Individual Tree Detection

Sam Ericksen, Oren Nardi

2022-03-15

data collected by Oren Nardi, at College of the Redwoods campus in Eureka.

Paper and code snippets by Sam Ericksen

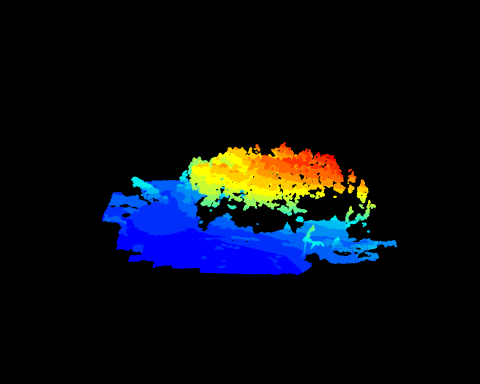
Images processed in WebODM using the defualt “Forest” parameters

library(lidR)  
library(lidRplugins)  
library(raster)  
library(rgdal)  
library(rgl)

## Reading a Point Cloud

First, using the LidR package, we read in the dataset in question. In this case, to make the script more universally available for all users we use the choose.file() function. This function opens a file explorer box that allows the user to choose any file from there computer as they wish. We also want to do a quick initial validity test on the data - simply plotting it.

las <- readLAS(file.choose())



We can also run a deep check on the las to see if any issues are present that may impede future analysis using las\_check(). It’s also nice to check and see if the file is already normalized visually, this can be done by plotting the minimum height values.

## las\_check(las)  
  
epsg(las) ##get the coordinate system of the file. Here we get 32610, which has a vertical unit of meters, so we know the Z values will be in meters

## [1] 32610

## plot(pixel\_metrics(las, ~min(Z), res = 10))  
##   
## min(las@data$Z)  
##

Here both las\_check() and a plot of minimum height values indicate a non-normalized point cloud (all values are above 0 meters). Additionally, the deep check of the file indicates a few interesting anomolies that need to be addressed:

* There are a few duplicate points that should be filtered out
* Ground points are not classified
* There is some issue with the RGB data, but this is likely only an issue for LidR, and can be ignored for now

Duplicates can be filtered very quickly:

las <- filter\_duplicates(las)

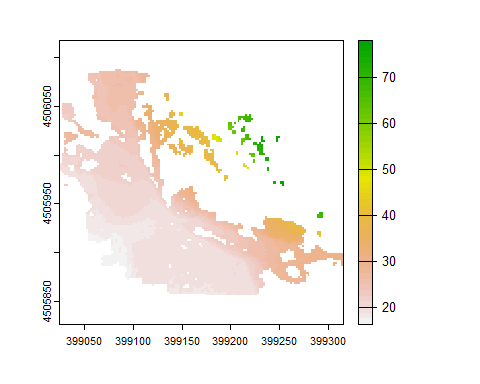
## Classifying Ground Points

When working with point clouds derived from photogrammetry, there is often no classifications initially associated with the points. ASPRS provides a [structure for classification of point clouds](chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/viewer.html?pdfurl=https%3A%2F%2Fwww.asprs.org%2Fwp-content%2Fuploads%2F2010%2F12%2FLAS_Specification.pdf&clen=3783348&chunk=true) to work with, but for now our main concern are ground points. Without ground points classified it is not possible to create a canopy height model, as the canopy height is derived in relation to the ground surface.

When we plotted the point cloud earlier we noticed that there aren’t many points under the canopy, this makes ground classification a little tricky. Fortunately, there were ground points collected on the outskirts of the area of interest, so we can work with that.

Because of the lack of ground under the canopy we will use a Cloth Simulation Filter (Zhang et al. 2016). There are a few attributes that can be adjusted by the user, and the most important in this case is cloth rigidness, although 2L is generally for flat ground, to force some interpretation of the under-canopy ground we will use this setting. Once ground points are classified we plot just the ground points (ASPRS class 2L) to see how it performed.

las <- classify\_ground(las, csf(rigidness = 2L))  
  
plot(pixel\_metrics(las[las@data$Classification==2L], ~mean(Z), res = 2))



plot(las[las@data$Classification==2L])

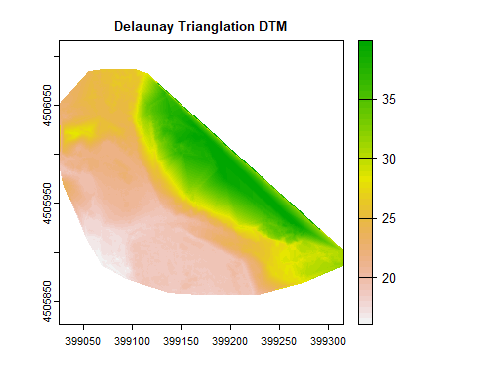
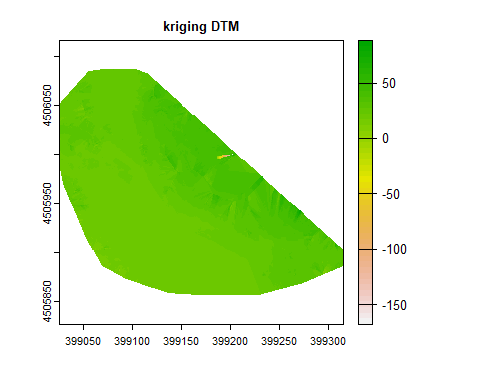
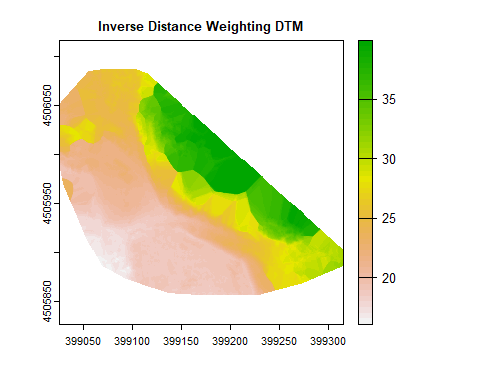
Plotting the average height value of the ground points shows that there are clearly some miss-classified ground points back where the canopy is very dense (Northeast corner). It looks like we can safely remove ground points above 40m during creation of a canopy height model.

## Creating a Canopy Height Model

The first step in creating a canopy height model will be the creation of a digital terrain model (DTM). There are a three methods of this included in the LidR package:

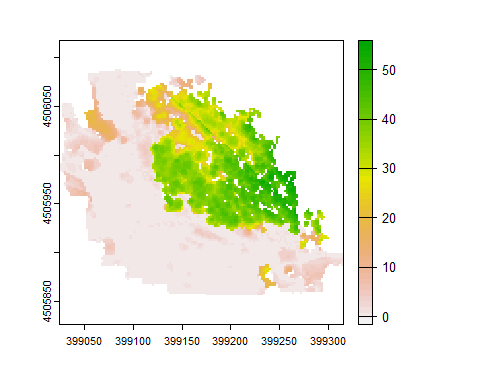
* K-Nearest Neighbor Methods
  + Inverse Distance Weighting
  + Krigging
* Delaunay Triagulation

grnd\_points <- filter\_poi(las, Classification == 2L, Z < 40) ##filter point cloud to include only non-anomalous ground points  
  
idw\_terrain <- rasterize\_terrain(grnd\_points, algorithm = knnidw())  
krig\_terrain <- rasterize\_terrain(grnd\_points, algorithm = kriging())  
tin\_terrain <- rasterize\_terrain(grnd\_points, algorithm = tin())



Here we can see the Kriging does not deal with extrapolation well with default parameters. IDW and TINing seem to give reasonable outputs and a closer inspection seems to indicate that TINing gave a smother slope transition when extrapolating under-canopy ground points. We will continue with the canopy height model using the DTM created using TINing, but one couldn’t be blamed for considering using the IDW terrain model either.

## normalize the point cloud heights  
las\_norm <- normalize\_height(las, algorithm = tin\_terrain)  
plot(pixel\_metrics(las\_norm, ~max(Z), res = 2))



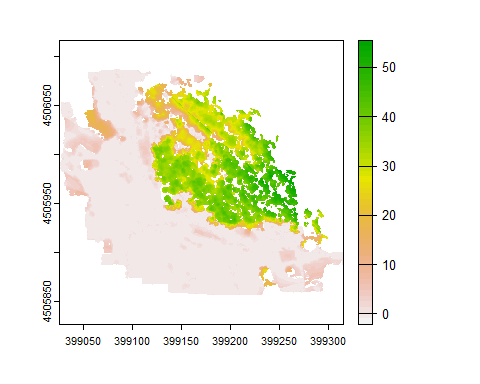
## Classify Noise

Upon close inspection of the dataset there does appear to be some non-contiguous points, specifically near tops of trees. Ground control tree measurements, taken by Oren Nardi, indicate the tallest tree to be around 150’, while the current maximum normalized height is over 170’. It is possible that some of this is due to the

\*Trim extents

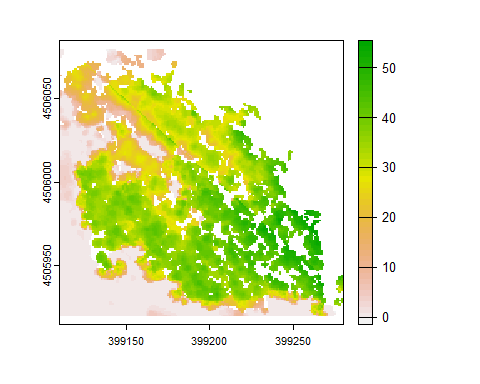
### Trimming Extents

The first thing we want to do is trim the extents of the point cloud to the area of interest. We can use our map from earlier to get some extents from:



Here it looks like the extents of the forest are about (399110 ,4505920, 399280, 4506080)

las\_norm\_clip <- clip\_rectangle(las\_norm, 399110,4505920, 399280, 4506080)  
  
plot(pixel\_metrics(las\_norm\_clip, ~mean(Z), res = 1))



Great, that’s a pretty good cut bounding box for the forested area. Now we need to filter out flat areas, as local maximum filters will pick up slight variations in these areas and miss-classify them as trees.

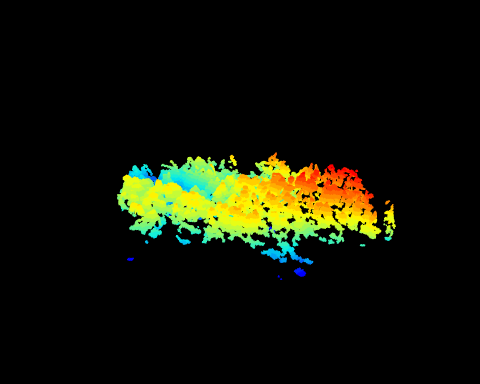
### Removing Non-Tree Values

Looking at the map above it is pretty clear that no trees are under 5m with enough overlap to be captured in photogrammetry. So we can filter those out, and we can also filter out ground points.

las\_Trees <- filter\_poi(las\_norm\_clip, Classification != 2L, Z > 10)

## Warning in snapshot3d(scene = x, width = width, height = height): webshot = TRUE  
## requires the webshot2 package; using rgl.snapshot() instead

## Warning in knitr::include\_graphics(snapshot3d(scene = x, width = width, : It is  
## highly recommended to use relative paths for images. You had absolute paths: "C:  
## \Users\samer\AppData\Local\Temp\RtmpmUyhLu\file526865335dce.png"



Nice, that looks like a bunch of trees! a few outlying points still though, lets see if a noise filter will take care of that.

### Noise Classification

The lidR package supports two methods for noise classification, Statistical Outliers Removal (SOR) which just removes points that are considered outliers by being some multiple of standard deviations from the mean height, or an Isolated Voxels Filter (IVF) which voxelizes the point cloud, and removes cuboids of points which have less than a specified number of points in their surrounding voxels.

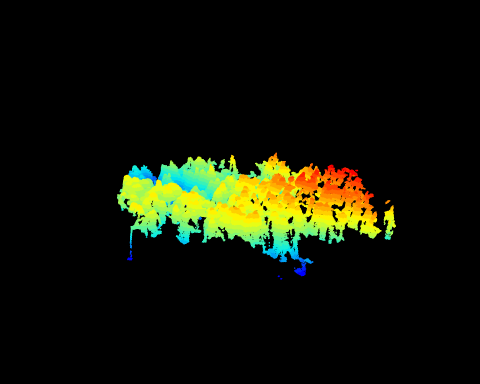
We can see in the previous 3D plot that most of our outlying points are actually detached from from neighbors. This is the method we will use as many of our outlying points are discontiguous from our forest points.

Finally we smooth the cloud out slightly, just as a measure to avoid multiple maxima in small areas (sometimes caused by split crowns or noisy points)

las\_trees\_ivf <- las\_Trees %>%   
 classify\_noise(ivf(res = .5, n = 40)) %>%   
 filter\_poi(las\_Trees@data$Classification!=18)  
  
smooth\_ivf <- smooth\_height(las\_trees\_ivf, 1, 'gaussian', 'circle', sigma = 2)

## Warning in snapshot3d(scene = x, width = width, height = height): webshot = TRUE  
## requires the webshot2 package; using rgl.snapshot() instead

## Warning in knitr::include\_graphics(snapshot3d(scene = x, width = width, : It is  
## highly recommended to use relative paths for images. You had absolute paths: "C:  
## \Users\samer\AppData\Local\Temp\RtmpmUyhLu\file52683758506b.png"



## Individual Tree Detection

(Young, Koontz, and Weeks 2021)

## Still To Come…

* *Individual Tree Detection*
* *Indv. Tree points geometry*
* *plot CHM with individual trees overlaid*

## References

Young, Derek J. N., Michael J. Koontz, and JonahMaria Weeks. 2021. “Optimizing Aerial Imagery Collection and Processing Parameters for Drone-Based Individual Tree Mapping in Structurally Complex Conifer Forests,” September. <https://doi.org/10.32942/osf.io/p7ygu>.

Zhang, Wuming, Jianbo Qi, Peng Wan, Hongtao Wang, Donghui Xie, Xiaoyan Wang, and Guangjian Yan. 2016. “An Easy-to-Use Airborne LiDAR Data Filtering Method Based on Cloth Simulation.” *Remote Sensing* 8 (6): 501. <https://doi.org/10.3390/rs8060501>.