Overview

In this assignment, we'll analyze the differences between the LIDAR and UAS derived digital elevation models and digital surface models.

LIDAR and UAS DEM Differences

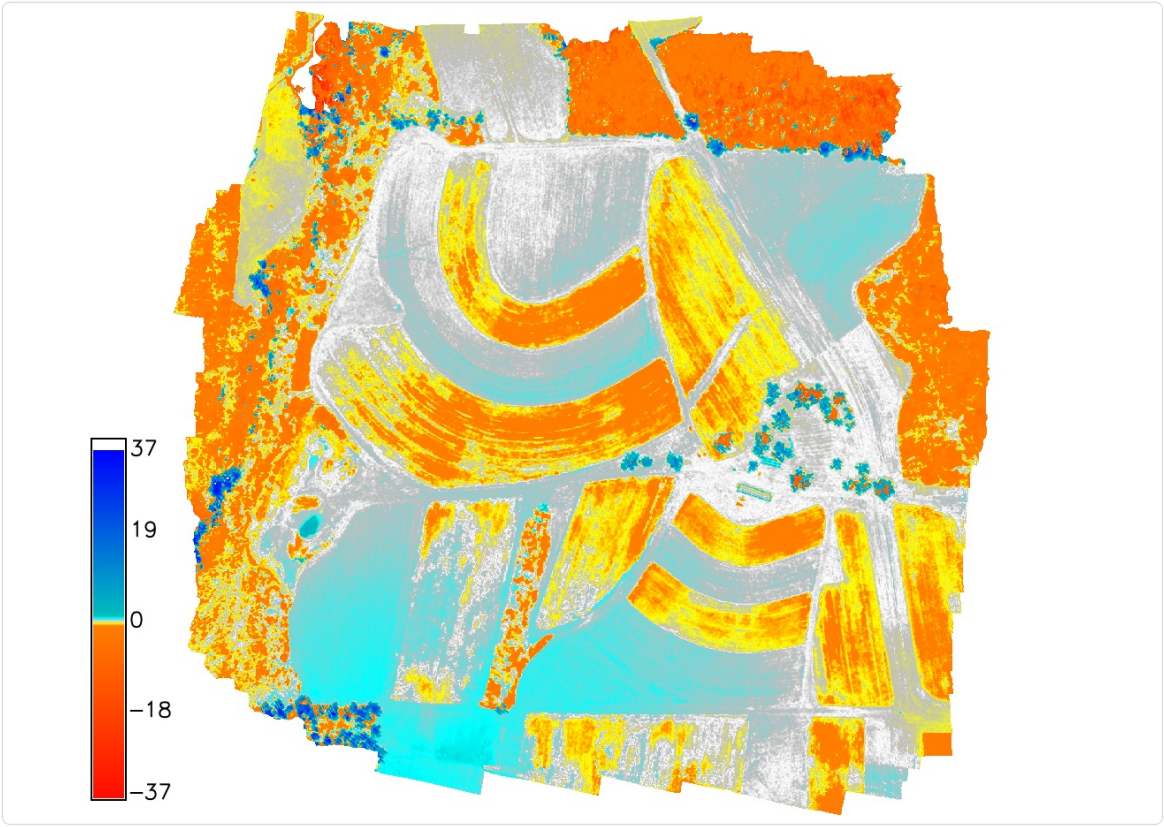
In this section, we'll evaluate the various DSMs in order to determine which are suitable for fusion. We'll compare differences at GCP locations, as well as identify potential systematic error and identify distortions.

From the previous assignment, we know that the `mid_pines_ground_elev` raster has an average error of around 5cm when compared to the GCPs. Therefore, we'll use it as a baseline to compare the UAS DSMs to in order to determine which UAS DSM we'd like to use in our fusion. Compute these differences using `r.mapcalc`:

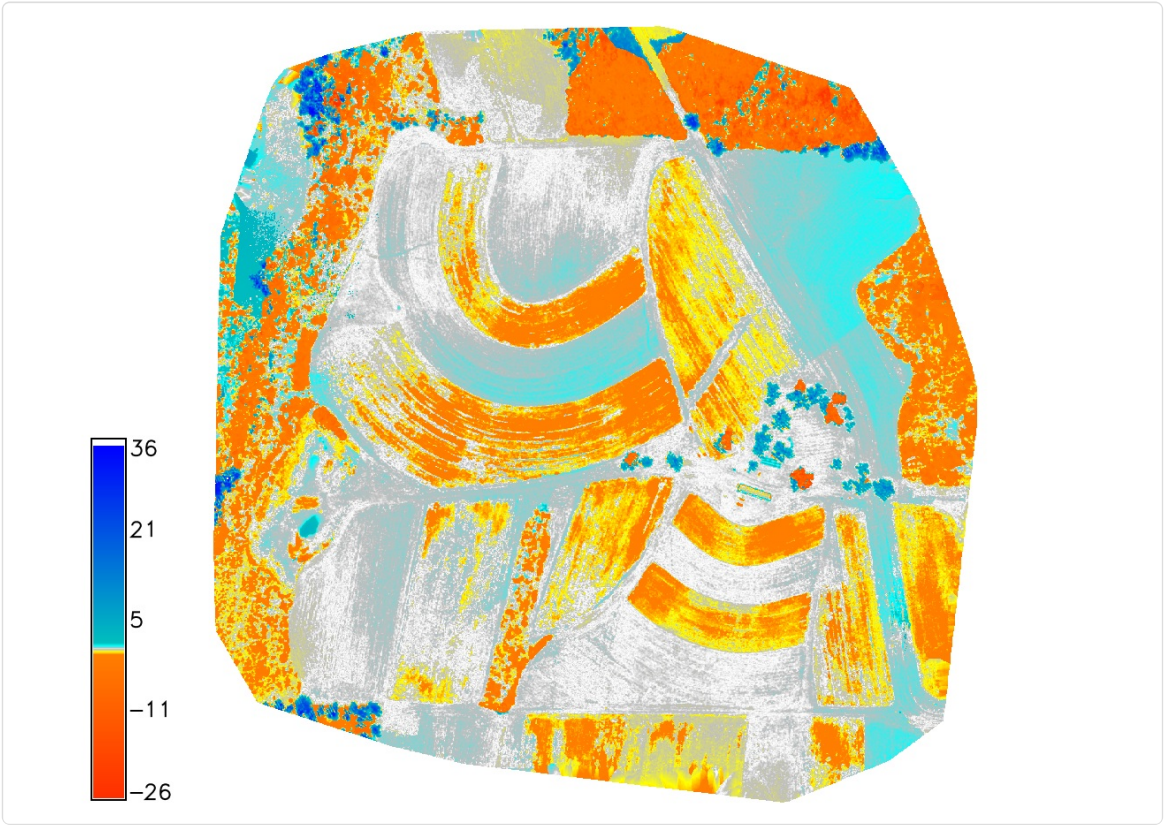
```
g.region rast=mid_pines_ground_elev -p
r.mapcalc "diff_lidardsm_agi_june = mid_pines_ground_elev -
2015_06_20_DSM_agi_11GCP"
r.mapcalc "diff_lidardsm_pix4d_june = mid_pines_ground_elev -
2015_06_20_pix4d_11GCP_dsm"
```

Using the following color table, we can see the differences between the UAS and LIDAR derived DSMs:

```
-40 red
-1 orange
-0.5 yellow
-0.1 grey
0 white
0.1 grey
0.5 cyan
1 aqua
35 blue
```



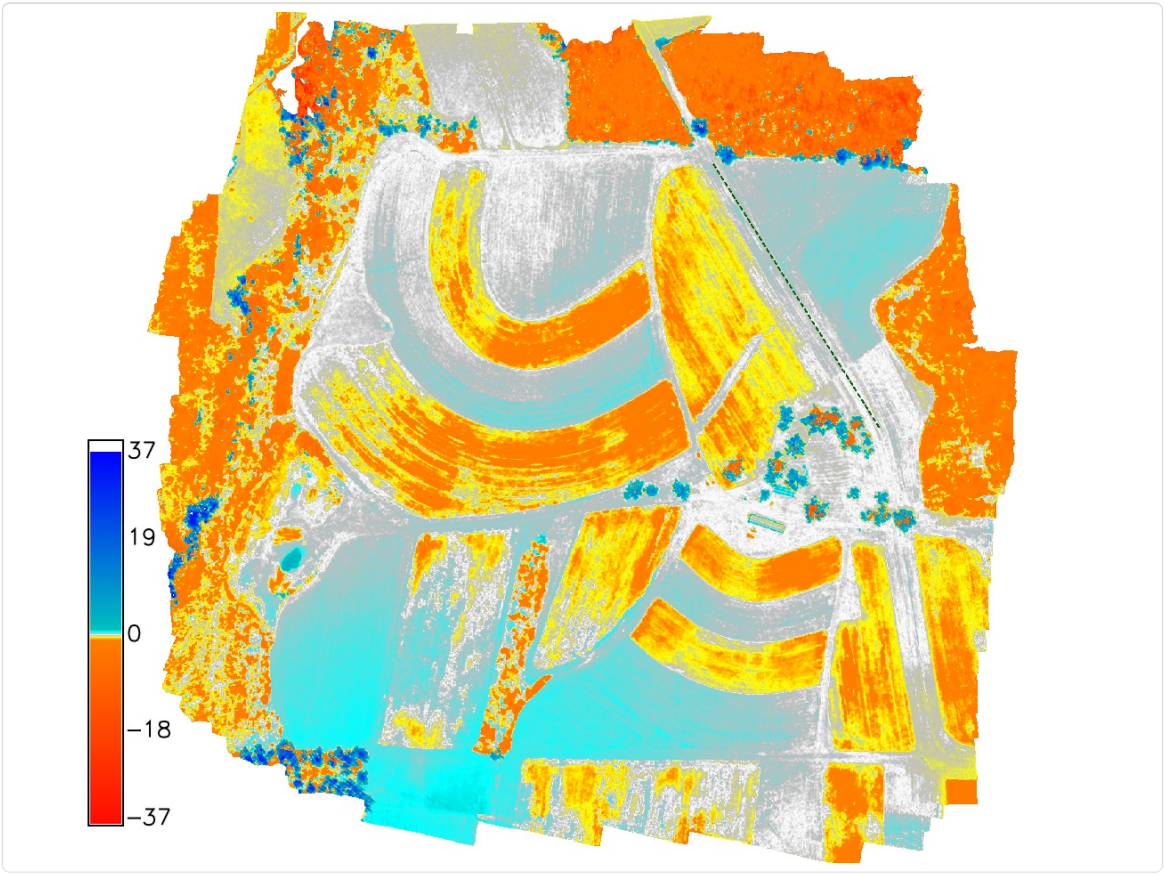
Difference between LIDAR and Agisoft DSMs



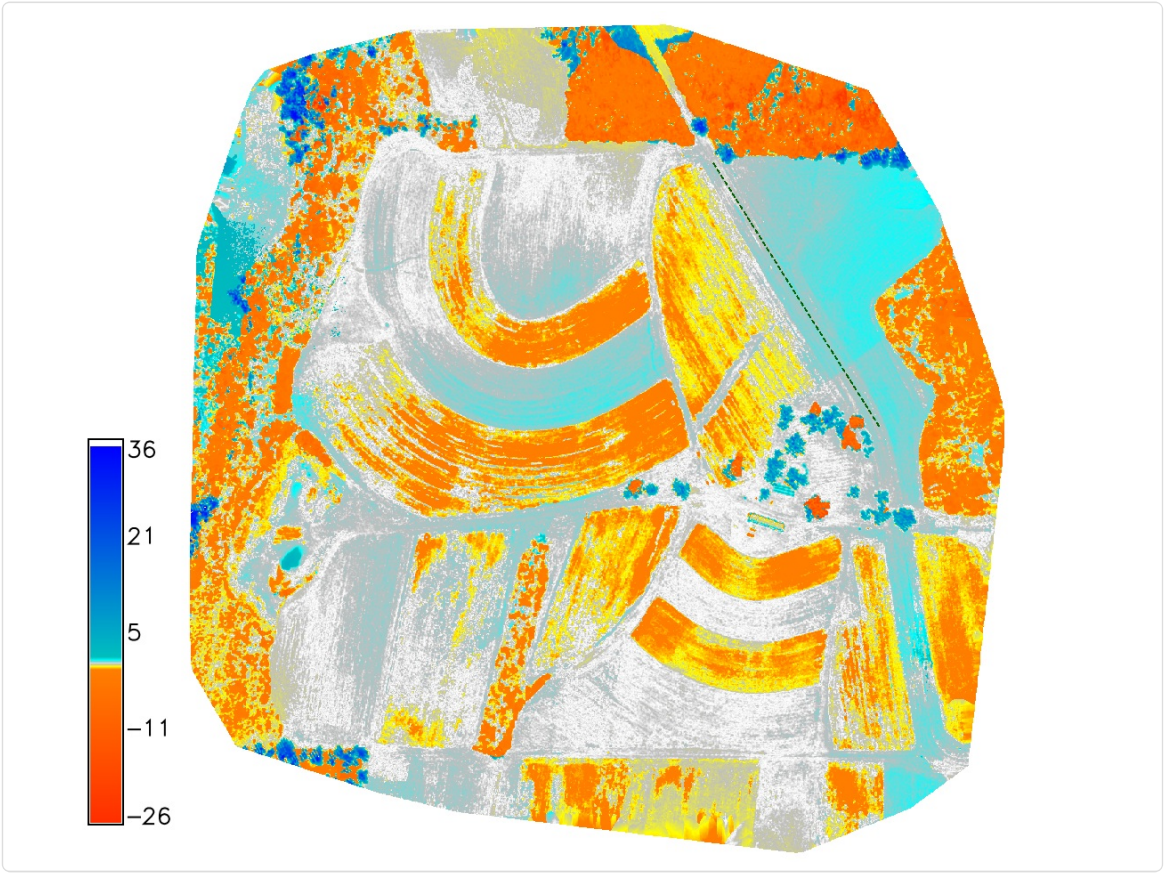
Difference between LIDAR and Pix4D DSMs

Comparing the two difference rasters, we can see that both the Agisoft and Pix4D DSMs are fairly

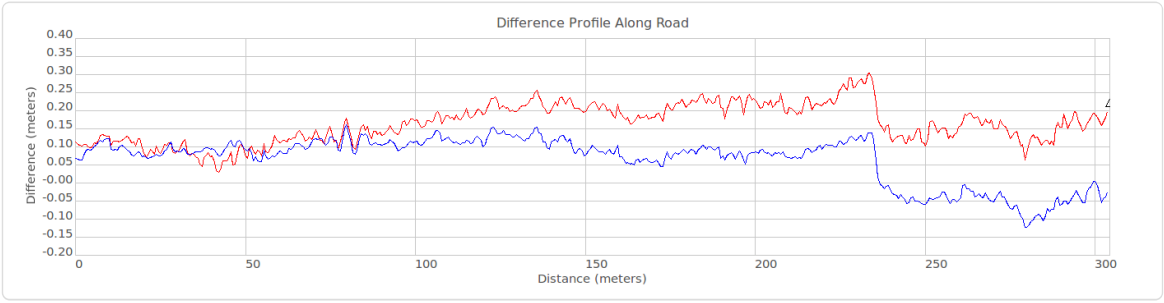
accurate (as compared to the LIDAR DSM) on bare earth surfaces. The Agisoft DSM appears to be systematically lower than the LIDAR DSM in a large portion of the southern fields, while the Pix4D DSM appears to by systematically lower than the LIDAR DSM in a large portion of the northeast corner of the raster. We can use the profile tool to further show this systematic error (i.e. averaging the differences in a given area doesn't reduce the overall difference).



Profile Line on the Agisoft Difference Raster

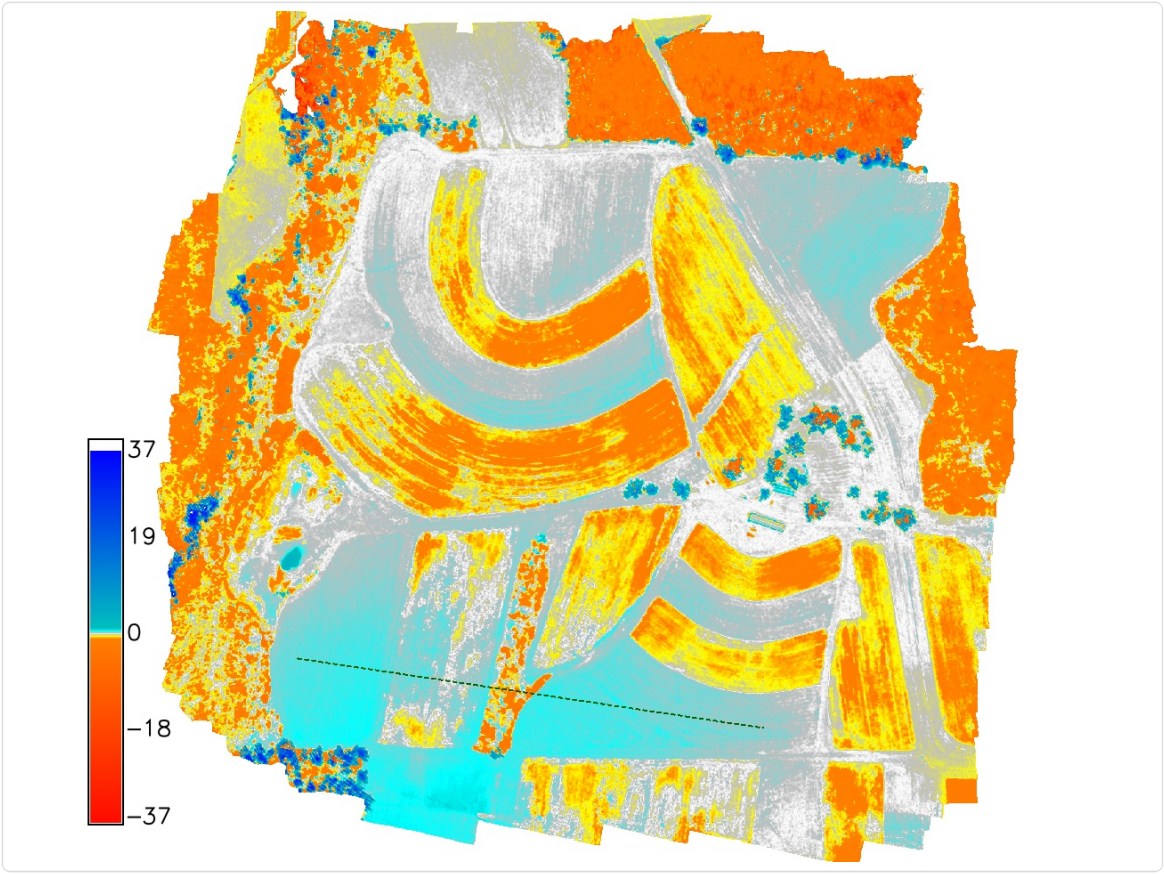


Profile Line on the Pix4D Difference Raster

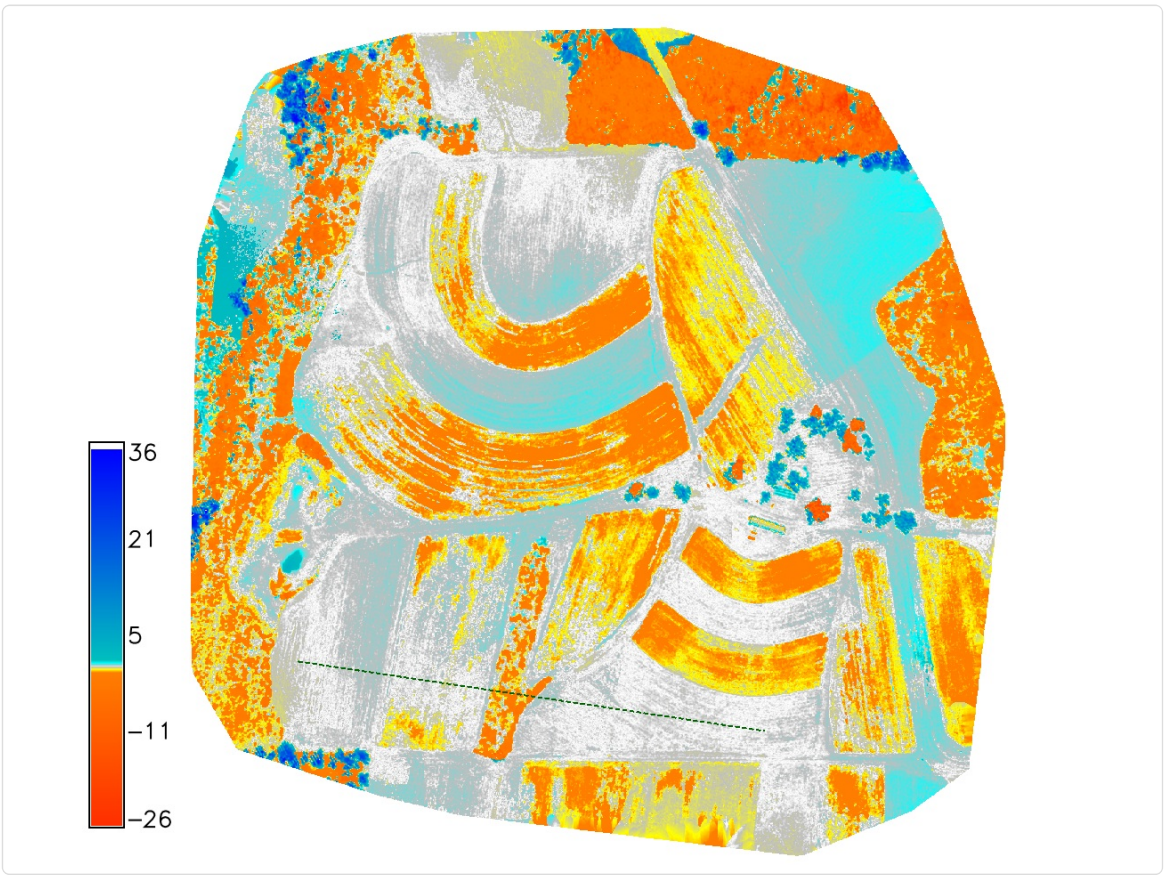


Plot of Difference Values Along Profile Line

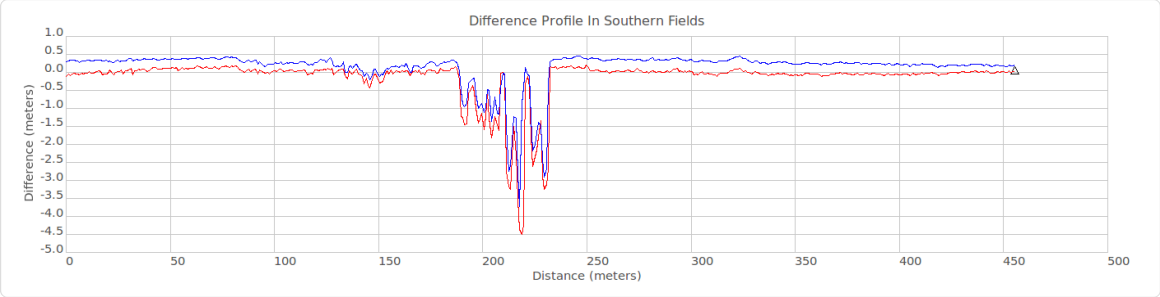
This plot clearly shows a divergence in differences between the two software packages along the road. The Agisoft differences (in blue) tend to hang out around zero difference, not really showing a systematic difference. Pix4D (in red) on the other hand, has a gradual increase in difference along the profile line, indicating that the Pix4D raster is typically lower in elevation than the LIDAR raster. Additionally, we see a sharp drop of about 15cm in both difference plots at about 230m in the profile plot. Because it occurs in both plots, we can infer that the drop (assuming it does not actually exist in the real world) is an artifact found in the LIDAR DSM. Going back to Assignment 5, recall that there was a distinct line going through the center of the flight-line overlap that we determined to be an artifact of the classification algorithm.



Profile Line on the Agisoft Difference Raster



Profile Line on the Pix4D Difference Raster



Plot of Difference Values Along Profile Line

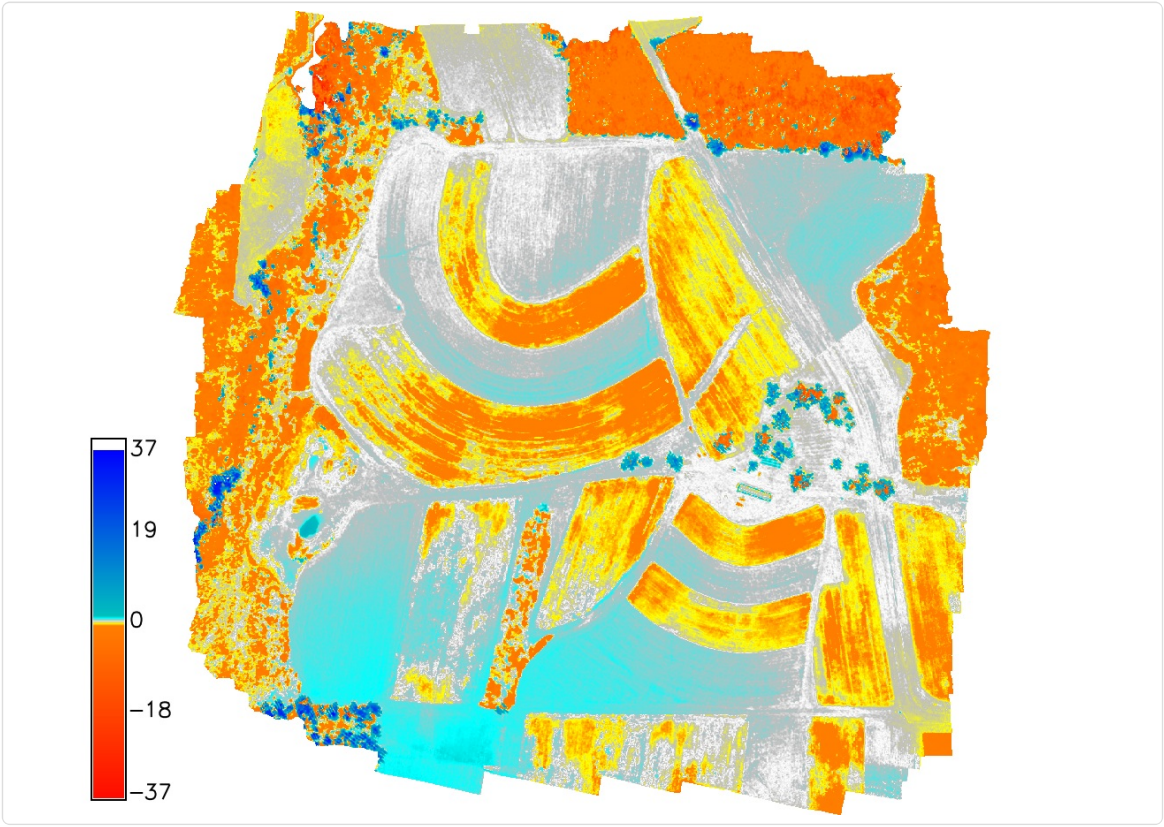
This plot clearly shows that the Pix4D (in red) DSM has very little difference from the LIDAR DSM along the profile line in the southern fields. The Agisoft (in blue) DSM, however, is systematically lower than the LIDAR DSM (because difference values are positive). Additionally, we can see a large jump in negative differences in both rasters where the profile line crosses over an area of dense vegetation. Because the imagery used to generate the UAS DSMs was taken (years) later than the LIDAR data was collected, there are significant differences in vegetation height. This plot suggests that there was much taller vegetation in that are when the UAS was flown as compared to when the LIDAR data was collected.

Correcting LIDAR Shift

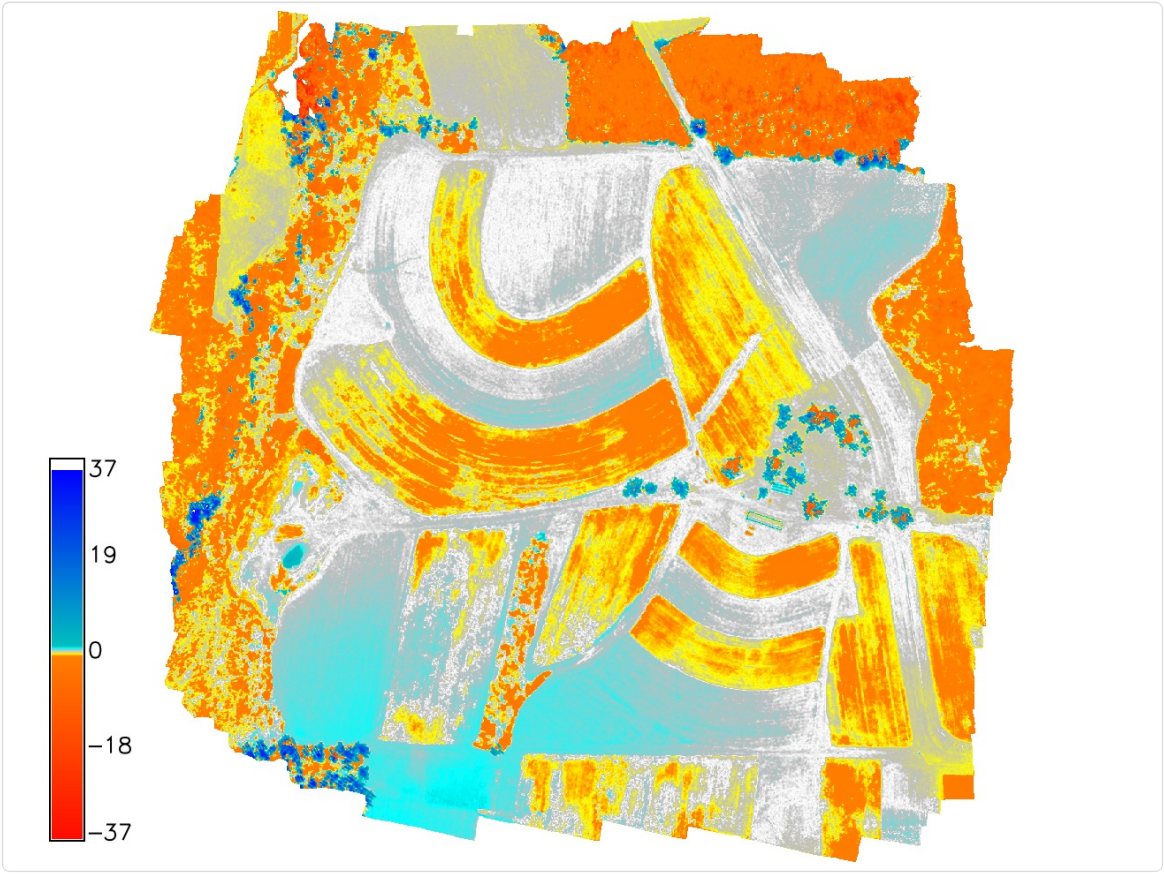
As we have seen in the previous section, there are areas of systematic error between the LIDAR derived and UAS derived DSMs. We have seen that this difference is very small, but if we wanted to correct for this data, we could determine an overall systematic shift and subtract it from either the LIDAR or UAS DSMs. Here, I'm going to correct the LIDAR DSM since it is used to calculate differences in both of the UAS DSMs. Recall from Assignment 5 that we had a very small (on the order of a few centimeters) difference in elevation at the GCP locations. Averaging the differences at each GCP location gives an average difference of 0.051m (5.1cm). This is a good starting point when considering a shift value. We apply this shift and calculate the new differences using `r.mapcalc`.

```
g.region rast=mid_pines_surface_elev -p
r.mapcalc "corrected_lidar = mid_pines_surface_elev - 0.051"
r.mapcalc "corrected_agisoft_diff = corrected_lidar -
2015_06_20_DSM_agi_11GCP"
r.mapcalc "corrected_pix_diff = corrected_lidar -
2015_06_20_pix4d_11GCP_dsm"
```

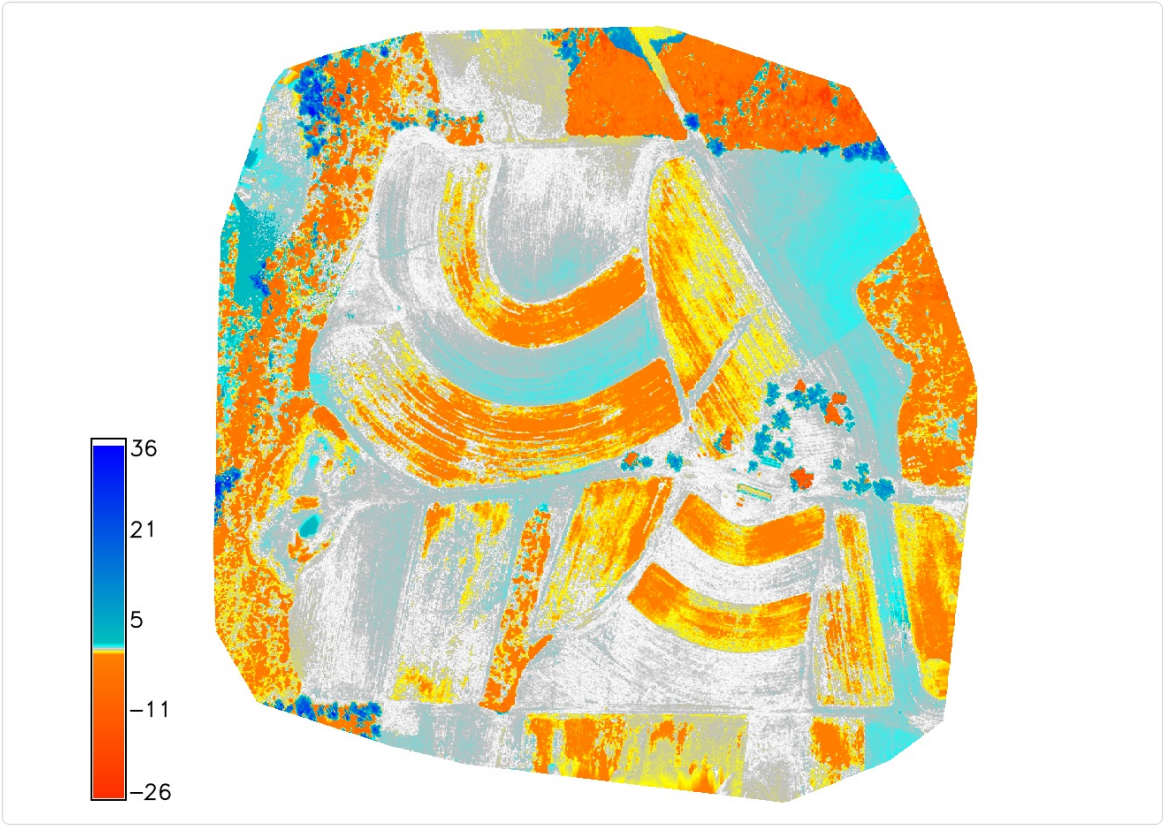
The following four images show the differences before and after correction of the LIDAR DSM:



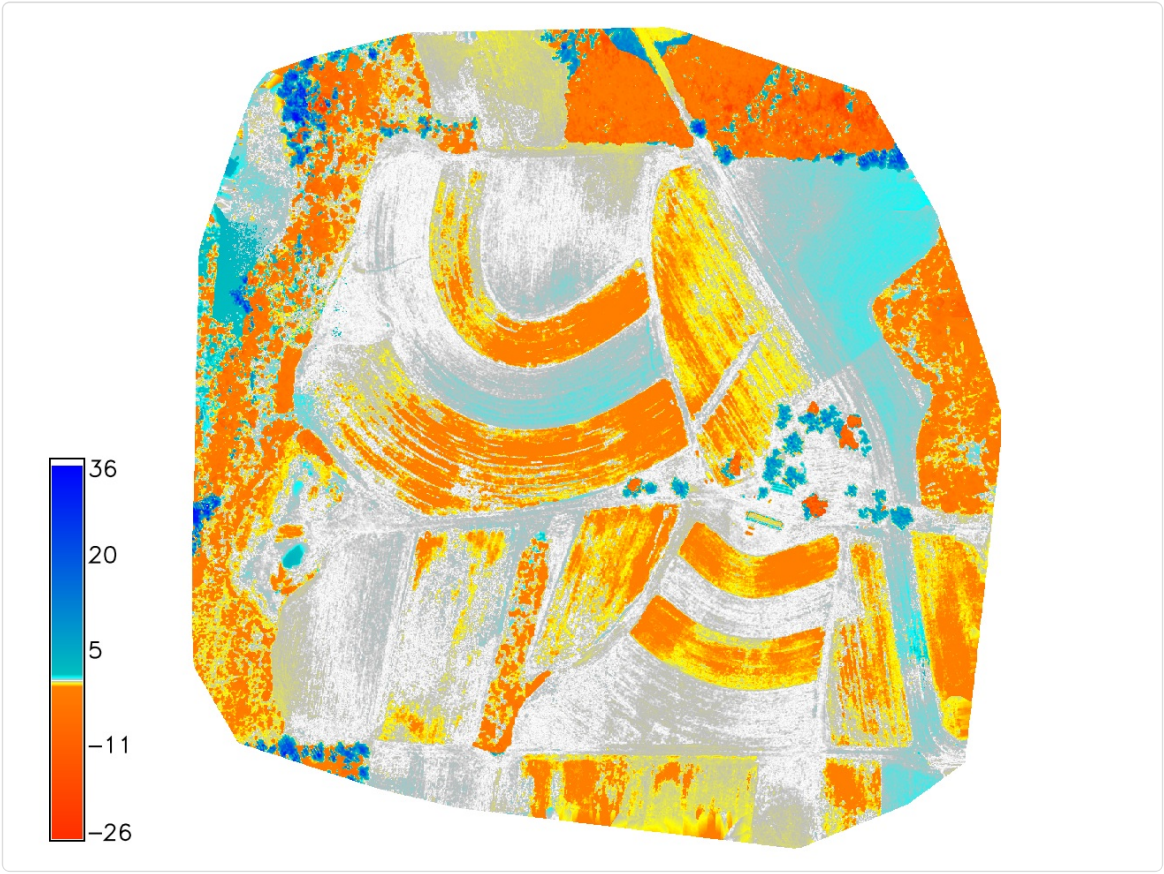
Original Agisoft/LIDAR Differences



Corrected Agisoft/LIDAR Differences



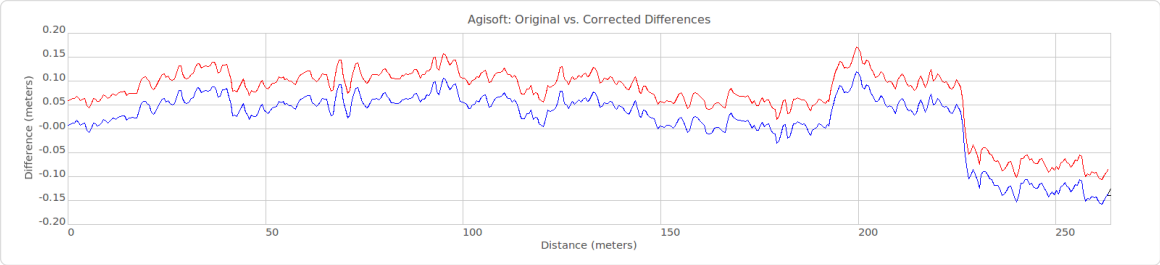
Original Pix4D/LIDAR Differences



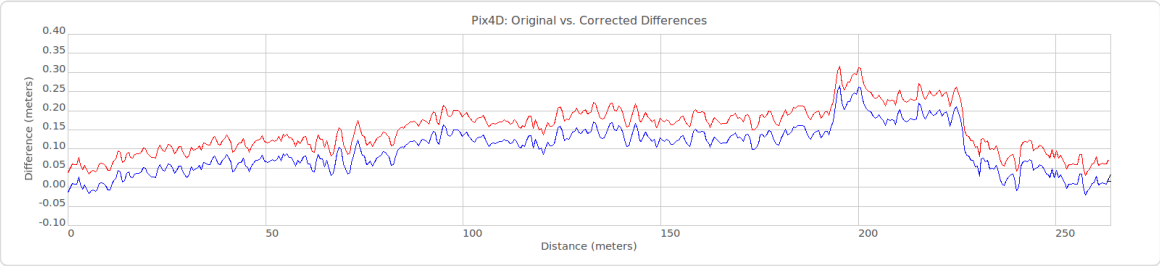
Corrected Pix4D/LIDAR Differences

As we can see, in both cases the differences appear to have shrunk. This can be confirmed by once again viewing profiles along selected areas. The profiles in the following sections show the original differences we calculated along with the new, corrected, differences we've calculated along profiles similar to those used in the previous section.

Road Profiles

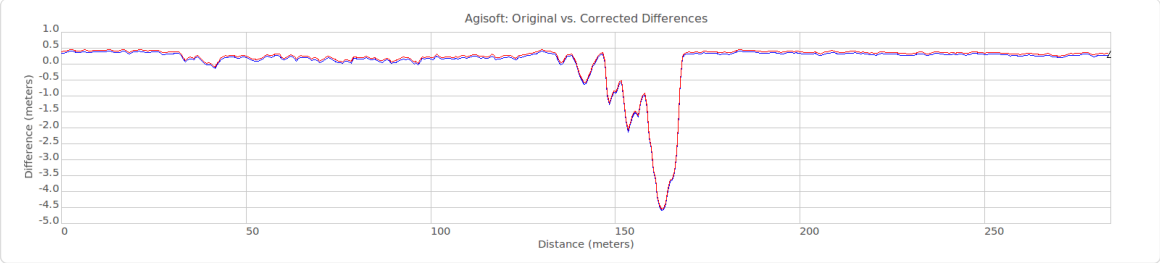


Agisoft: Original (red) vs. Corrected (blue)

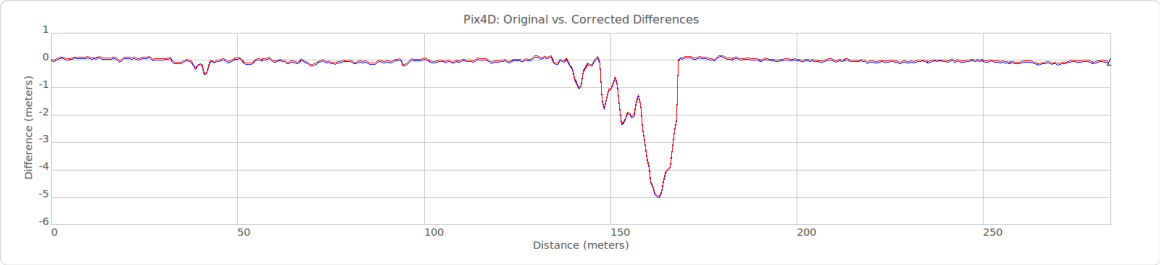


Pix4D: Original (red) vs. Corrected (blue)

Southern Field Profiles



Agisoft: Original (red) vs. Corrected (blue)



Pix4D: Original (red) vs. Corrected (blue)

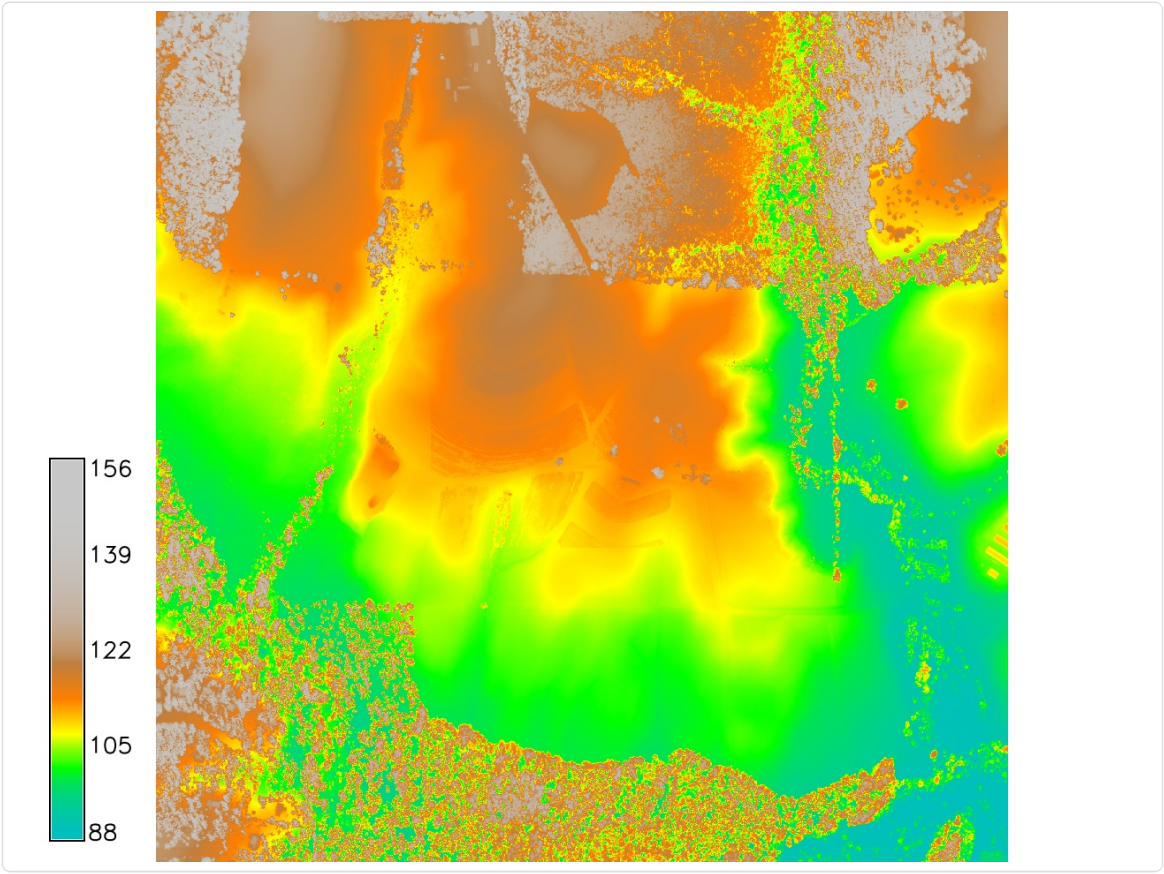
Patching UAS DSM with LIDAR DSM

In this section, we'll be patching the UAS DSM with data from the LIDAR DSM. We'll use the corrected LIDAR DSM in order to minimize hard jumps at the edges of the patch. Start by selecting a region within the UAS DSM that has very little distortion or differences from the corrected LIDAR DSM and clipping it down to only include that region. I'll be using the Pix4D DSM.

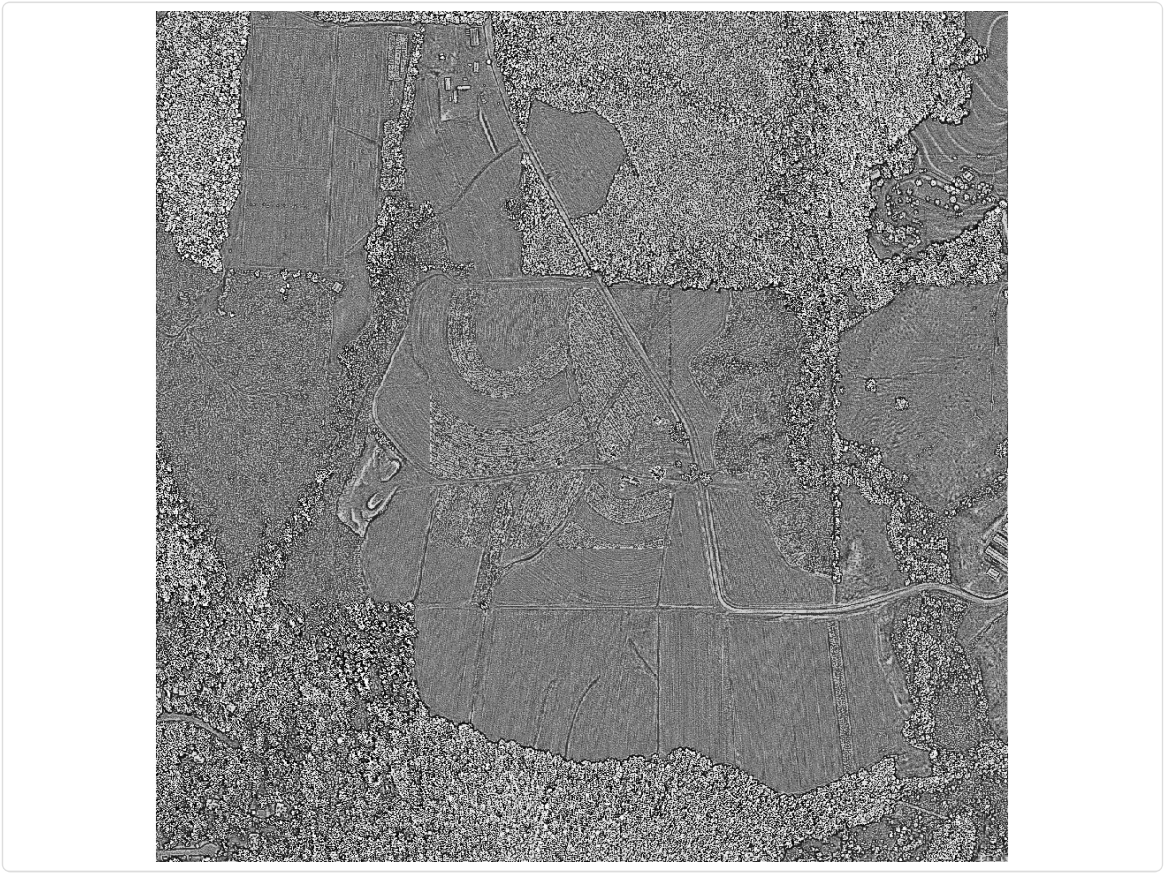
```
g.region n=219720.9 s=219257.1 w=636762.9 e=637190.4 res=0.3 -p
r.mapcalc "pix_clipped = 2015_06_20_pix4d_11GCP_dsm"
```

Now, patch the DSM with data from the corrected LIDAR DSM. We can use `r.local.relief` to check for these patch edges. Note that the resolution of both the LIDAR DSM and the UAS DSM are the same.

```
g.region rast=corrected_lidar -p
r.patch pix_clipped,corrected_lidar out=lidar_uas_patch
r.local.relief lidar_uas_patch out=lidar_uas_patch_relief
```



The Pix4D DSM Patched with the LIDAR DSM



Local Relief of the Patched Pix4D DSM

The edges of the patch are clearly visible in both images. The first image, colored by elevation with histogram equalization enabled, reveals the edges most clearly in the orange regions, especially on the western side of the patch. The second image, which shows the local relief, highlights the edge for just about all of the patch. To try to minimize this edge, we'll first try to compute the raster using an overlap between the UAS and LIDAR DSMs. Inside the overlap we'll average the values from both the UAS and LIDAR DSMs. First, set the region to include a 6 meter overlap:

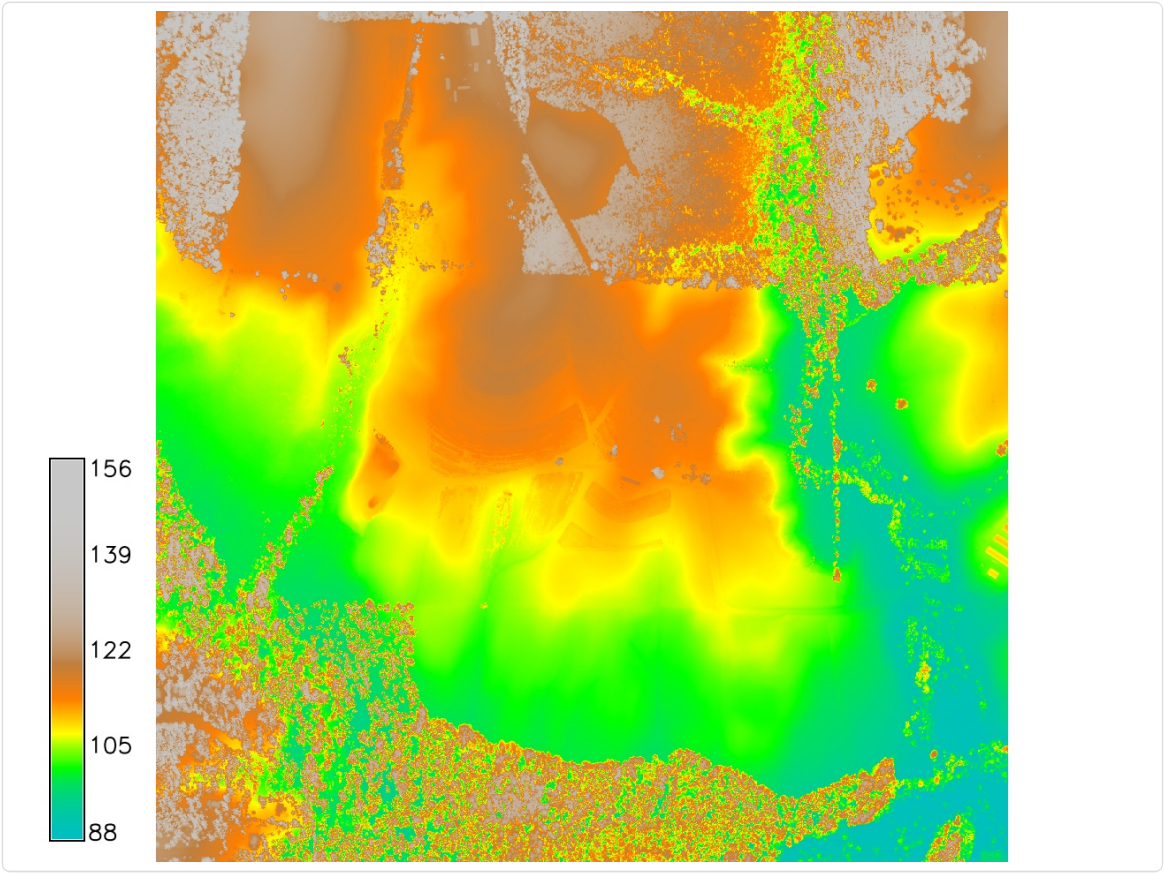
```
g.region rast=pix_clipped -p
g.region s=s-6 n=n+6 e=e+6 w=w-6 -p
```

Now, within this region, compute a raster with the LIDAR and UAS averages in the overlap and UAS values everywhere else:

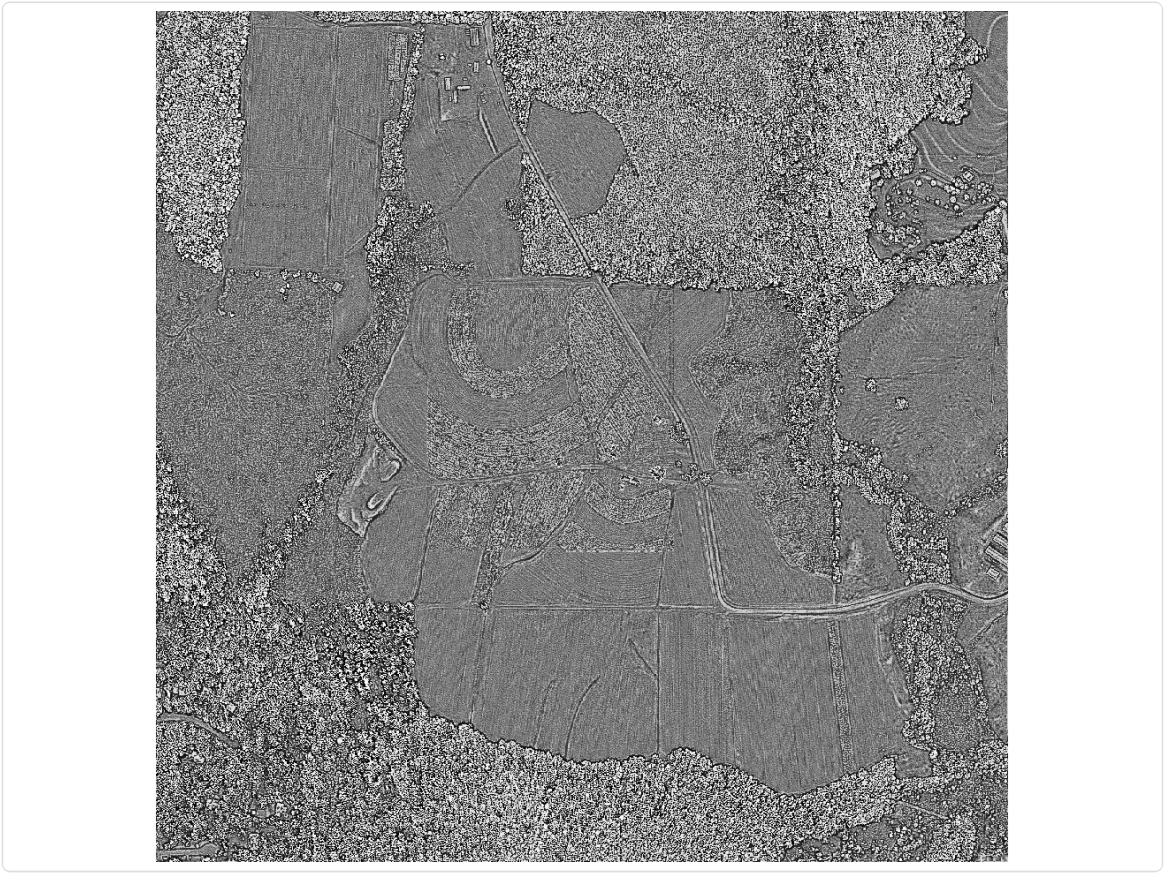
```
r.mapcalc "lidar_uas_overlap_averages = if(isnull(pix_clipped),
(2015_06_20_pix4d_11GCP_dsm+corrected_lidar)/2.,pix_clipped)"
```

Now merge the `lidar_uas_overlap_averages` raster, which contains UAS data on the interior and averaged data on the overlap, with the LIDAR DSM for the entire study area. Once again, we'll check for the edges using `r.local.relief`.

```
g.region rast=corrected_lidar
r.mapcalc "lidar_uas_patch_overlap =
if(isnull(lidar_uas_overlap_averages), corrected_lidar,
lidar_uas_overlap_averages)"
r.local.relief lidar_uas_patch_overlap
out=lidar_uas_patch_overlap_relief
```

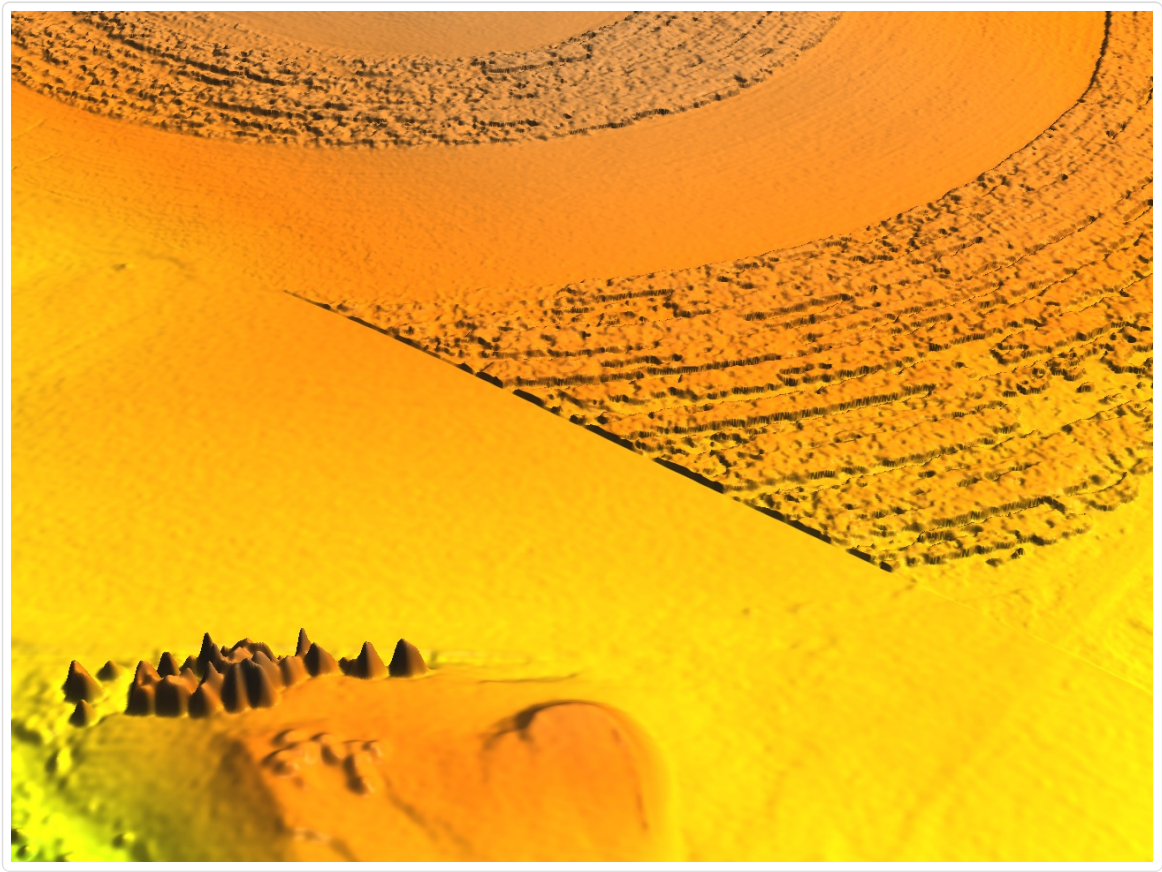


The Pix4D DSM patched with the LIDAR DSM, using average values in a 6m overlap

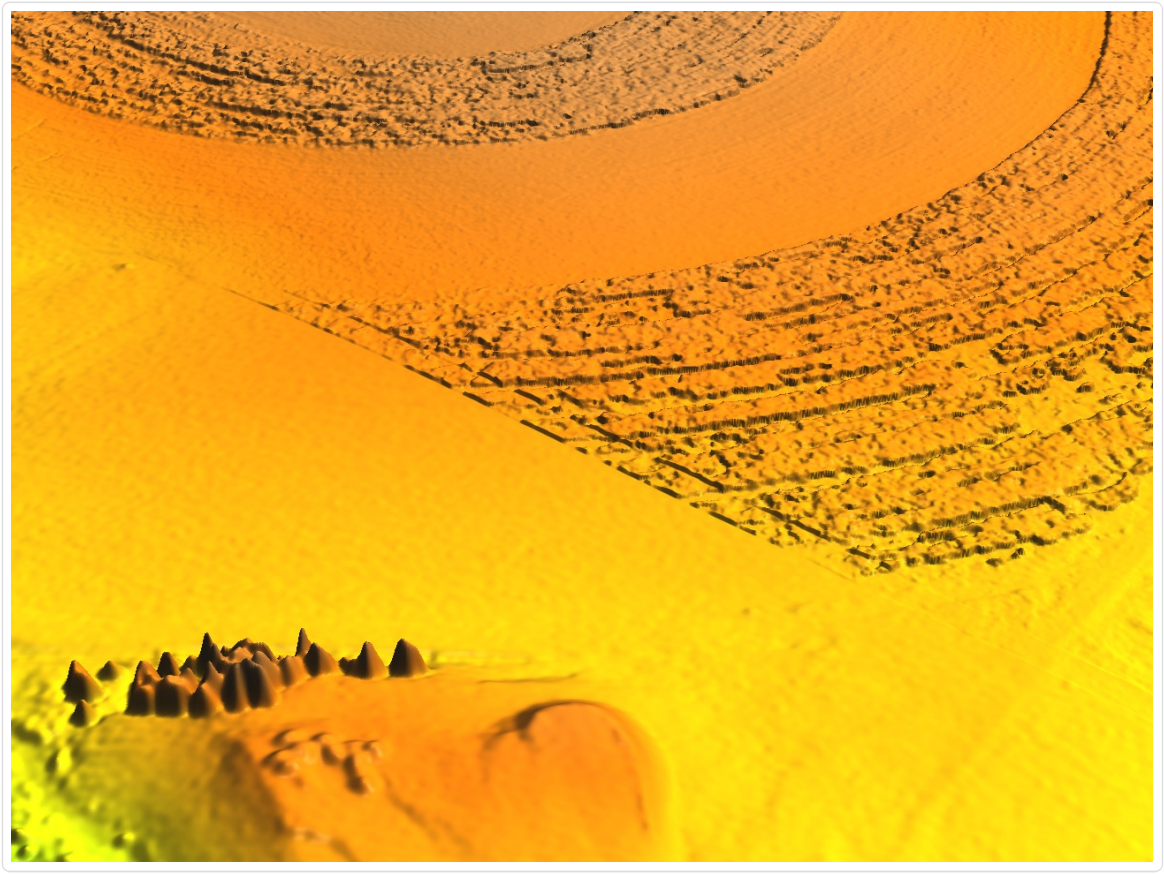


Local relief of the patched Pix4D DSM, using average values in a 6m overlap

We now have a smoother overlap, but the edges are still visible. We can see the edges especially well if we view the maps in 3D. The 3D view also reveals that the raster with the averages in the overlapped area appears to now have two edges, and interior and an exterior edge.



The Pix4D DSM patched with the LIDAR DSM



The Pix4D DSM patched with the LIDAR DSM, using average values in a 6m overlap

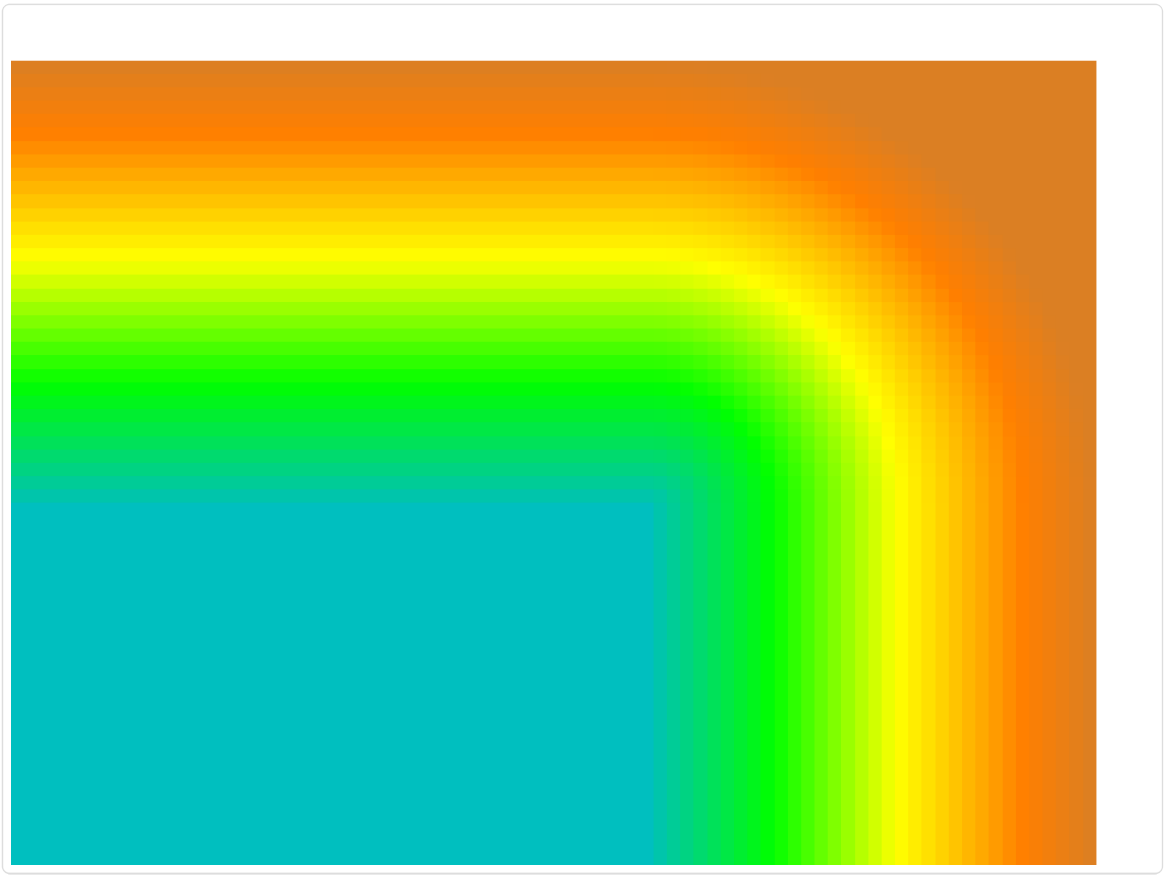
So how do we transition smoothly from the UAS data to the LIDAR data? We'll have to use a weighted average in the area of the overlap. What this means is that in the area of overlap, we want the average value that is calculated in any given cell to be weighted according to it's distance from either edge of the overlap. Cells that are very close to the inner edge (i.e. the UAS data) should give heavy weight to the value of the UAS data in that cell when calculating the average. On the flip side, cells that are very close to the outer edge (i.e. the LIDAR data) should give heavy weight to the value of the LIDAR data in that cell when calculating the average. In this example, we'll use a 10 meter overlap. First, we'll need to set the region to 10m smaller on each side than our clipped UAS DSM:

```
g.region g.region rast=pix_clipped -p
g.region s=s+10 n=n-10 e=e-10 w=w+10 -p
```

Next, we'll create a raster map that includes the distance from the UAS data (i.e. the interior data) of each cell in the overlap. First, create a dummy raster of all ones that fills our interior region. Then, create a new raster called `distance` using `r.grow.distance`. What this will do is fill any cells that are null (i.e. all of the ones in the overlap area) with a value that corresponds to the distance to the nearest non-null cell (i.e. the box of dummy ones we created). Finally, any distances that are greater than 10, which will be in the corners of our overlap (think Pythagorean Theorem) will need to be bumped down to 10 so that they can be used in the calculations in the next step.

```
r.mapcalc "dummy = 1"
```

```
r.grow.distance -m input=dummy distance=distance
r.mapcalc "distance = if(distance>10,10,distance)" --overwrite
```

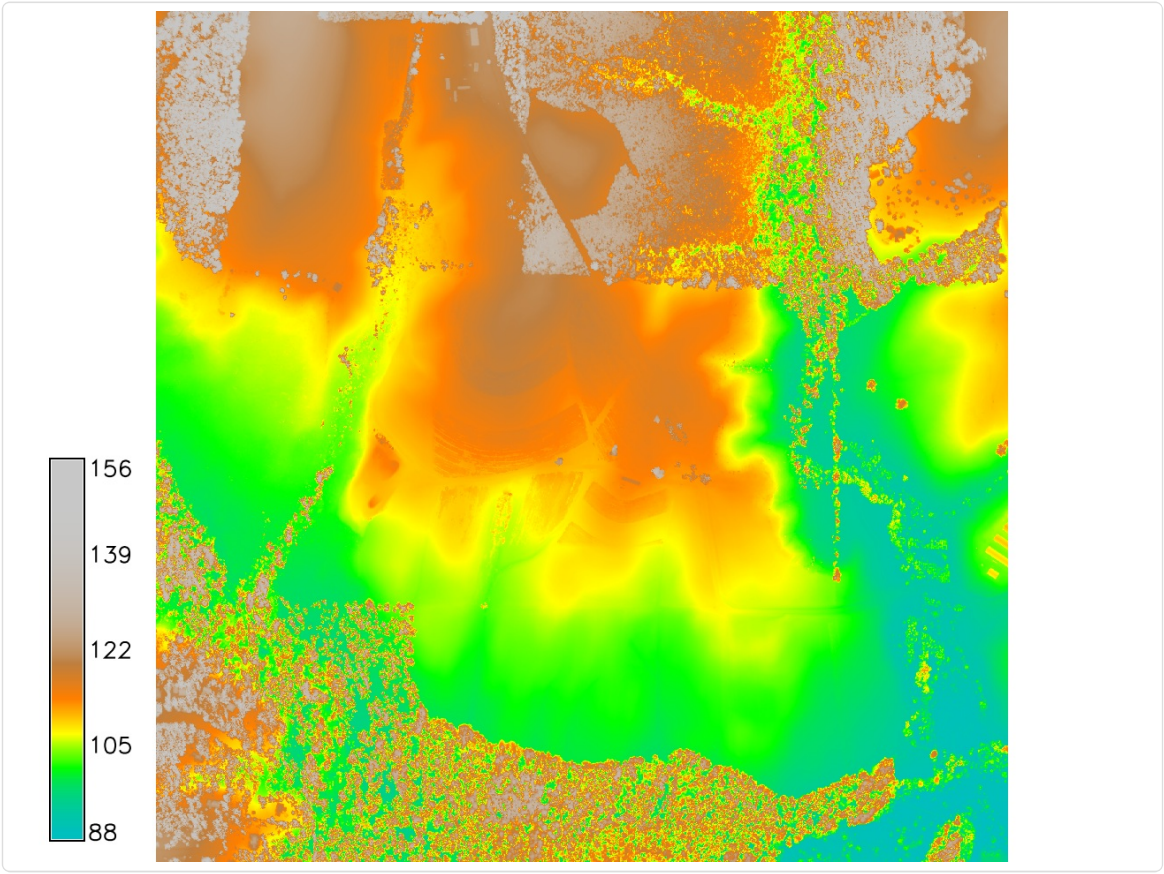


The `distance` raster. Values range from 0 (blue) to 10 (dark orange).

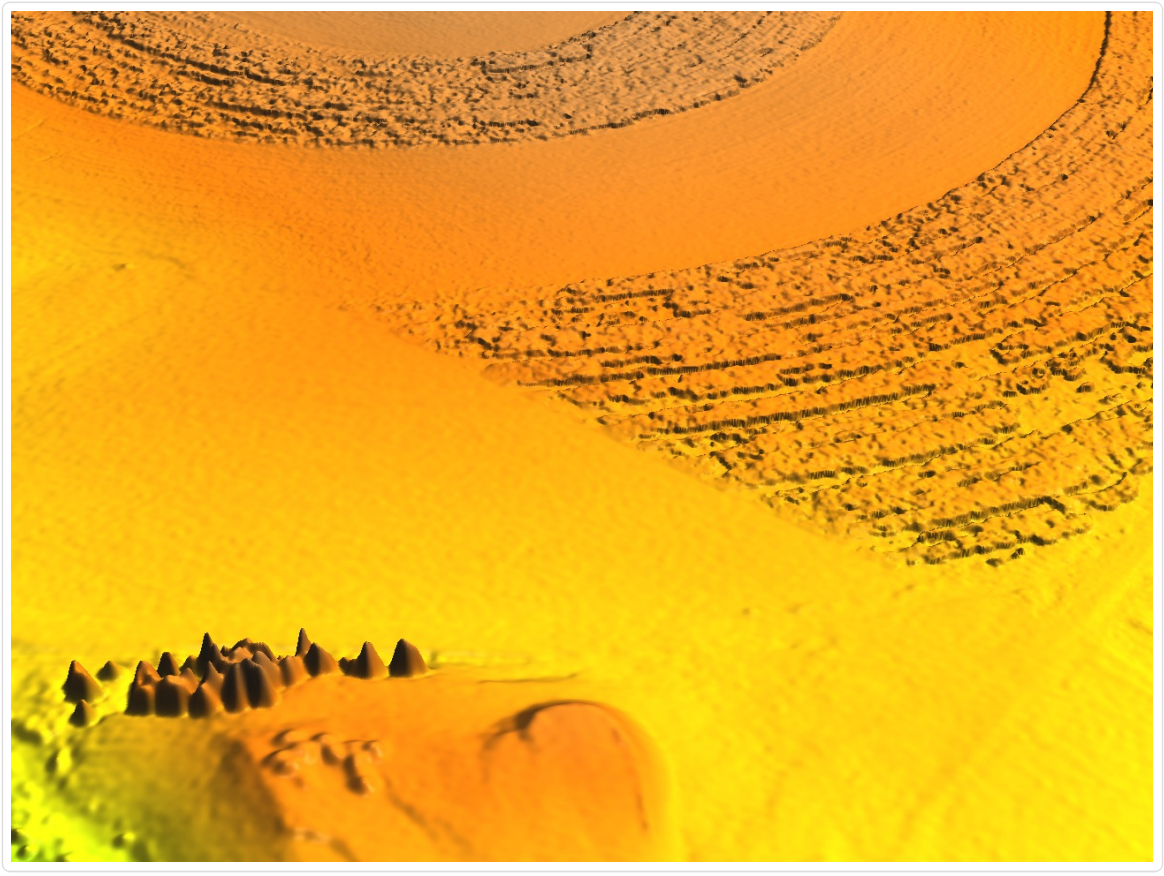
Finally, reset the region to that of the LIDAR DSM and calculate the weighted averages:

```
g.region rast=corrected_lidar -p
r.mapcalc "lidar_uas_patch_smooth = if(isnull(distance),
corrected_lidar, (1-distance/10.)*pix_clipped +
(distance/10.)*corrected_lidar)"
```

Now we have a much smoother transition between the UAS DSM and the LIDAR DSM, as can be seen in the following 2D and 3D images:



The Pix4D DSM patched with the LIDAR DSM using a weighted average in a 10m overlap



The Pix4D DSM patched with the LIDAR DSM using a weighted average in a 10m overlap

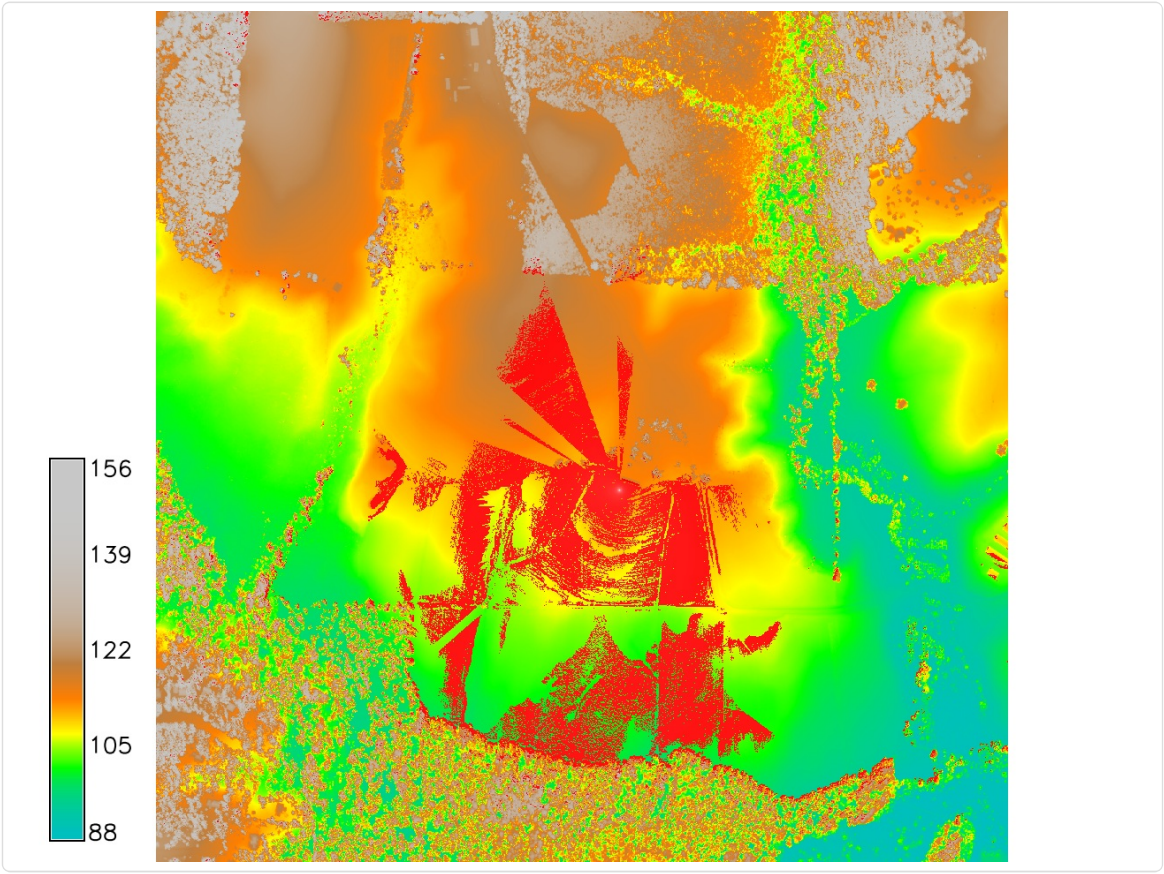
Viewshed Analysis

Next, we'd like to determine what a camera would be able to see throughout the growing season if it were permanently installed. Because we have DSMs with both grown crops and without any crops, we can compare the visibility in both scenarios. Additionally, we can expand the area of the dataset that includes crops (the UAS data) with the LIDAR DSM to get a better idea for the viewshed. We'll use the provided polygon (in `fields.pack`) to extract the fields from the UAS DSM and apply patching.

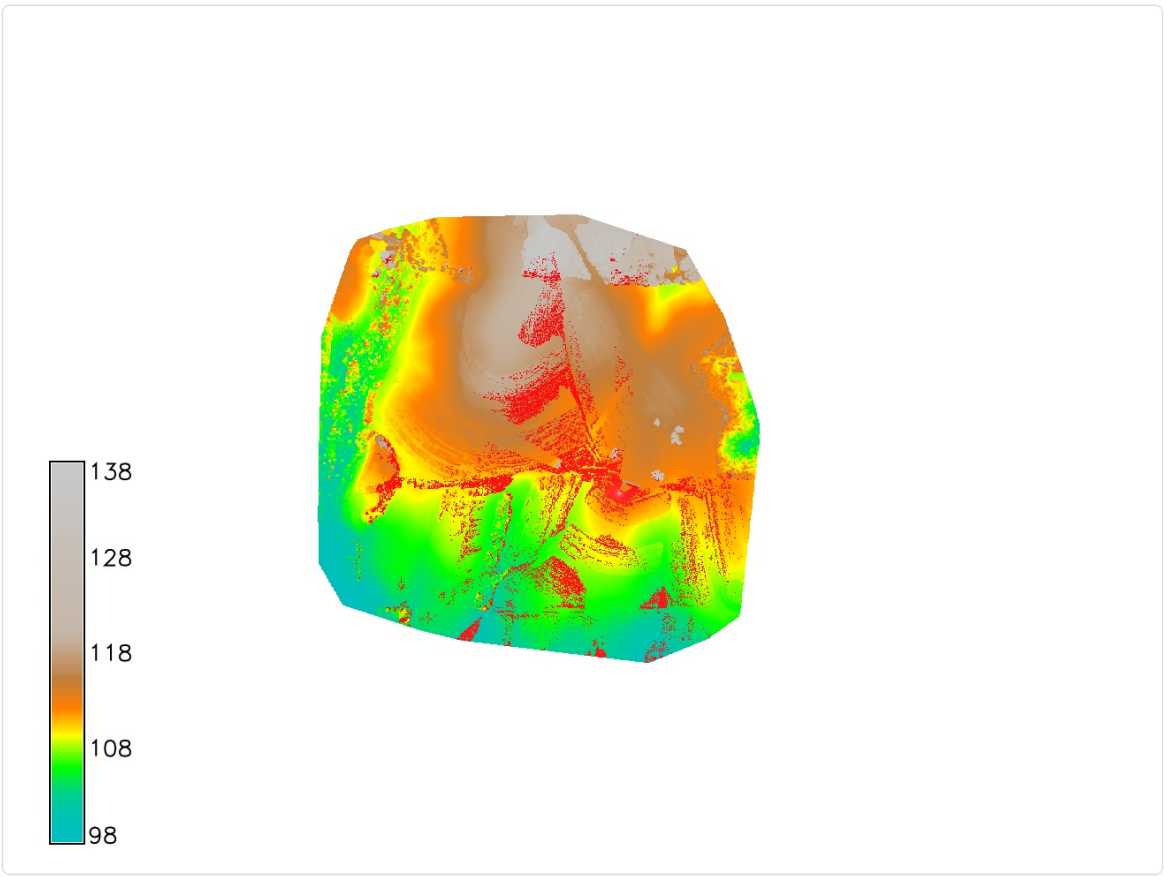
```
r.unpack -o fields.pack
r.mapcalc "uas_fields = 2015_06_20_pix4d_11GCP_dsm * fields"
r.patch uas_fields,corrected_lidar out=lidar_uas_patch_fields
```

Next, create the viewshed rasters using `r.viewshed`. Note that I have it set to use up to 10GB of RAM. Make sure you set this value to something appropriate for your machine.

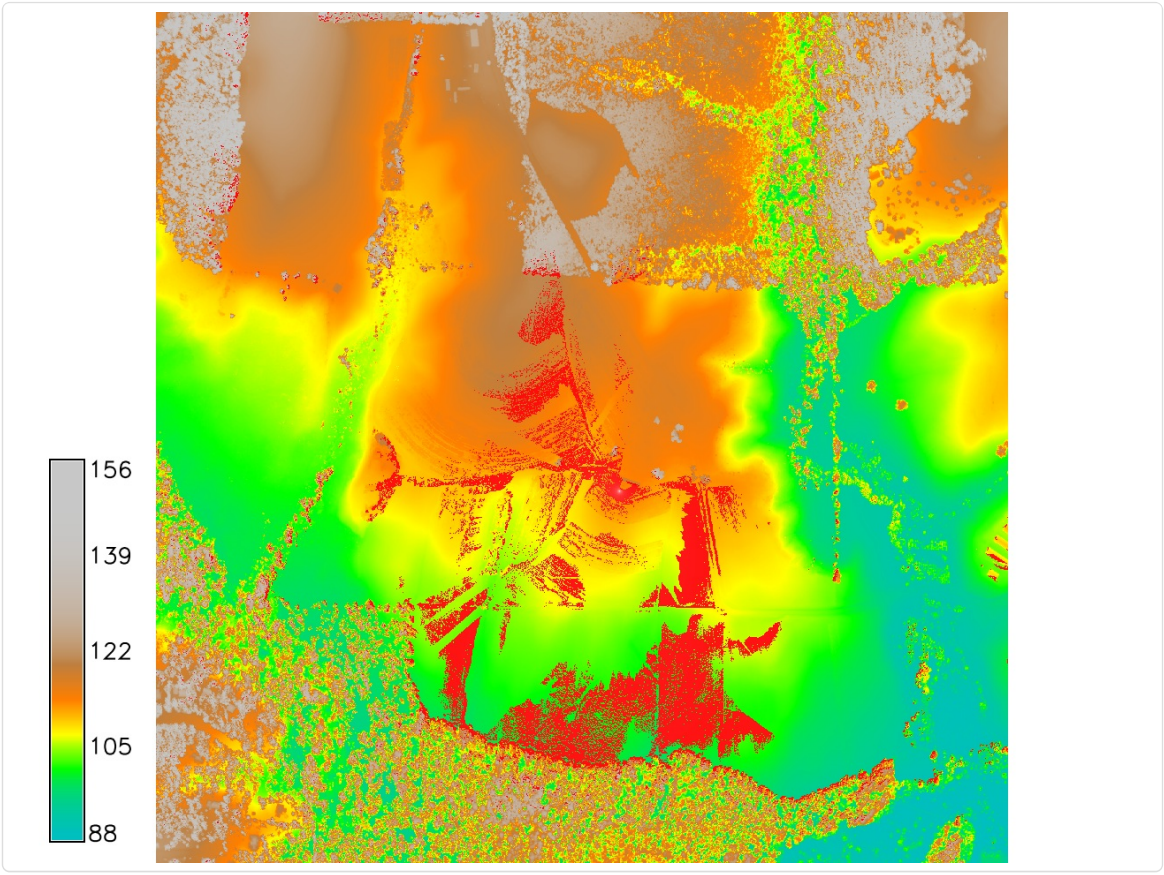
```
r.viewshed input=corrected_lidar output=viewshed_lidar
coordinates=637100,219360 memory=10000
r.viewshed input=2015_06_20_pix4d_11GCP_dsm output=viewshed_uas
coordinates=637100,219360 memory=10000
r.viewshed input=lidar_uas_patch_fields output=viewshed_lidar_uas
coordinates=637100,219360 memory=10000
```



The corrected LIDAR DSM viewshed

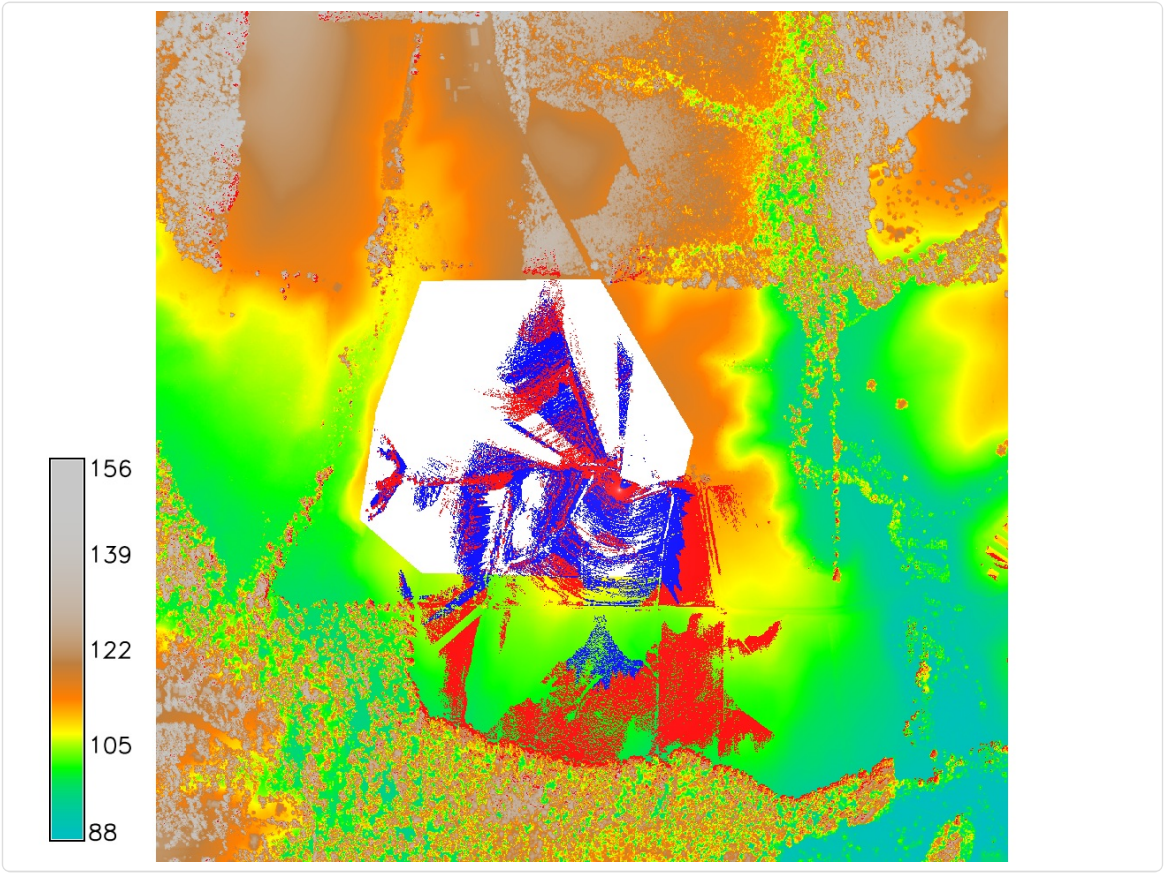


The Pix4D DSM viewshed

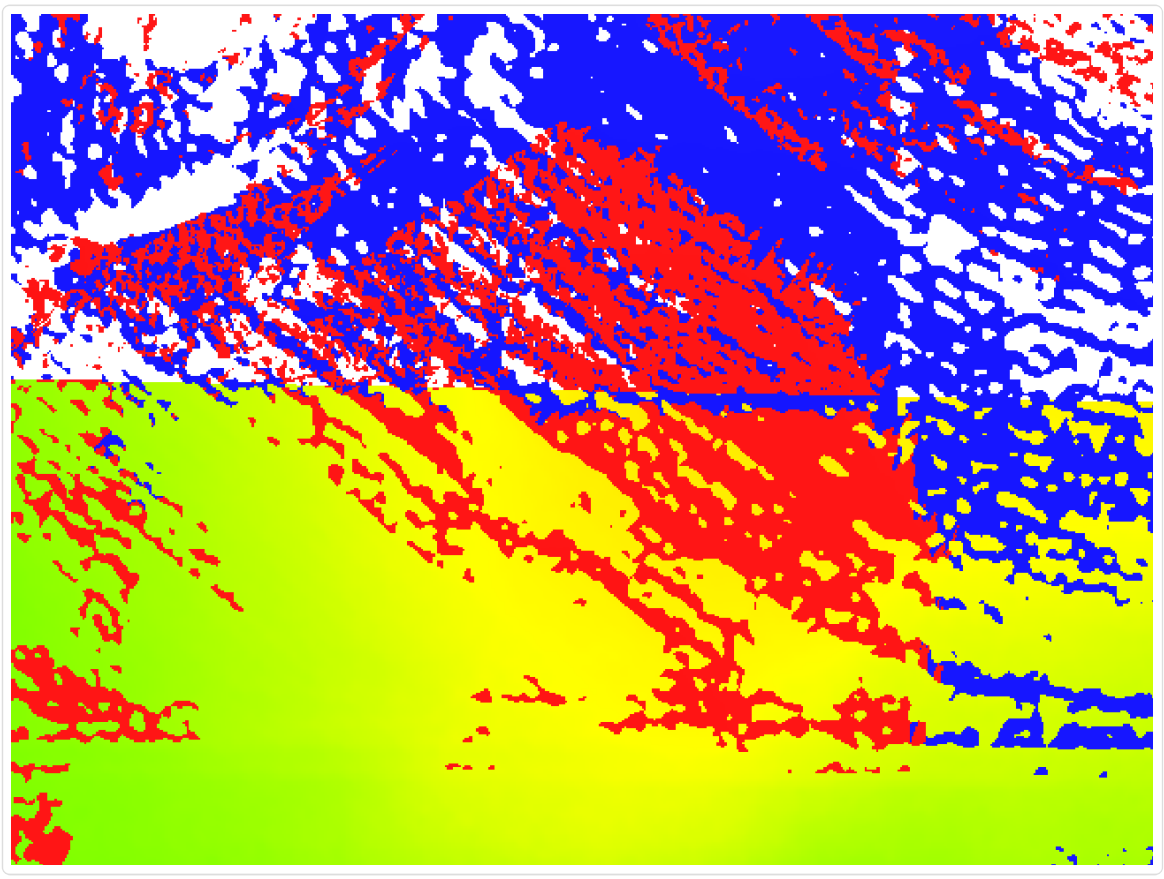


The combined viewshed

Note that there is much more visibility in the LIDAR DSM compared to the UAS DSM. This is because there was taller vegetation in the area when the UAS was flown, vegetation that blocks the view of anything behind it. Additionally, if we take a look at the combined viewshed (in blue) with the LIDAR viewshed (in red) on top of the outline of the fields polyon, we can see that the edge where the patching occurred is visible in the combined viewshed.



The combined viewshed (red) and LIDAR viewshed (blue) with the patch area (white)



Zoomed in view to show edge of patch is visible in the combined viewshed (red)