backpack	High res under-canopy data similar to TLS. Trained on x,y,z and return intensity data.

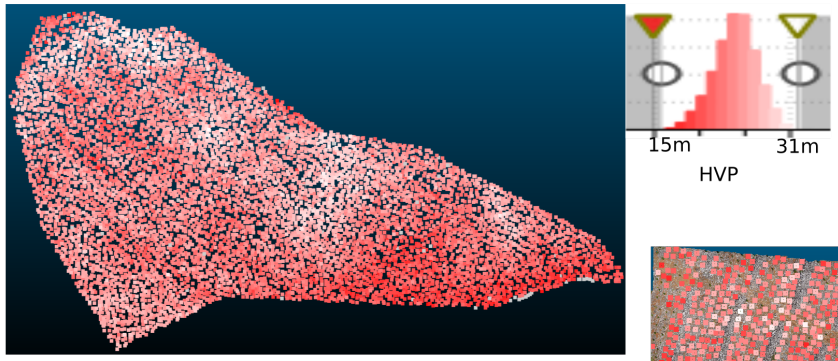
Train your own model

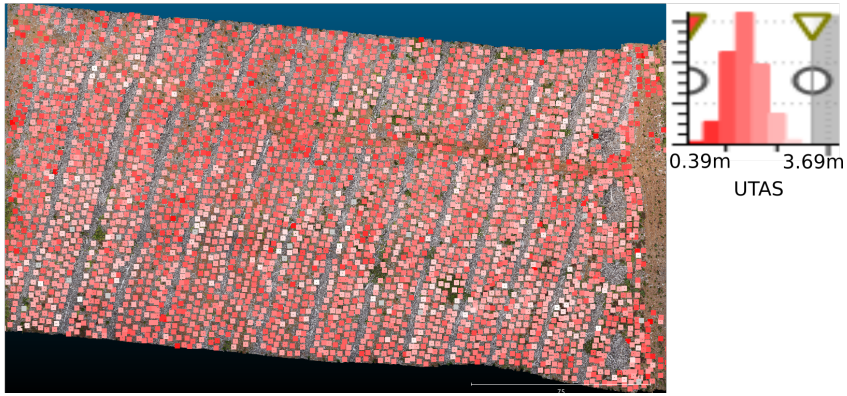
- …or train your own custom model!
- It is a bit more involved (data prep, labelling, using darknet), but instructions and helper code available in the repo

Inventory examples









Inventory examples

Dataset	Filename	Model used	Script	No. trees detected	Typical height range (m)	Mean height (m)	Std height (m)
Tumut (high res ALS)	V1_Scanner1_161011_220153_crop001.las	tumut1	tumut_detect1.py	523	22-49	39.68	6.31
Tumut FCNSW (high res ALS)	FCNSW_FertTrialAOI_VUX1_conv.las	tumut1	tumut_detect1.py	1626	19-33	25.99	2.55
HVP (low res ALS)	saltwater_31B_1_2.las	hvp	hvp_detect.py	8693	15-31	23.95	3.50
UTAS seedlings	transect_large_vis_dense_down5cm.las	utas	utas_detect.py	3618	0.39-3.69	2.18	1.47
Rotorua Hovermap	Interpine_02_Output_laz1_2_down5cm.las	hovermap	hovermap_detect.py	811	-	-	-
Rotorua Hovermap (high res crop)	Interpine_02_Output_laz1_2_down5cm.las	hovermap	hovermap_detect.py	45	32-42	38.93	2.06