Overview

Tristan Dyer

In this assignment we'll import, analyze, and explore LIDAR point cloud data for the Mid Pines area, we'll compute bare earth and canopy surfaces from the LIDAR data, and use visualization techniques to highlight subtle terrain features. We'll cover two different ways of bringing LIDAR data into GRASS GIS, weighing the pros and cons of each.

A number of commands will be run from the command-line in this assignment. When starting GRASS GIS, a shell window will be opened. This is the shell window that I will be running most GRASS modules from. Commands run from this window are indicated by a callout box with a green border, as shown here.

```
tristan@tristan-PC:$ echo "Hello, World"
Hello, World
```

Quick Data Exploration

First, we'll take a quick look at the data we're going to be using in this assignment. We can easily explore LAS files (doc) using the libLAS utility applications which are included with GRASS GIS. Use the `lasinfo` utility program to display information about the data we're using.

```
tristan@tristan-PC:$ cd /path/to/data/directory
tristan@tristan-PC:$ lasinfo mid_pines_spm_2013.las
```

This will print a lot of information about the file, most of which is stored in the file header. We are interested in the both the bounding box and the number of points by classification.

```
Minimum and Maximum Attributes (min,max)
------------------------------------------------------------
Min X, Y, Z: 636271.27, 218694.44, 88.73
Max X, Y, Z: 637795.27, 220218.44, 155.81
Bounding Box: 636271.27, 218694.44, 637795.27, 220218.44
```

```
Point Classifications
------------------------------------------------------------
```

```
1340658 Unclassified (1)
2580704 Ground (2)
66 Low Point (noise) (7)
1960603 Reserved for ASPRS Definition (11)
```

We want to set the GRASS region to be equal to that of the data (i.e. the bounding box of the data). The `r.in.lidar` (doc) module can print out a region based on the bounding box of the data, which can then be copy/pasted into the `g.region` module to set the region.

```
tristan@tristan-PC:$ r.in.lidar -sgo input=mid_pines_spm_2013.las
Over-riding projection check
n=220218.440000 s=218694.440000 e=637795.270000 w=636271.270000
b=88.730000 t=155.810000
```

We're interested in the N/S and E/W bounding box, and will only need 1m accuracy, so set the region as follows:

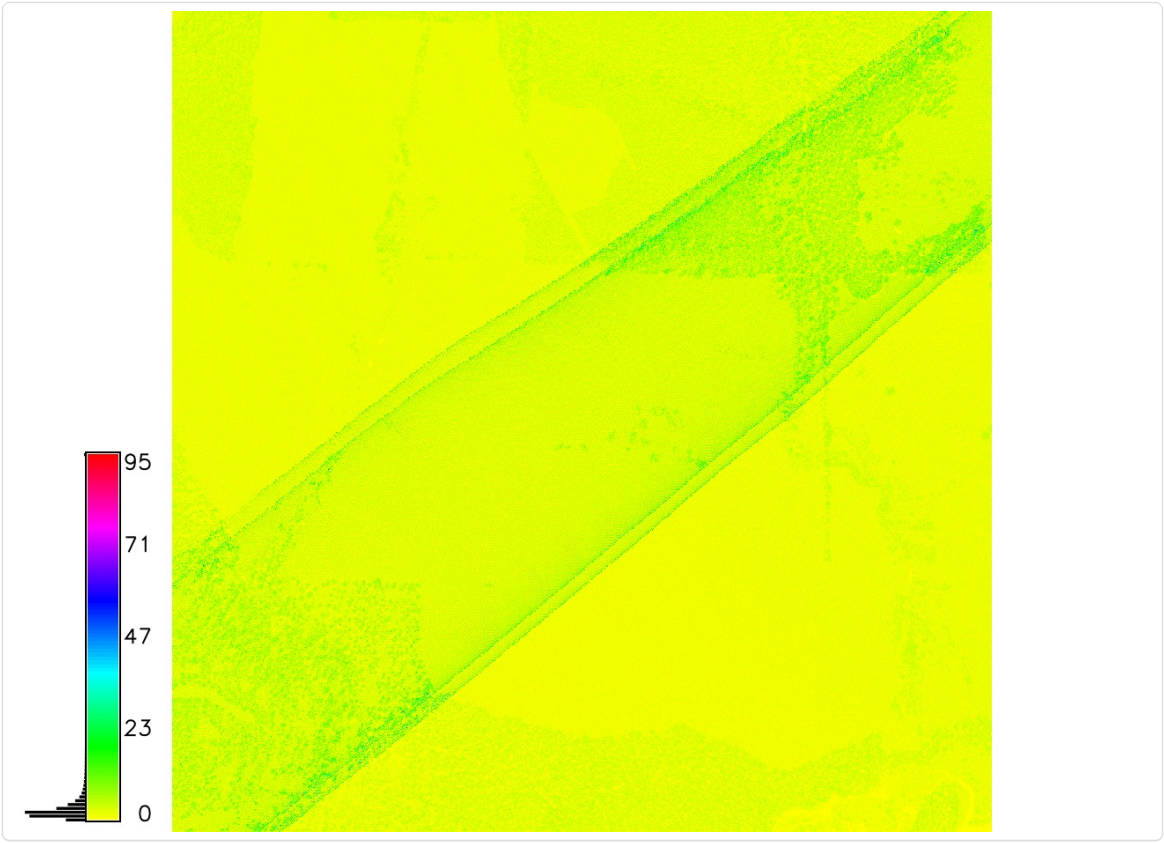
```
tristan@tristan-PC:$ g.region n=220218 s=218694 e=637795 w=636271
res=1 -pa
projection: 99 (Lambert Conformal Conic)
zone: 0
datum: nad83
ellipsoid: grs80
north: 220218
south: 218694
west: 636271
east: 637795
nsres: 1
ewres: 1
rows: 1524
cols: 1524
cells: 2322576
```

The `-a` flag ensures that the region will be aligned to the resolution as opposed to the specific bounds, although in this case it is not necessary, as both are able to be propery aligned.

Point Density

Next, we'd like to calculate the point density of the LIDAR data inside of the region we just defined. This means we want to count how many LIDAR points are inside of each cell in the computational region. Use the `method=n` option of `r.in.lidar` to count the number of points per cell.

```
tristan@tristan-PC:$ r.in.lidar -o input=mid_pines_spm_2013.las
output=mid_pines_dens_all method=n
```

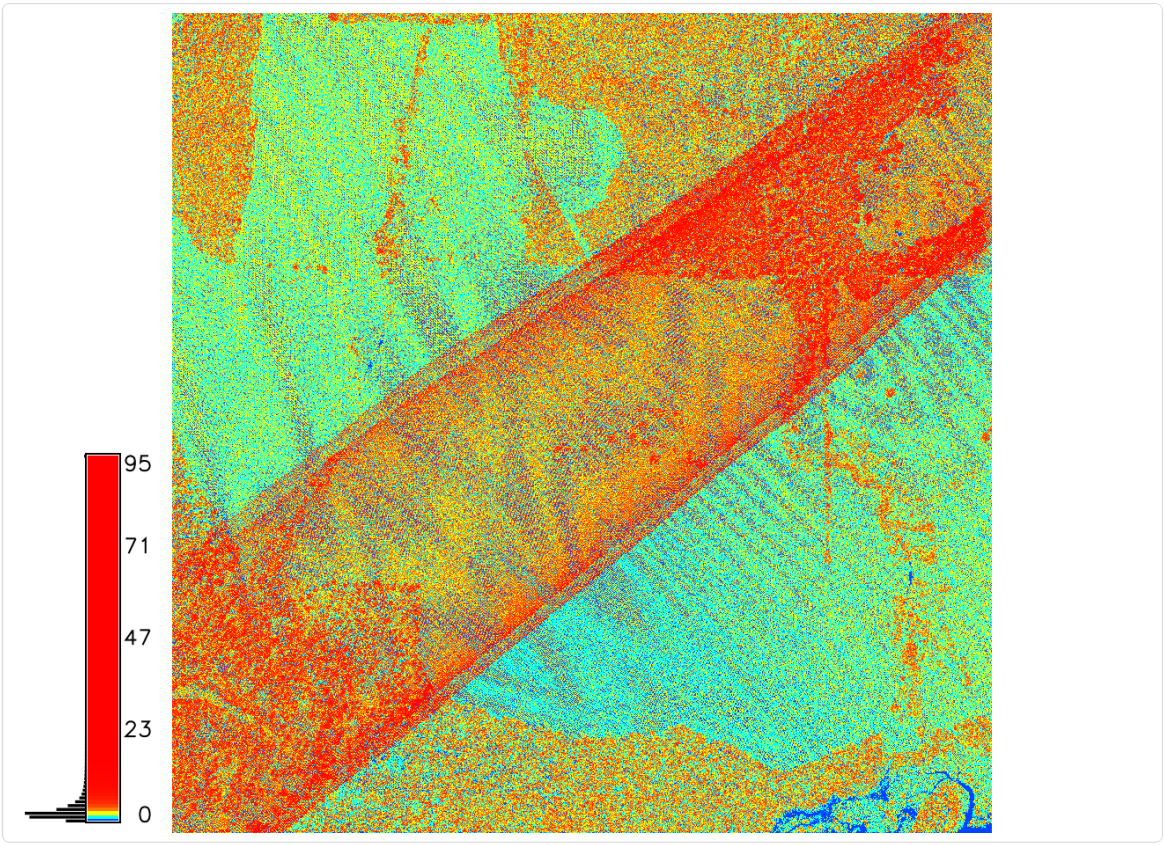


Point Density of All Returns

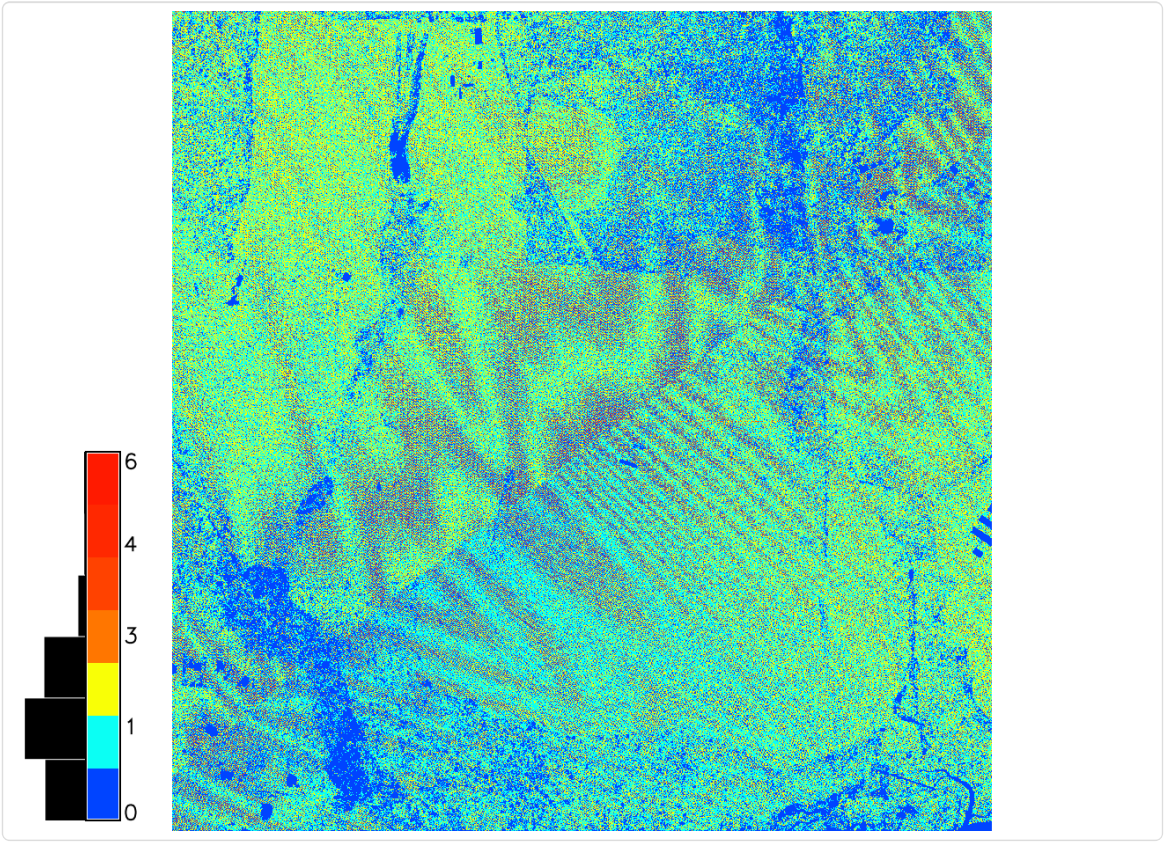
In this image, we can clearly see the overlap of the flight path. There is a higher density of points in the overlapping area because the LIDAR scans that area twice. Additionally, we can see that there is a higher density of points where there is dense vegetation. This is because the LIDAR scanner is capable of receiving multiple returns from a single light pulse. This is due to the fact that the light pulse emitted from the scanner essentially acts the same way a flashlight does. When you shine a flashlight onto a wall, you see a large circle of light. Rather than the laser pulse hitting some object at some infinitely small point, the laser pulse will project some (small) circle of light onto whatever object it hits first. If the footprint of the emitted laser pulse happens to fall partially on a close object and partially on a far-away object, the scanner will detect two points. This is much more likely to happen when scanning dense vegetation as opposed to flat ground, so we get a higher density of points in the areas of dense vegetation.

Now, say we want to compute the density of only *ground* points. We can filter out non-ground points when reading in the data from file by specifying a class filter with the `r.in.lidar` command.

```
tristan@tristan-PC:$ r.in.lidar -o input=mid_pines_spm_2013.las
output=mid_pines_dens_ground class_filter=2 method=n
```

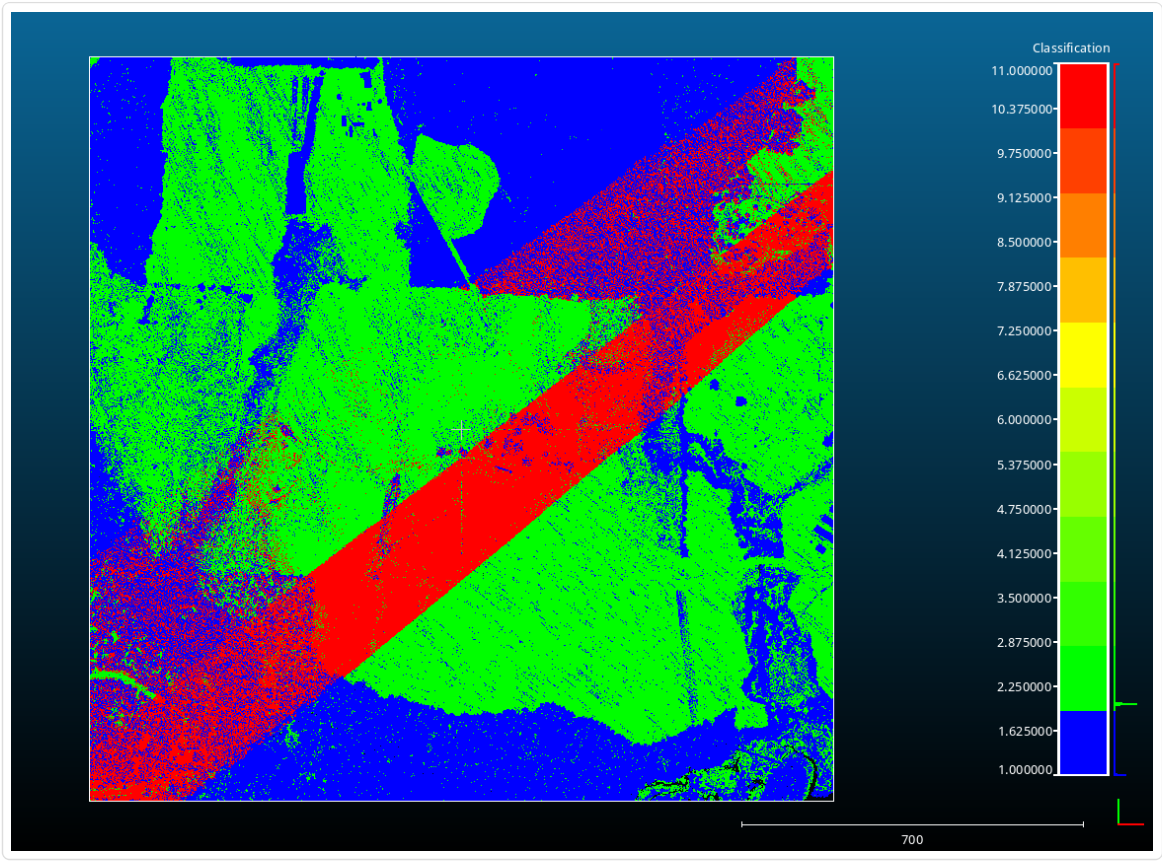


Point Density of All Returns



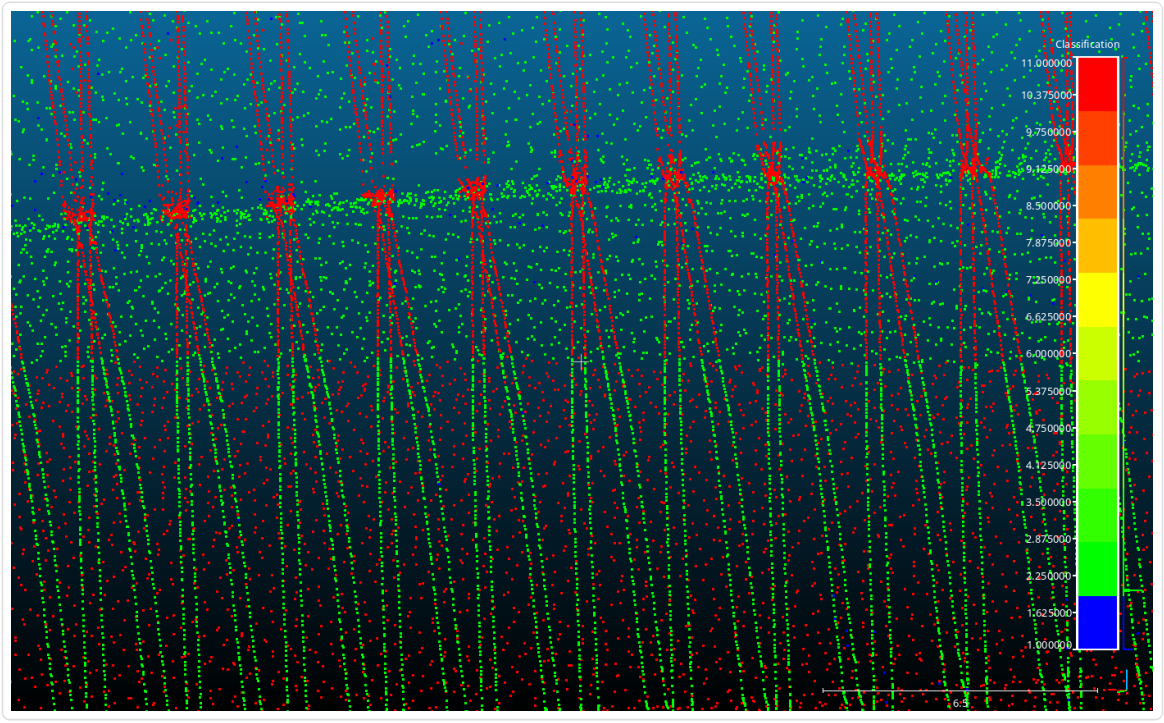
Point Density of Ground Returns

Comparing the density of all returns with the density of ground returns (with histogram equalization applied to both, as seen above) yields some interesting results. First, we see that areas of dense vegetation typically have either 0 or 1 ground return per cell. This is expected, as it is much less likely for a laser pulse to reach the ground when there is vegetation in the way. Next, we see that the large swath of high density points where the flight lines overlapped is not visible in the ground returns. In the open fields, this would seem contradictory because almost every return from those areas should be ground returns. This tells us that whatever algorithm was used to perform the classification probably only allowed a certain number of ground return points per some unit of area. Using CloudCompare to visualize all points in the cloud by classification, we can see that a lot of the returns in the overlapping area are classified as 11 (Reserved for ASPRS Definition), meaning that the classification algorithm probably determined the most likely ground point and left the rest for further classification.



All points colored by classification in CloudCompare

Finally, we see that there is a very distinct line running through the exact center of the flight-line overlap. My best guess is that whatever algorithm was used to perform the classification ran along the center of each flight path and looked out to the center of the overlaps on either side of the flight path. The image above that shows the density of all returns does not have this line, which suggests that the line is only part of the point classification. Additionally, if we use CloudCompare to zoom in to the line and view it from an angle, it appears that the classification method inverts along the line (possibly due to the algorithm following the direction of the flight path) for no apparent reason.



The line that runs through the middle of the flight path overlap

Raster Binning and Classification

In this section, we wish to create a classified raster map where the value of each cell is one of the following classes:

- **1** - ground
- **2** - vegetation
- **3** - buildings

In order to do this, we'll create three different raster maps. The first is a DSM, in which each cell of the raster is equal to the elevation of the highest LIDAR return in that cell, regardless of classification.

```
tristan@tristan-PC:$ r.in.lidar -o input=mid_pines_spm_2013.las
output=mid_pines_all_max method=max resolution=3 class_filter=1,2
```

Note that we're creating a 3m resolution raster. In the previous section, we were looking at the density of a 1m raster, and most of the cells either had 0, 1, or 2 returns per cell. In order to produce a raster with fewer holes, we increase the resolution to 3m in order to guarantee that points will fall within practically every cell.

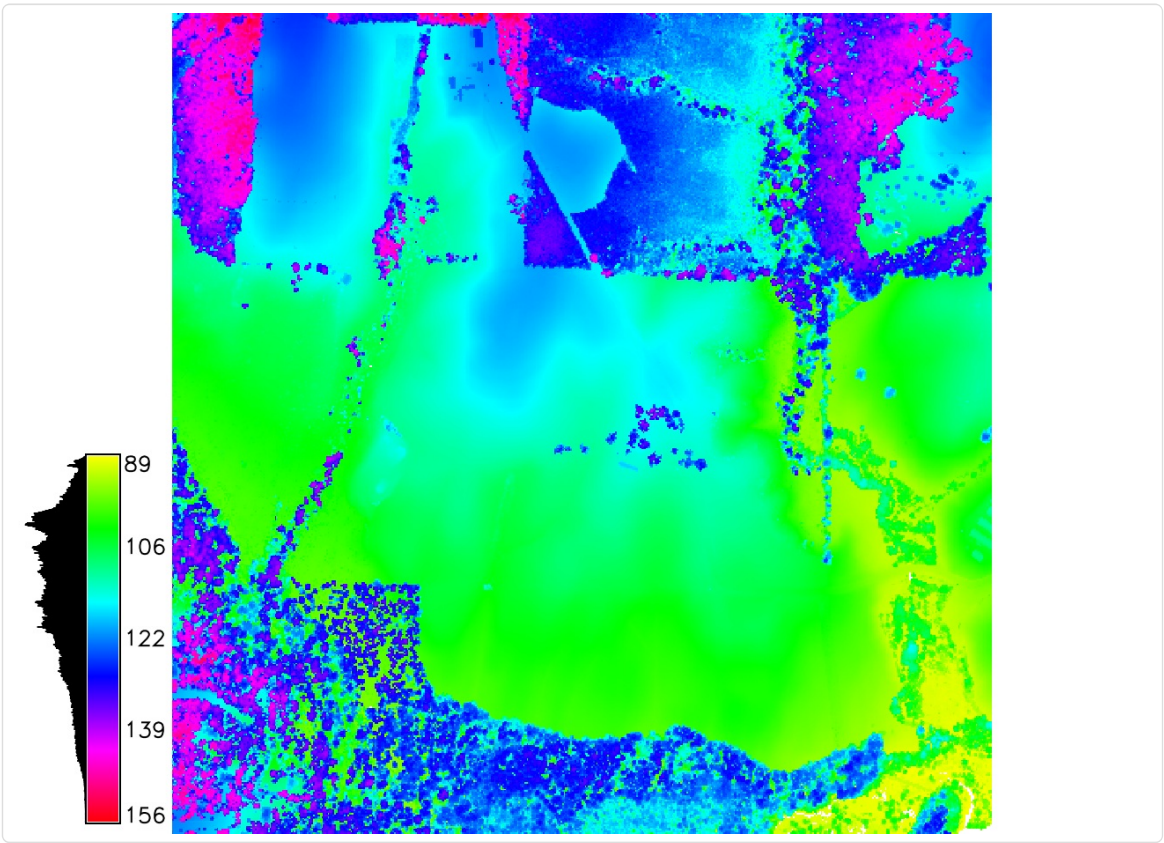
Next, we'll create a surface based on last return, which will represent the vegetative canopy. Note that in this command, we are only considering last returns. It is important to note that a LIDAR pulse is *only considered to have a last return if there is more than one return* . This means that a LIDAR pulse that produces a single return will not classify that return as first and last, it will only be classified as first. We'll take the average of all last returns in each cell in order to represent the average depth that

the LIDAR pulses were able to penetrate the canopy in each cell.

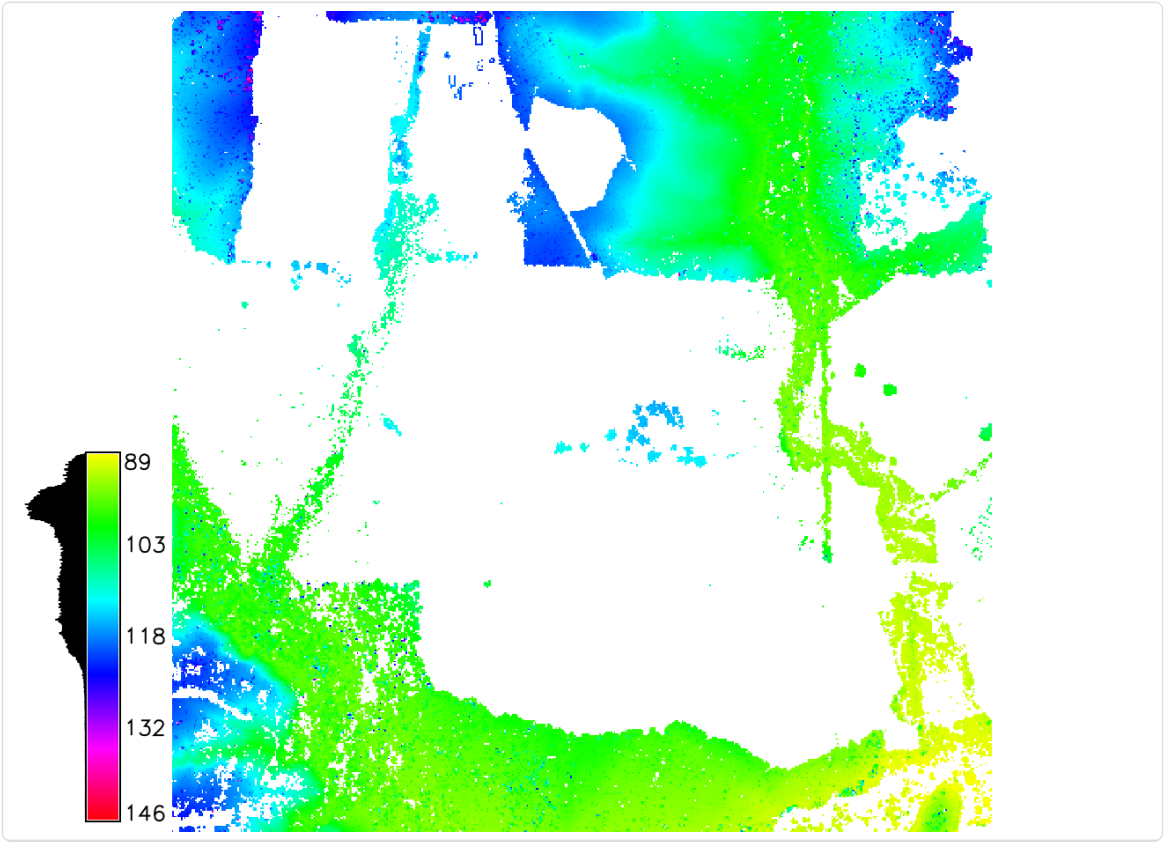
```
tristan@tristan-PC:$ r.in.lidar -o input=mid_pines_spm_2013.las
output=mid_pines_last_mean resolution=3 method=mean
return_filter=last class_filter=1,2
```

Finally, we create a raster which represents the ground surface based on the point classification in the LAS file.

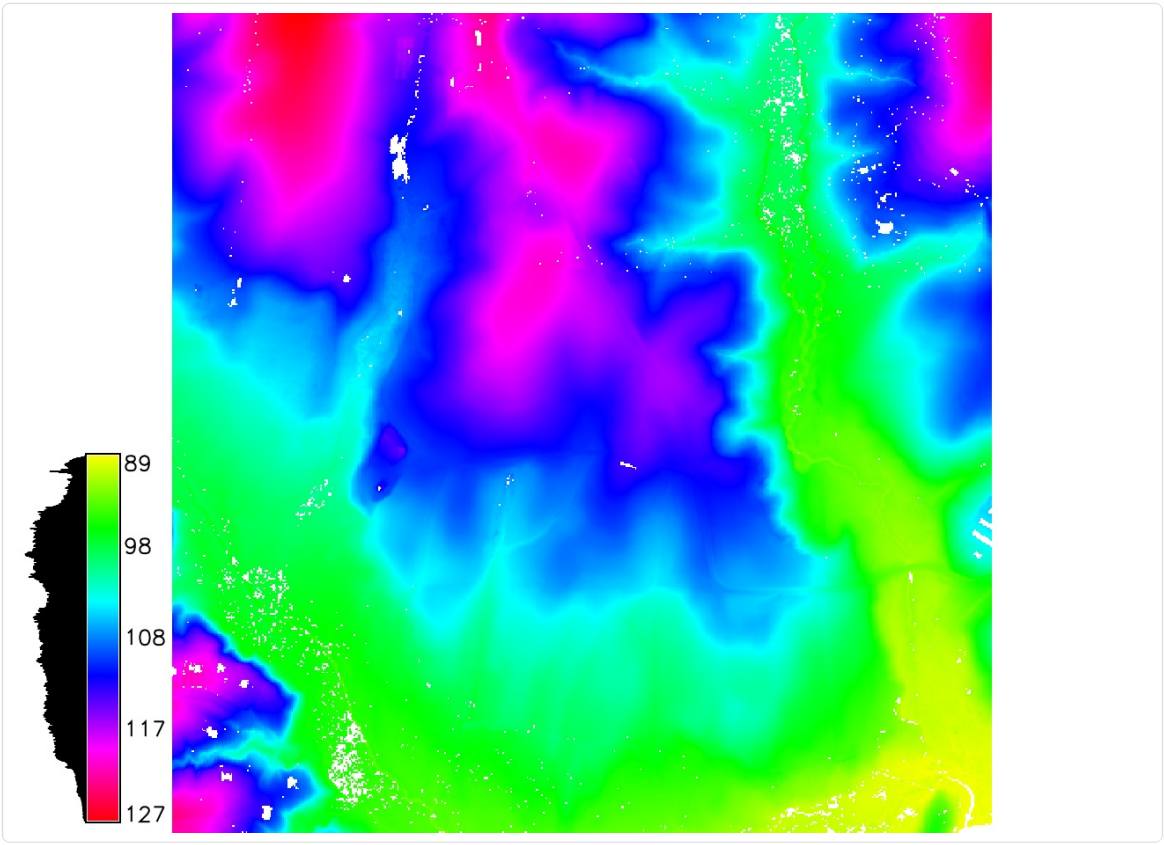
```
tristan@tristan-PC:$ r.in.lidar -o input=mid_pines_spm_2013.las
output=mid_pines_ground_mean resolution=3 class_filter=2
```



DSM



Canopy



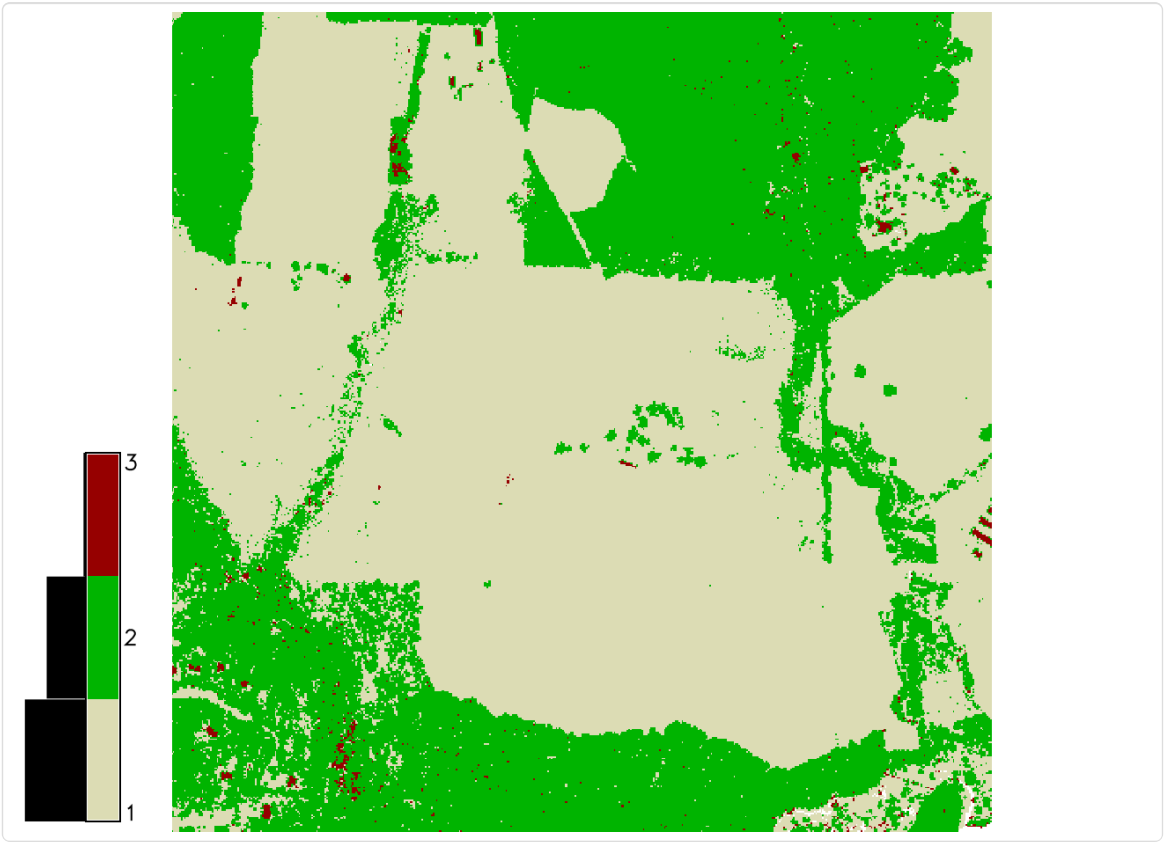
Ground

Now we combine the surfaces using the following logic:

- If the cell is not null in the canopy raster, the cell is vegetation (class 2)
- Otherwise, if the cell is not null in the ground raster, the cell is ground (class 1)
- Otherwise, if the cell is not null in the DSM, the cell is a building (class 3)
- Otherwise, there's nothing in the cell and it is null

which can be run in `r.mapcalc` as follows:

```
tristan@tristan-PC:$ r.mapcalc "classes = if( !
isnull(mid_pines_last_mean), 2, if( !isnull(mid_pines_ground_mean),
1, if( !isnull(mid_pines_all_max),3, null())))"
```



Raster Classification

High Resolution DEM/DSM

In this section, we will import the point cloud as vector data, rather than as raster, and use it to generate a higher resolution DEM and DSM using spline interpolation. First, import the point cloud twice, once with all first-returns and once with only points classified as ground.

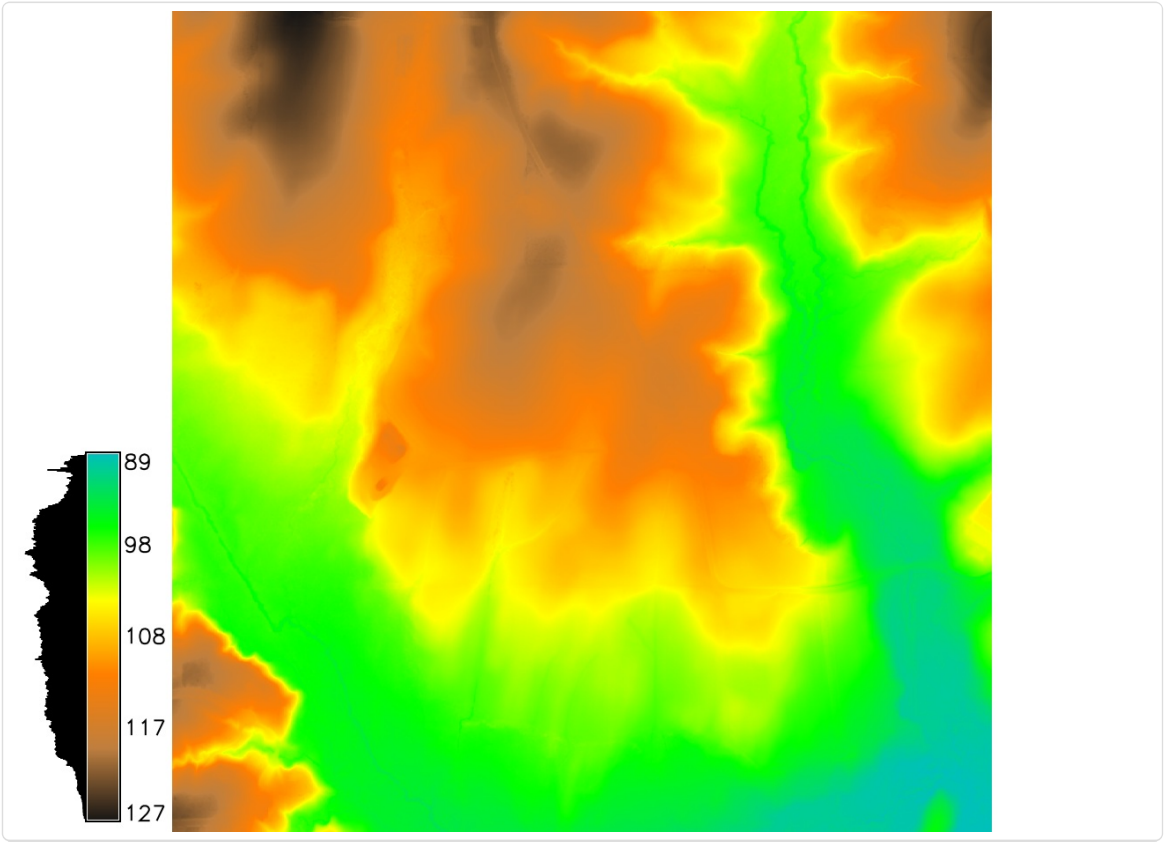
```
tristan@tristan-PC:$ v.in.lidar -t -o input=mid_pines_spm_2013.las
output=mid_pines_ground class_filter=2
tristan@tristan-PC:$ v.in.lidar -t -o input=mid_pines_spm_2013.las
output=mid_pines_surface class_filter=1,2 return_filter=first
```

Next, check the overall point density using `v.outlier` (doc).

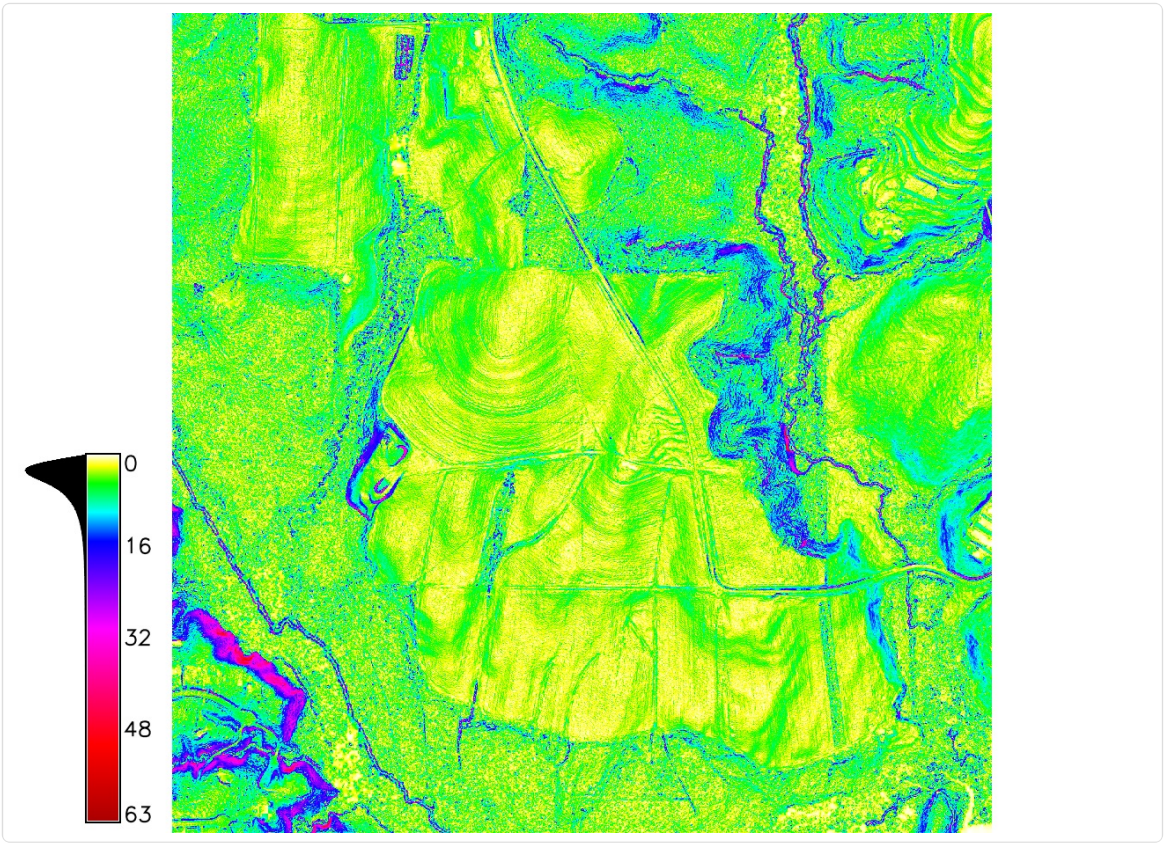
```
tristan@tristan-PC:$ v.outlier -e input=mid_pines_ground output=dummy
outlier=dummy
Estimated point density: 1.111
Estimated mean distance between points: 0.9487
```

As you can see, the point cloud has a density of approximately 1.1 points per square meter and the average distance between points is almost a meter. Therefore, we want to choose a resolution that will result in holes, and thus allow interpolation to occur. We'll choose a resolution of 0.3 meters. Set the resolution using `g.region` and generate the rasters using `v.surf.rst` (doc).

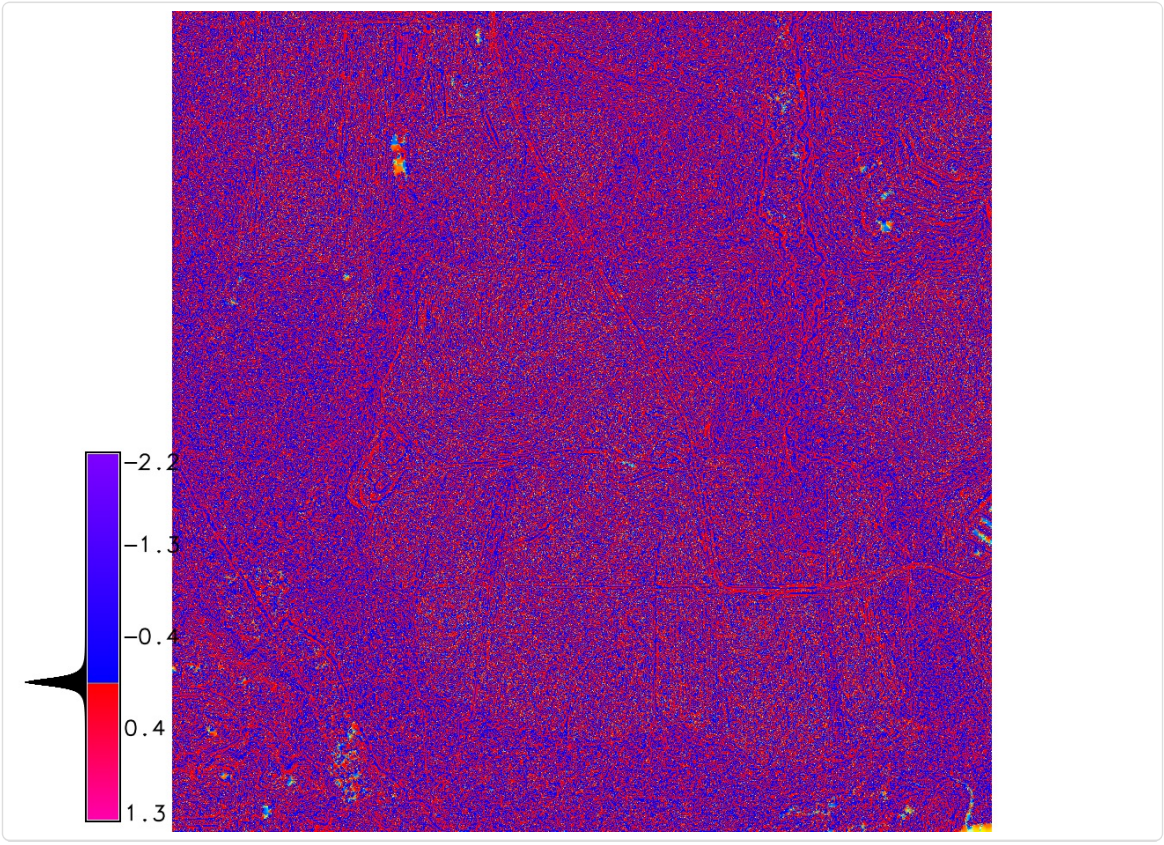
```
tristan@tristan-PC:$ g.region res=0.3
tristan@tristan-PC:$ v.surf.rst input=mid_pines_ground
elevation=mid_pines_ground_elev slope=mid_pines_ground_slope
pcurv=mid_pines_ground_profcurv npmin=80 tension=20 smooth=1
tristan@tristan-PC:$ v.surf.rst input=mid_pines_surface
elevation=mid_pines_surface_elev slope=mid_pines_surface_slope
pcurv=mid_pines_surface_profcurv npmin=80 tension=20 smooth=1
```



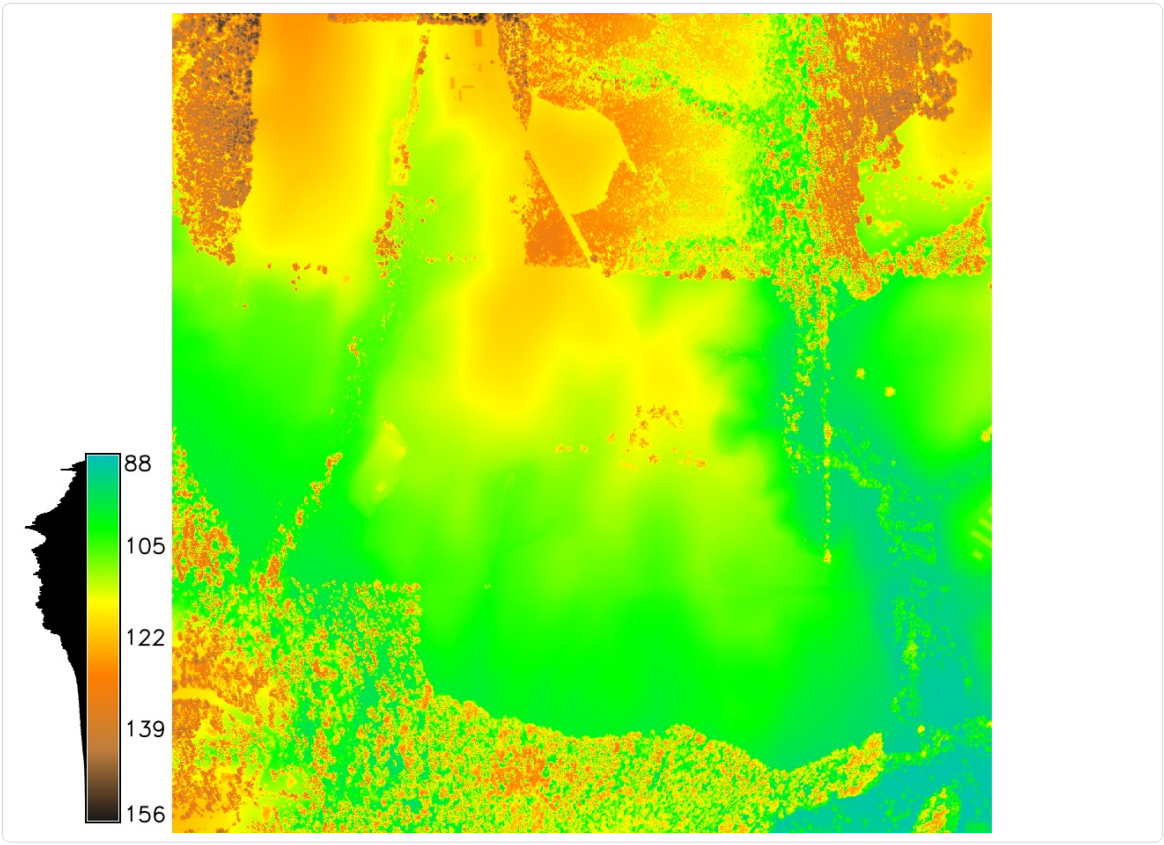
Ground Elevation



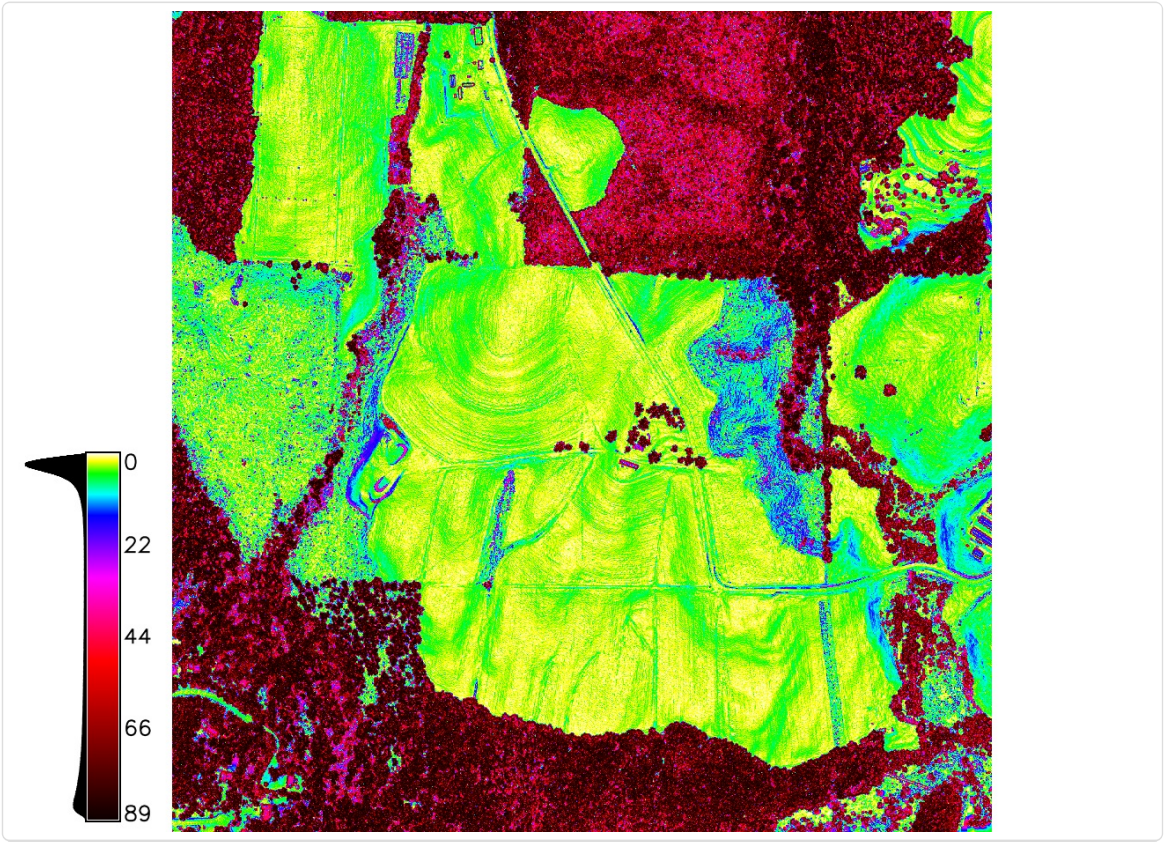
Ground Slope



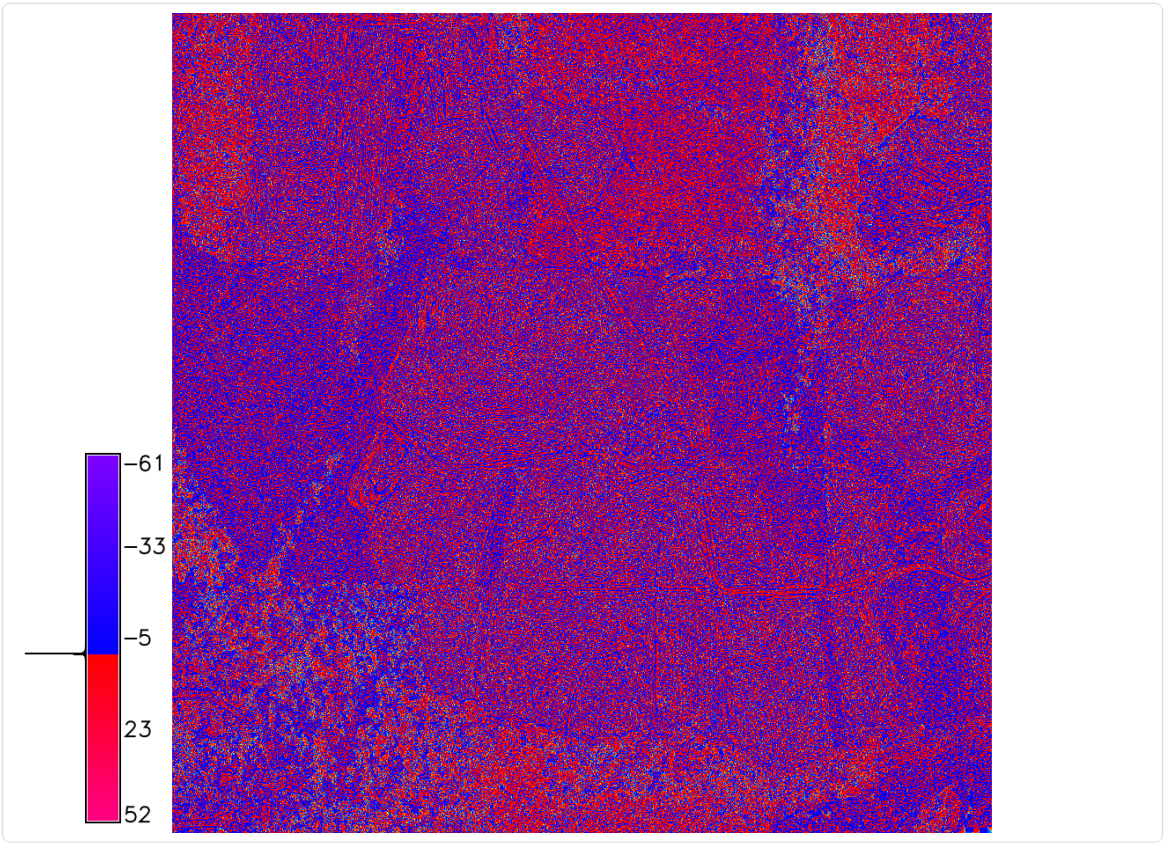
Ground Profile Curvature



Surface Elevation

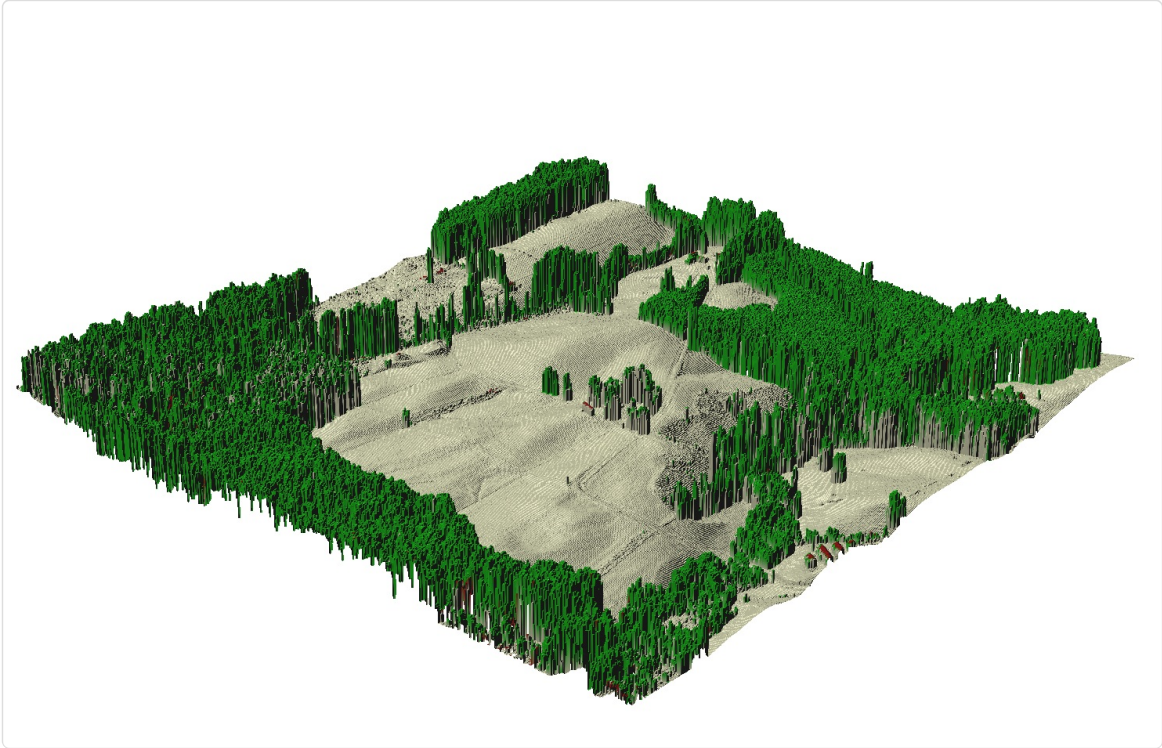


Surface Slope

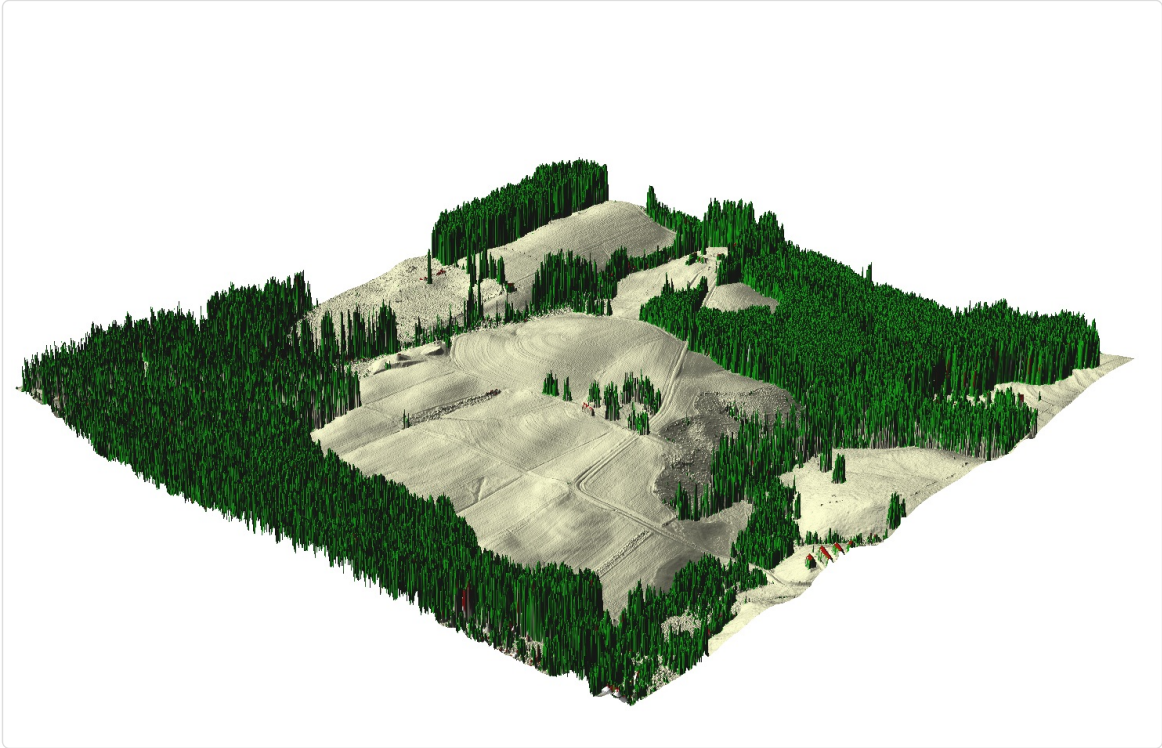


Surface Profile Curvature

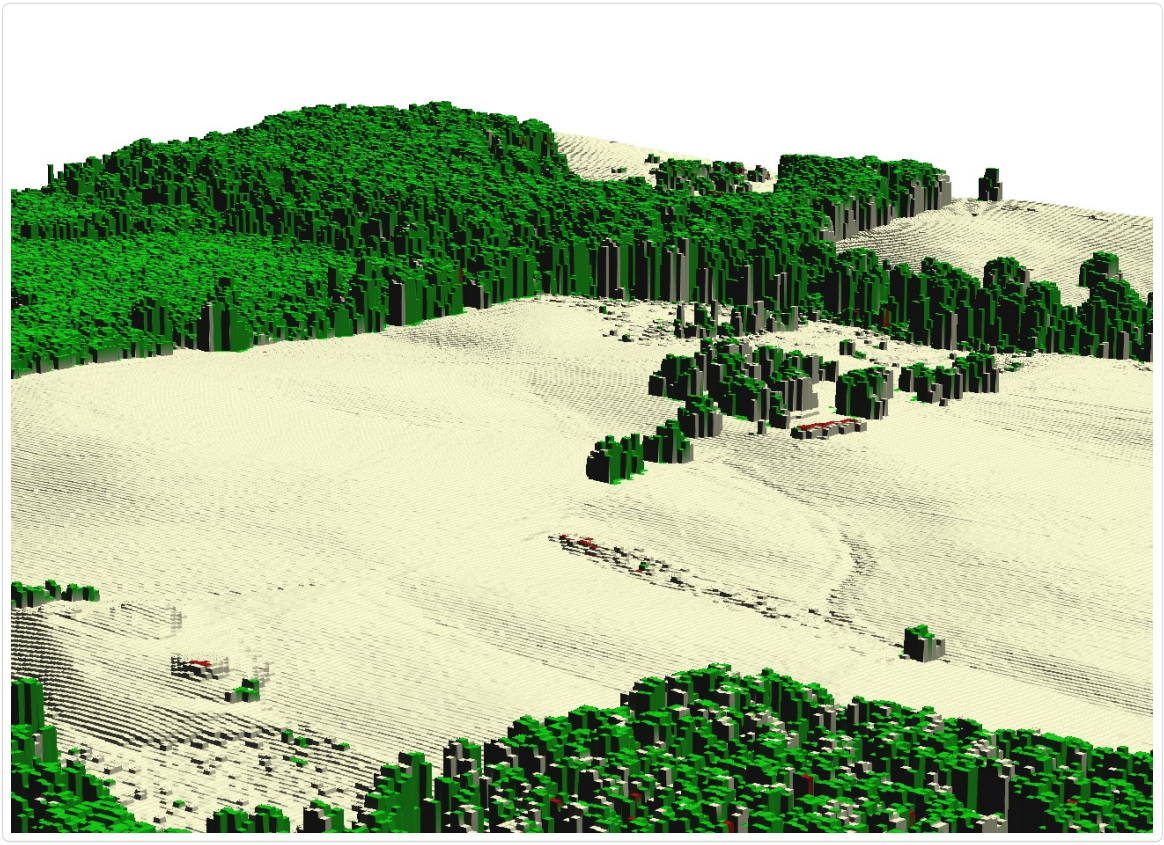
As you can see, we have now generated a higher resolution DEM and DSM compared to using `r.in.lidar`. If we view them in 3D, we can easily see the extra detail. The following four images show a 3D view of both the 1m and 0.3m resolution DSMs (it's useful to right-click and open each in a new tab to be able to toggle quickly between them) colored by the classifications we created in the previous section.



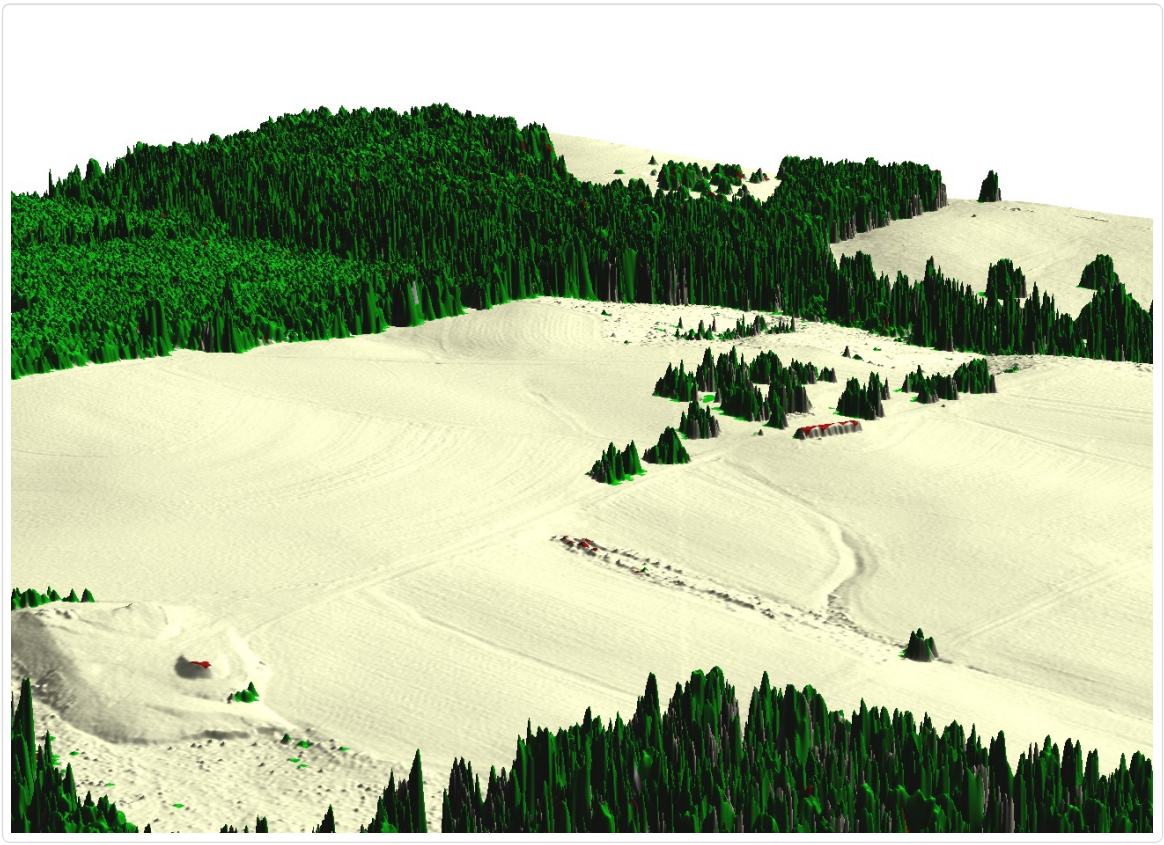
1m resolution DSM from `r.in.lidar`



0.3m resolution DSM from `v.in.lidar`



1m resolution DSM from `r.in.lidar` (zoomed)



0.3m resolution DSM from `v.in.lidar` (zoomed)

This 3D animation allows us to see the where the surface canopy separates from the ground in the 0.3m resolution rasters, using a cutting plane to view inside of and underneath the canopy surface:

Visualization of ground DEM and DSM using cross-sections

DEM and GCP Differences

Next, we'll compute the difference in elevation between the GCPs and the high resolution ground surface we've just generated from the LIDAR data. The GCP locations are in the PERMANENT mapset as `GCP_12_degrees`. We're going to copy that data to a new vector dataset, add two columns (one for the DEM elevation, and one for the difference between the DEM elevation and the GCP elevation), put the elevation values from the `mid_pines_ground_elev` raster at the GCP locations into the table, and finally, compute and display the differences.

```
tristan@tristan-PC:$ g.copy vect=GCP_12_degrees,GCP_12_differences
tristan@tristan-PC:$ v.db.addcolumn map=GCP_12_differences
columns="dem_height DOUBLE, height_difference DOUBLE"
tristan@tristan-PC:$ v.what.rast -i map=GCP_12_differences
raster=mid_pines_ground_elev column=dem_height
tristan@tristan-PC:$ v.db.update map=GCP_12_differences
column=height_difference qcolumn="ASL - dem_height"
tristan@tristan-PC:$ v.db.select map=GCP_12_differences
columns=ASL,dem_height,height_difference
```

These commands will print the followed data (which I have cleaned up for presentation):

GCP Elevation (m)	DEM Elevation (m)	Difference (m)
105.06	105.0564	0.0036
104.533	104.5714	-0.0384
111.888	111.8264	0.0616
115.335	115.4862	-0.1512
111.315	111.3821	-0.0671
112.526	112.5003	0.0257
106.947	106.9645	-0.0175
111.069	111.1376	-0.0686
109.971	110.0439	-0.0729
107.193	107.3439	-0.1509
109.034	109.0509	-0.0169
116.12	116.2348	-0.1148

Next, compute the univariate statistics.

```
tristan@tristan-PC:$ v.univar -e GCP_12 column=height_difference
number of features with non NULL attribute: 12
number of missing attributes: 0
number of NULL attributes: 0
minimum: -0.1512
maximum: 0.0616
range: 0.2128
sum: -0.6074
mean: -0.0506167
mean of absolute values: 0.0657667
population standard deviation: 0.0639817
population variance: 0.00409366
population coefficient of variation: 1.26404
sample standard deviation: 0.0668267
sample variance: 0.00446581
kurtosis: -1.22607
skewness: -0.0610031
1st quartile: -0.1148
median (even number of cells): -0.05275
3rd quartile: -0.0169
90th percentile: 0.0257
```

From this statistical data we can see that, on average, the LIDAR derived DEM was within about 5cm

of the elevation of the GCPs. This is very accurate, which is not unusual for LIDAR data, but I would be interested in seeing how the LIDAR data was georectified. If it was done so using these GCPs, then we would expect the accuracy to be at least this high.