Drone Shrub Volume Subset Test: Individual Shrub Detection and Delineation via CHM and Direct Point Cloud Segmentation

Code: Abhinav Shrestha | Project PI: Georgia R. Harrison

2023-04-25

Contents

Load necessary libraries

```
require(lidR)
require(terra)
require(raster)
require(viridisLite)
require(ForestTools)
require(sp)
require(sf)
require(rgdal)
```

Data preparation: clipping point cloud (subset)

```
pointCloud_toClip <- readLAScatalog("~/PATH/NAME.las")

# check the CRS of the imported las is the same as clipping boundary polygon
pointCloud_toClip$CRS

# import shapefile with which the point cloud data is clipped to
clipBoundary_shp <- st_read(dsn = "~/PATH/FILE.shp", layer = "FILE.shp")

# assuming that 'ID' is an attribute of the shapefile 'clipBounday_shp'
opt_output_files(pointCloud_toClip) <- "~/PATH/NAME_{ID}"

clipped_pointCloud <- clip_roi(las,sf)
```

Creating a Canopy Height Model (CHM) from point cloud

STEP 1: Load point cloud dataset file

```
# load point cloud (.las/.laz) file

las <- readLAS("Outputs\\Clipped_SubsetPC\\sub_subset_PointCloud.las")
```

STEP 2: Classify ground and non-ground points

```
## Sequence of windows sizes
ws <- seq(3,12,3)
## Sequence of height thresholds
th <- seq(0.1,1.5,length.out = length(ws))
## Set threads for classification:
set_lidr_threads(4)
## Classify ground
ground_points <- classify_ground(las, pmf(ws,th))
```

```
writeLAS(ground_points, "Outputs\\Clipped_SubsetPC\\ground_Classif.las")
```

NOTES:

- The `classify_ground'` function takes a very long time to run/process.
  - The issue might be with the algorithm not being parallel-computing friendly (mentioned in the documentation: "In lidR some algorithms are fully computed in parallel, but some are not because they are not parallelizable").
  - Even using the set_lidr_threads to '4' (max available threads), the classify_ground function only uses 1 thread.

- Therefore, the above code chunk consists of a `writeLAS` function as the ground classified point cloud was exported and saved on a local drive so that ground classification would not have to be performed every time the script was run (after the initial run).
- The code chunk below consists of script to import the ground classified point cloud. The same process/logic is applied to the height normalized point cloud and other products produced in this workflow.

```
ground_points <- readLAS("Outputs\\Clipped_SubsetPC\\ground_Classif.las")

plot(ground_points,
     size = 3,
     bg = "white",
     color = "Classification",
     pal = forest.colors(2))
rgl::rglwidget()
```

STEP 3: Create digital terrain model (DTM) with ground classified points

```
## Defining dtm function arguments
cs_dtm <- 1.0 # output cellsize of the dtm

## Creating dtm using Invert distance weighting (IDW)
dtm <- grid_terrain(ground_points, cs_dtm, knnidw())
```

NOTES:

- Min cellsize possible is 0.01, when 0.001 is set, then the grid_terrain function will output `"Error: memory exhausted (limit reached?)" and "Error: no more error handlers available (recursive errors?); invoking 'abort' restart"`
- as resolution of processed DEM was around 0.70 m, cell size set to 1.0 (equivalent to 1 m as native coordinate system is UTM in m)

Plotting created DTM

```
plot_dtm3d(dtm)
rgl::rglwidget()
```

STEP 4: Create height normalized point cloud with DTM

```
hnorm <- normalize_height(ground_points, dtm)
plot(hnorm,
     size = 3,
     bg = "white"
     )

# Export Normalized point cloud .las file
writeLAS(hnorm, "Outputs\\Clipped_SubsetPC\\height_Normalized.las")
```

Reading in and displaying the height normalized point cloud created in the previous code chunk.

```
hnorm <- readLAS("Outputs\\Clipped_SubsetPC\\height_Normalized.las")

plot(hnorm,
     size = 3,
     bg = "white")
rgl::rglwidget()
```

STEP 5: Create CHM using height nomalized point cloud and DTM

```
# Defining chm function arguments
cs_chm <- 0.2 # output cellsize of the chm

## Creating chm
chm <- grid_canopy(hnorm, cs_chm, p2r(na.fill = knnidw(k=3,p=2)))

plot(chm,
     col = col,
     main = "grid_canopy method with IDW fill")

# Exporting CHM
writeRaster(chm,
            'Outputs\\DSM_DTM_CHM_files\\sub_subsetCHM_point02.tif')

## Load CHM
chm <- raster("Outputs\\DSM_DTM_CHM_files\\sub_subsetCHM_point02.tif")
```

NOTES:

- Min cell size seems to be 0.01 since if 0.001 is set "Error: cannot allocate vector of size 1.7 Gb" is returned.
- But having too fine of a cellsize for chm might cause issues in processing time for automatic shrub detection as it uses a moving local maxima filter, i.e., it takes much longer for the kernel to move from pixel to pixel.

Summary of data products generated (from point cloud to CHM)

Intitial point cloud

```
plot(las,
     size = 3,
     bg = "white",
     color = "Z",
     pal = height.colors(25))
rgl::rglwidget()
```

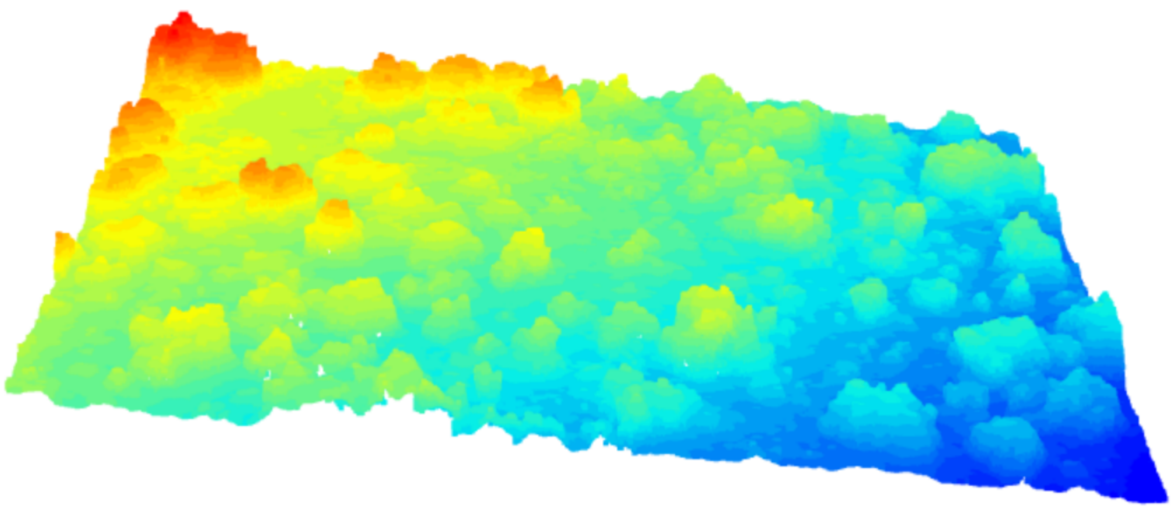

Figure 1: *Point cloud of subset site. Colored by height scale: red = higher elevation, blue = lower elevation*

Ground/non-ground classified point cloud

```
plot(ground_points,
     size = 3,
     bg = "white",
     color = "Classification",
     pal = forest.colors(2))
rgl::rglwidget()
```

Figure 2: *Ground classified subset point cloud*

Digital Terrain Model (DTM)

```
plot_dtm3d(dtm)
rgl::rglwidget()
```

Height normalized point cloud (point cloud with effect of terrain removed)

```
plot(hnorm,
     size = 3,
     bg = "white")
rgl::rglwidget()
```

Canopy Height Model (CHM)

```
plot_dtm3d(chm)
rgl::rglwidget()
```

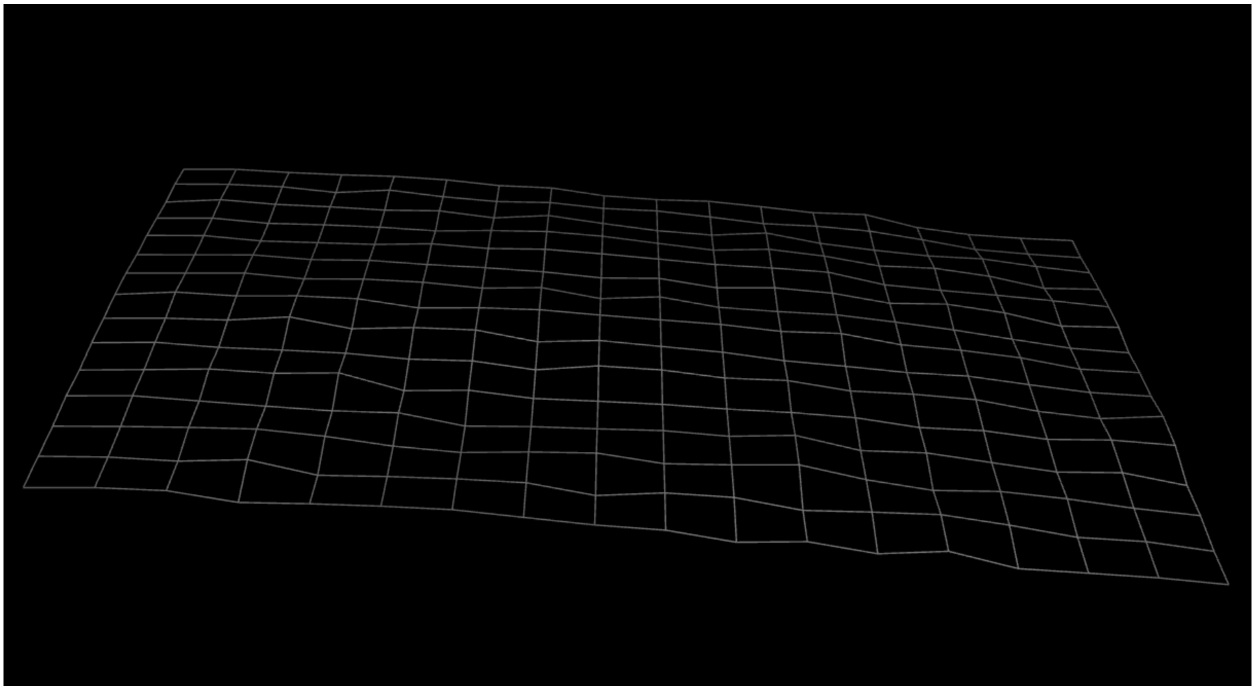

Figure 3: *3D visualization of DTM raster*

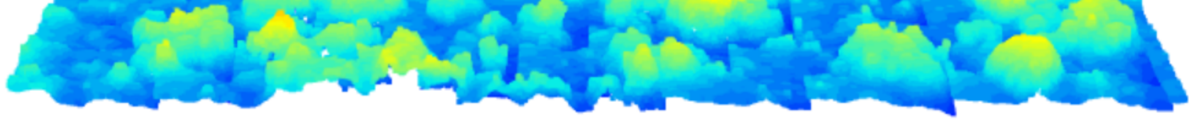

Figure 4: *Height normalized point cloud, Colored by height scale: red = higher relative height, blue = lower relative height*

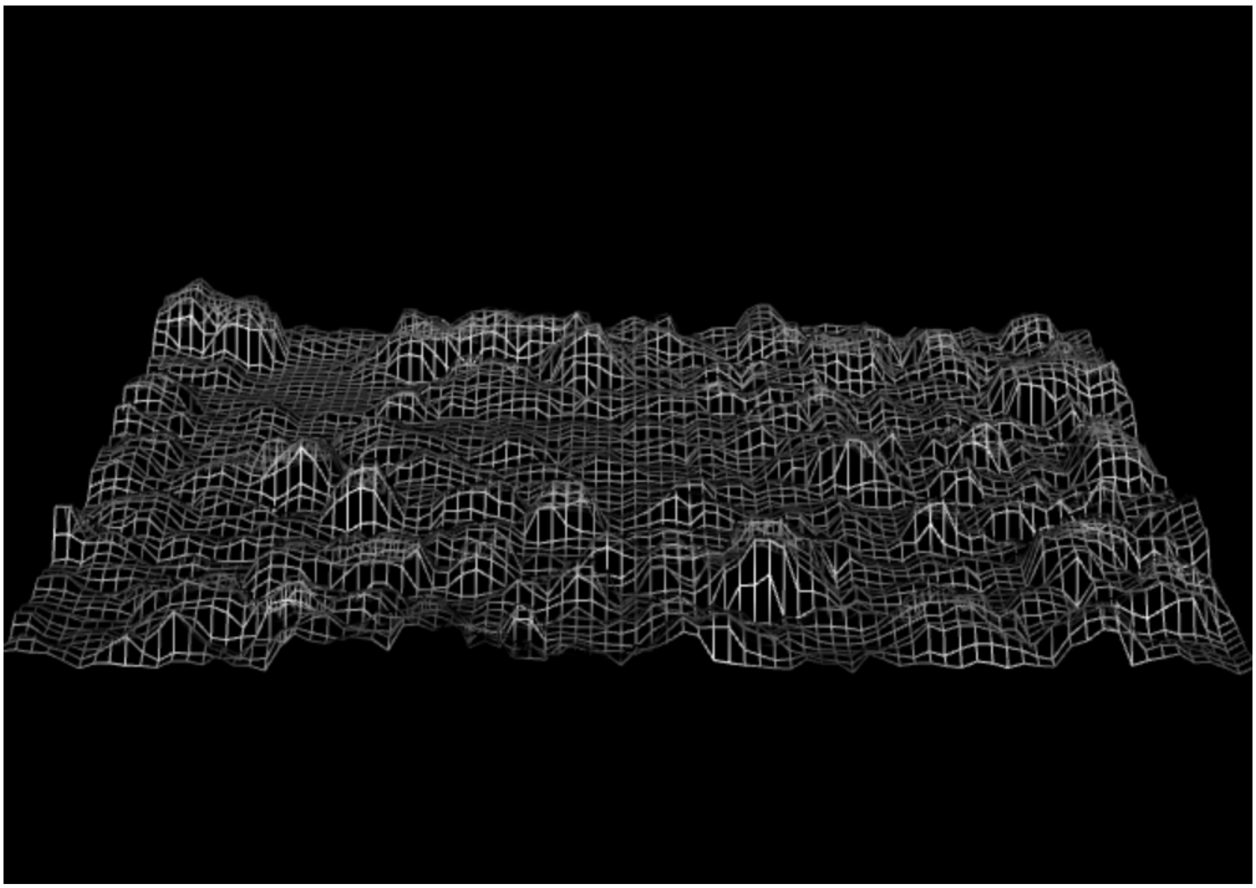

Figure 5: *3D visualization of CHM raster*

Individual shrub detection and segmentation using Local Maxima Filter (lmf, lidR) and Variable Window Filter (vwf, ForestTools)

Smoothing CHM

```
# CHM Smoothing (3x3 kernel, uses mean value)
library(rLiDAR)
schm <- rLiDAR::CHMsmoothing(chm, "mean", 3)
detach("package:rLiDAR", unload = TRUE)
```

Plotting original CHM and Smoothed CHM

```
par(mfrow = c(1,2), mar=c(5, 2.5, 2, 2))
plot(chm,
     col = height.colors(25),
     main = "CHM")
plot(schm,
     col = height.colors(25),
     main = "Smoothed CHM")
par(mfrow = c(1,1), mar=c(5, 4, 4, 2))
```

Individual shrub detection using lmf with lidR

```
shrubtops_lmf <- locate_trees(schm,lmf(2, hmin = 0, shape = "circular"))


plot(schm, col = height.colors(25), main = "ITD with LMF (lidR)")
plot(shrubtops_lmf$geometry, add = TRUE, col='black', pch = 1)

# Exporting individual detected shrubs as a point shapefile (LMF)

# remove Z values (cannot export as point shapefile with Z values)
shrubtops_lmf_shp <- st_zm(shrubtops_lmf)

#exporting as shapefile
st_write(shrubtops_lmf_shp,
         dsn = 'Outputs\\IndividualDetectedShrubs\\shrubtops_lmf.shp',
         driver = "ESRI Shapefile")
```

Individual shrub detection using vwf with ForestTools

```
### Set function for determining variable window radius
winFunction <- function(x){x * 0.04}
```

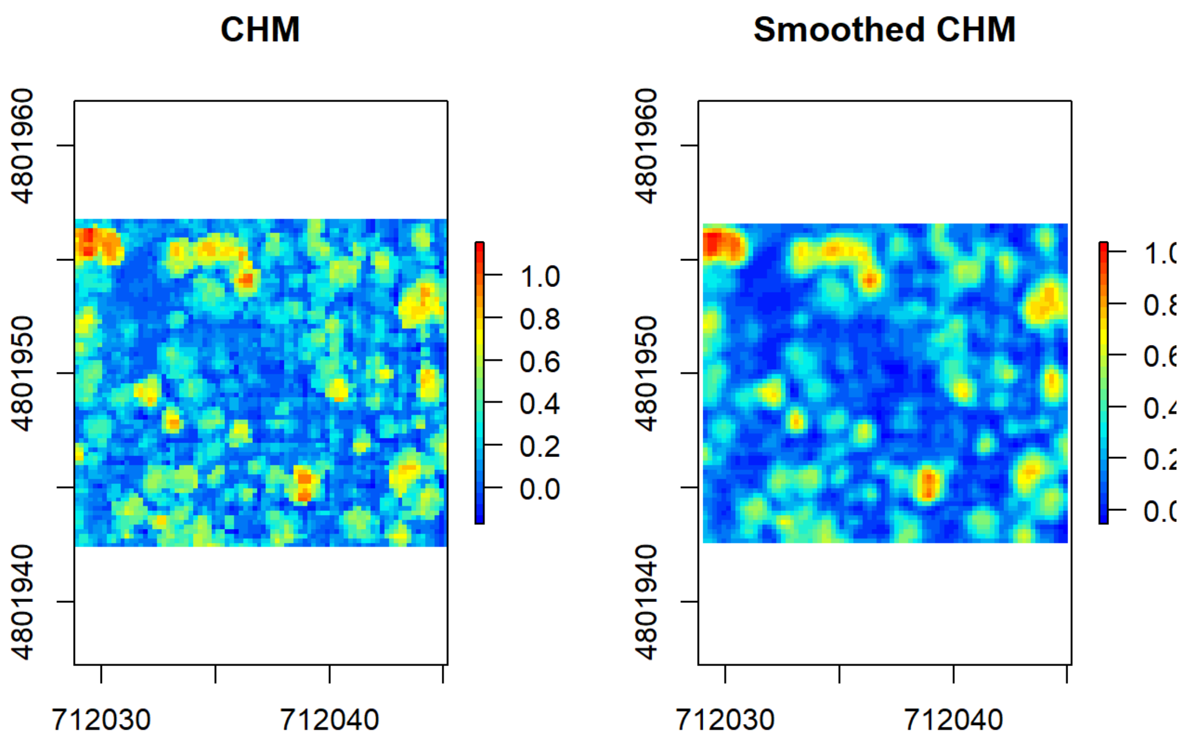

Figure 6: *CHM vs smoothed CHM plots*

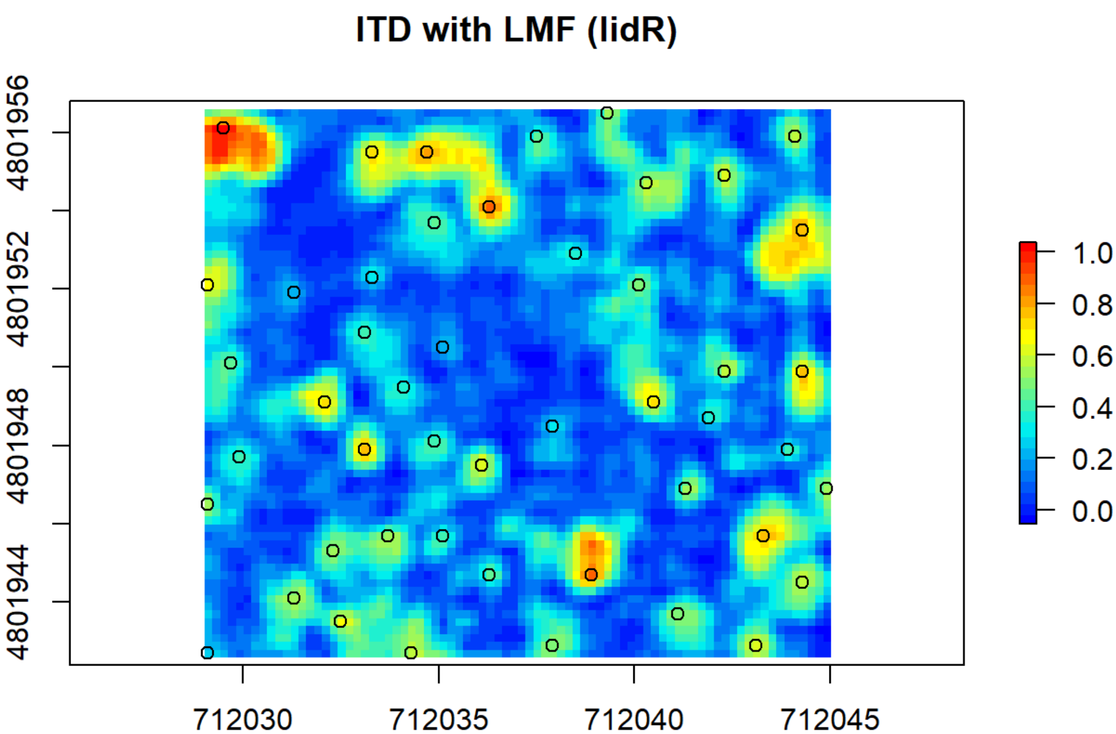

Figure 7: *Individual shrubs detected from LMF method*

```r
### Set minimum shrub height (shrub tops below this height will not be detected)
minHgt <- 0.001

### Detect shrub tops in canopy height model
shrubtops_vwf <- vwf(schm, winFunction, minHgt)

plot(schm, col = height.colors(25), main = "ITD with VWF (ForestTools)")
plot(shrubtops_vwf, add = TRUE, col='black', pch = 1)

# Exporting individual detected shrubs as a point shapefile (VWF)

# Converting spatialpointsdataframe to simple feature to export as shapefile
shrubtops_vwf_shp <- st_as_sf(shrubtops_vwf)

st_write(shrubtops_vwf_shp,
         dsn = 'Outputs\\IndividualDetectedShrubs\\shrubtops_vwf.shp',
         driver = "ESRI Shapefile")
```

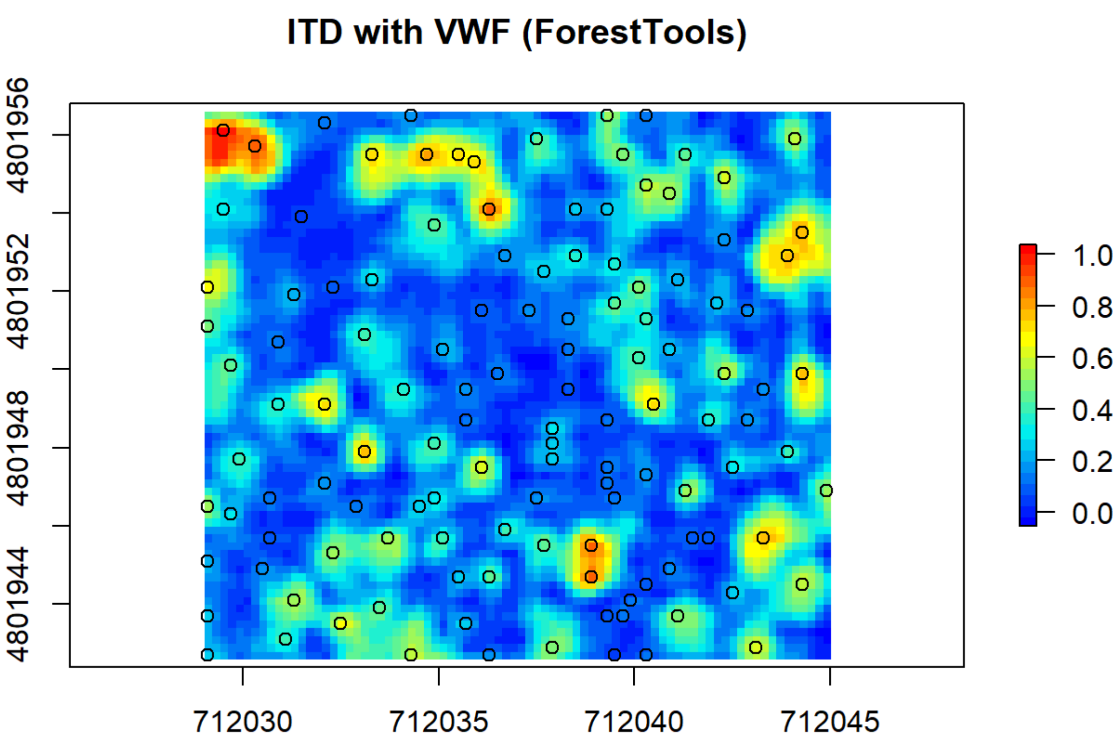

Figure 8: *Individual shrubs detected from VWF method*

Individual shrub delineation using `silva2016` algorithm with `lidR`

```
crowns_silva_lmf <- silva2016(schm,
                              shrubtops_lmf,
                              max_cr_factor = 1.56,
                              exclusion = 0.001)()
crowns_silva_vwf <- silva2016(schm,
                              shrubtops_vwf,
                              max_cr_factor = 1.56,
                              exclusion = 0.001)()

plot(schm, col = height.colors(25), main = "Shrub delineation silva2016 for LMF")
plot(crowns_silva_lmf, add = TRUE, legend = FALSE, col = 'black')

plot(schm, col = height.colors(25), main = "Shrub delineation silva2016 for VWF")
plot(crowns_silva_vwf, add = TRUE, legend = FALSE, col = 'black')
```

NOTE:

- Usually, trees are taller than they are wide. Hence default max crown diameter is set to 0.6 (`max_cr_factor`).
  - i.e., no larger than 60% of the total height of the tree.
- HOWEVER, shrubs are wider than they are tall (field data showed avg height = 56.8 cm vs avg width = 88.4 cm)
  - Therefore, in this case, `max_cr_factor` set by average height to d1 ratio from field data = `1.56`

Converting cell values of raster for export

- cell values with 1 = delineated shrub
- cell values with 0 = non-shrub

```
crowns_silva_lmf[!is.na(crowns_silva_lmf[])] <- 1
crowns_silva_lmf[is.na(crowns_silva_lmf[])] <- 0

crowns_silva_vwf[!is.na(crowns_silva_vwf[])] <- 1
crowns_silva_vwf[is.na(crowns_silva_vwf[])] <- 0

writeRaster(crowns_silva_lmf,
            'Outputs\\IndividualDetectedShrubs\\IndvShrubs_Silva_LMF.tif')

writeRaster(crowns_silva_vwf,
            'Outputs\\IndividualDetectedShrubs\\IndvShrubs_Silva_VWF.tif')
```

Point Cloud based direct segmentation to detect individual shrubs

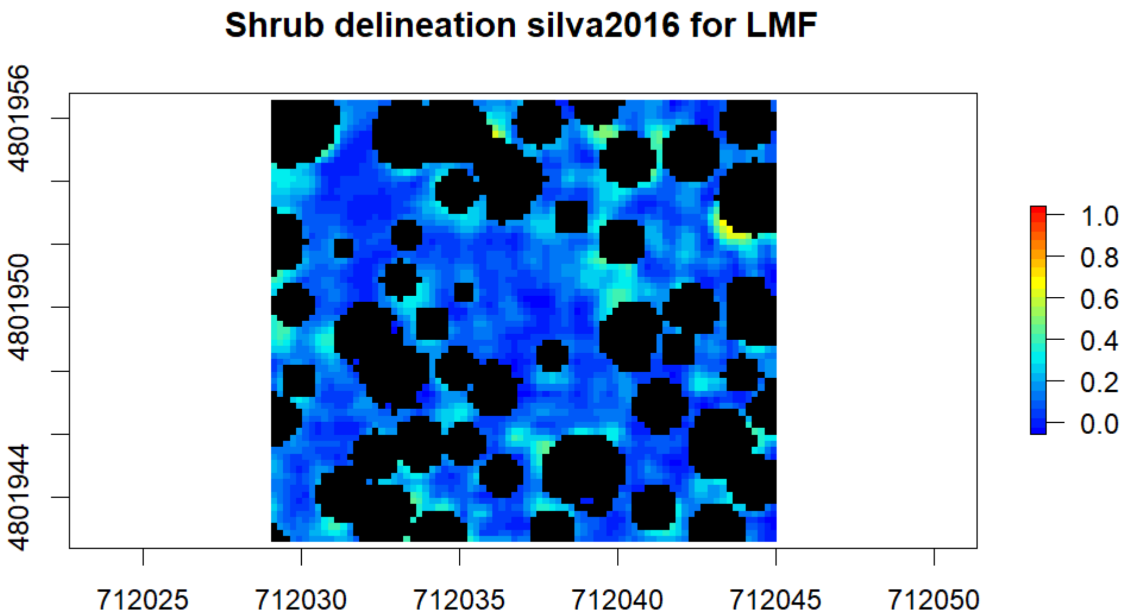

Figure 9: *Individual shrubs delineated from VWF method*

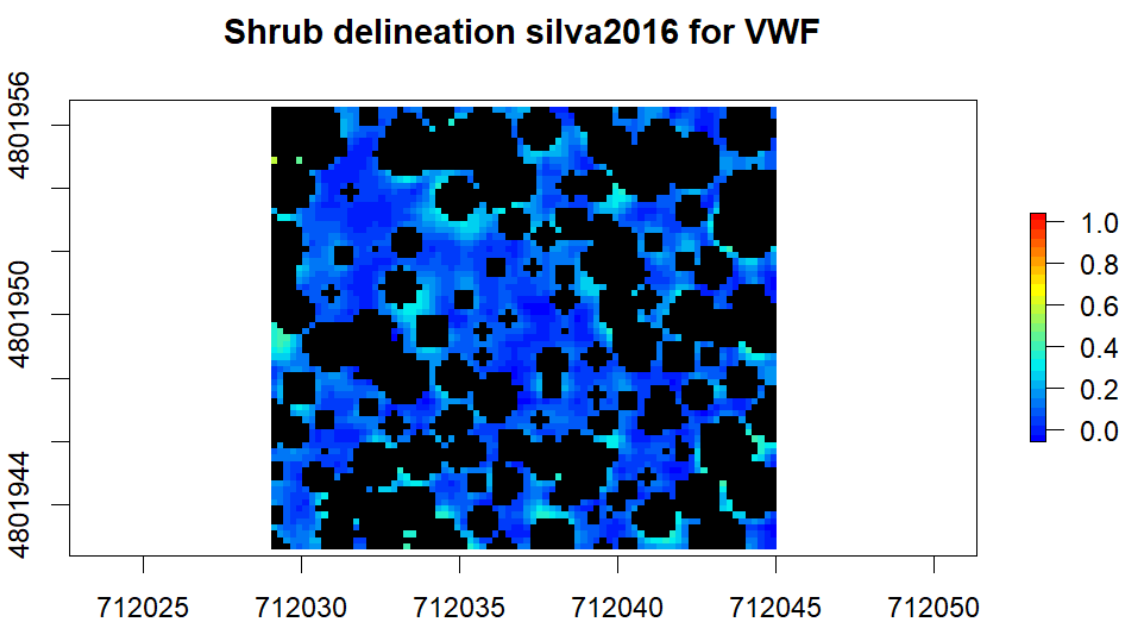

Figure 10: *Individual shrubs detected from VWF method*

```r
# remove ground; 2 = ground, 0 = non-ground
shrubOnly_filterLAS <- filter_poi(hnorm_PC, Classification == 0)

start_time <- Sys.time()
shrub_las_fullPC <- segment_trees(hnorm_PC,
                           algorithm = li2012(R = 0,
                                              dt1 = 1.0,
                                              dt2 = 1.0,
                                              speed_up = 3.5,
                                              hmin = 0.15),
                           attribute = "treeID")
end_time <- Sys.time()
time_taken <- end_time - start_time
print(time_taken)


start_time <- Sys.time()
shrub_las_NonGroundOnlyPC <- segment_trees(shrubOnly_filterLAS,
                           algorithm = li2012(R = 0,
                                              dt1 = 1.0,
                                              dt2 = 1.0,
                                              speed_up = 3.5,
                                              hmin = 0.15),
                           attribute = "treeID")
end_time <- Sys.time()
time_taken <- end_time - start_time
print(time_taken)

length(unique(shrub_las_fullPC@data$treeID)) # 49 segmented shrubs
                                             # total run time = 6.49288 mins

length(unique(shrub_las_NonGroundOnlyPC@data$treeID)) # 48 segmented shrubs
                                                      # total run time = 1.47063

# Exporting
writeLAS(shrub_las_fullPC,
         "Direct_PCSegmentation_IndvShrub\\pointcloud\\ShrubSeg_FullPC.las")

writeLAS(shrub_las_NonGroundOnlyPC,
         "Direct_PCSegmentation_IndvShrub\\pointcloud\\ShrubSeg_GroundRemoved.las")

# Import PC segmented individual shrub point cloud
# (if already run previous code in this code cell)
shrub_las_NonGroundOnlyPC <- readLAS("Outputs\\ShrubDelineation\\ShrubSeg_GroundRemoved.las")

plot(shrub_las_NonGroundOnlyPC,
     size = 3,
     color = 'treeID',
     pal = pastel.colors(50),
     bg = "white")
rgl::rglwidget()
```

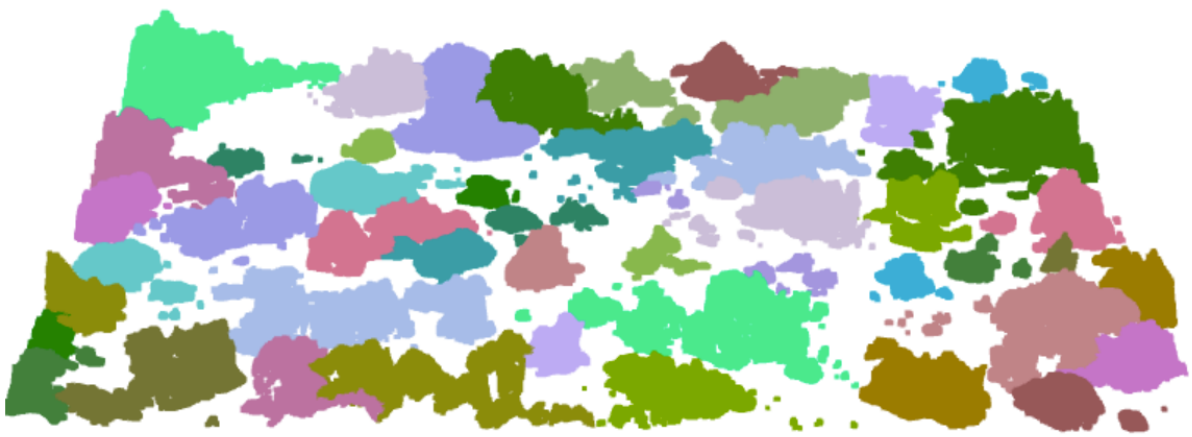

Figure 11: *Individual shrubs detected from direct point cloud segmentation*

Exporting individual shrubs

```
# export point cloud by individual shrub:

## set output directory
out_dir <- "Direct_PCSegmentation_IndvShrub\\pointcloud\\IndvShrubs_NonGroundPCSeg\\"

# removes any NAs
shrub_ID_list <- c(unique(na.omit(shrub_las_NonGroundOnlyPC@data$treeID)))

for (shrub in shrub_ID_list) {
 temp_las <- filter_poi(shrub_las_fullPC, treeID == shrub)
 file <- paste(as.character(shrub), "shrub.las", sep="")
 writeLAS(temp_las,paste(out_dir,file,sep = ""))
}
```

Creating polygon of delineated shrubs (for accuracy assessment)

```
# use "concave" to delineate detailed crowns from PC segments
shrubs_outline <- delineate_crowns(shrub_las_NonGroundOnlyPC,
                                   attribute = "treeID",
                                   type = "concave",
                                   concavity = 2)
```

```r
spplot(shrubs_outline, "treeID",
       col.regions = random.colors(length(unique(shrub_las_NonGroundOnlyPC@data$treeID))))


# Export as shapefile for accuracy assesment in ArcGIS

writeOGR(obj = shrubs_outline,
         dsn="Direct_PCSegmentation_IndvShrub\\IndvDelineatedShrub_shp\\IndvDelineatedShrub_PC.shp",
         layer="IndvDelineatedShrub_PC",
         driver="ESRI Shapefile",
         overwrite_layer = TRUE)

# OPTIONAL: convert outline vector to raster

require(fasterize)

shrubs_outline_sf <- st_as_sf(shrubs_outline)
shrubs_outline_raster <- raster(shrubs_outline_sf, res = 0.1)

shrubs_raster <- fasterize(shrubs_outline_sf, shrubs_outline_raster, field = NULL)

plot(shrubs_raster)

writeRaster(shrubs_raster,
            'Direct_PCSegmentation_IndvShrub\\raster\\DelineatedShrubs_PC.tif',
            overwrite = TRUE)
```

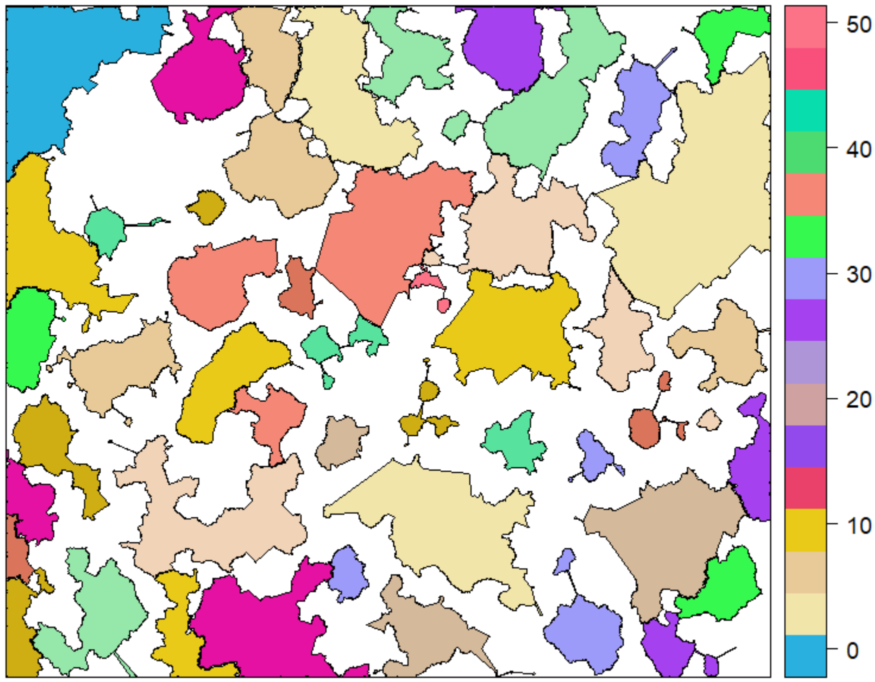

Figure 12: *Individual shrubs delineated from point cloud segmentation method*

References

- Creating a CHM from point cloud (TUTORIAL): https://r-lidar.github.io/lidRbook/chm.html
- Publication on tutorial for individual tree detection using point cloud data:
  - Main paper: https://www.degruyter.com/document/doi/10.1515/geo-2020-0290/html?lang=en
  - Tutorial in supplementary material: https://www.degruyter.com/document/doi/10.1515/geo-2020-0290/downloadAsset/suppl/geo-2020-0290_sm.pdf

Roussel, J.-R., Auty, D., Coops, N.C., Tompalski, P., Goodbody, T.R.H., Meador, A.S., Bourdon, J.-F., de Boissieu, F., Achim, A., 2020. lidR: An R package for analysis of Airborne Laser Scanning (ALS) data. Remote Sensing of Environment 251, 112061. https://doi.org/10.1016/j.rse.2020.112061

Plowright, A., Roussel, J.-R., 2021. ForestTools: Analyzing Remotely Sensed Forest Data.

Pebesma, E., 2018. Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal 10, 439–446.