1 Package Installation

Make sure that you have the appropriate packages installed. LASTools is a commercial product, but a 2020 trial version (with limited number of points that can be processed) is available on their webpage.

PDAL can be installed via miniconda. Download the appropriate installer for your system (Windows, Mac, Linux). On a Linux system, you can use

```
wget https://repo.anaconda.com/miniconda/Miniconda3-py310_23.3.1-0-Linux-x86_64.sh
```

and then install this with `./Miniconda3-py310_23.3.1-0-Linux-x86_64.sh`.

Next, You can install the required packages (also all Python packages that will be used during different processing steps).

NOTE: If you are on Windows system, you will need to replace \ with ^ or put everything on one line.

```
conda create -n SDA python=3.10
conda install tqdm pip scipy pandas numpy matplotlib scikit-image gdal ipython \
    statsmodels jupyter pyproj lastools pykdtree h5py plotly seaborn pytables \
    pdal python-pdal scikit-learn jupyterlab numba jupyter-resource-usage geopandas \
    rasterio xarray dask netCDF4 bottleneck lmfit xlrd -c conda-forge
conda activate SDA
pip install open3d laspy laszip jakteristics structure_tensor

#Add kernel to Jupyter Lab/Notebook:
python -m ipykernel install --user --name=SDA
```

2 Lidar Hope Fault - 2014 (OpenTopography)

You will need to have installed a recent version of PDAL and LAStools. Note that LAStools is a commercial product!

We are going to work with the file `Lidar_HopeFault2014_Opentopo.laz`.

2.1 LASTools

2.1.1 Tiling

First step is to tile the data into 200 m tiles with 10 m overlap and remove all classification and synthetic flags.

```
mkdir tiles_200m
wine /opt/LAStools/bin/lastile.exe -i Lidar_HopeFault2014_Opentopo.laz  \
    -set_synthetic_flag 0 -set_classification 0 -olaz -tile_size 200 -buffer 10 -odir  \
    tiles_200m -flag_as_withheld
```

There are some difficulties using the 64-bit version on some unix derivatives and `wine`.

If you are using lastools on a windows computer, you may want to set the path to the directory containing the binary files and use:

```
mkdir tiles_200m
lastile64.exe -i Lidar_HopeFault2014_Opentopo.laz -set_synthetic_flag 0  \
    -set_classification 0 -olaz -tile_size 200 -buffer 10 -odir tiles_200m  \
    -flag_as_withheld
```

2.1.2 Noise filtering

Noise should be sparse - in most cases, the standard parameters will work. If you expect more outliers or if your data have a higher noise level, you may need to adjust the filtering parameters. It is generally a good idea to perform some noise filtering.

```
mkdir tiles_200m_noise
wine /opt/LAStools/bin/lasnoise.exe -classify_as 7 -cores 12 -i tiles_200m/*.laz -olaz  \
    -odir tiles_200m_noise -odix _n
```

If you want to verify the noise classification, combine into one file:

```
wine /opt/LAStools/bin/las2las.exe -i tiles_200m_noise/*n.laz -olaz -merged -o  \
    Lidar_HopeFault2014_Opentopo_noise.laz -drop_withheld
```

There are 8 noise pixels (taken from lasinfo).

```
wine /opt/LAStools/bin/lasinfo.exe Lidar_HopeFault2014_Opentopo_noise.laz
```

2.1.3 Ground-Classification

Second step is to perform a ground classification. We can do this for one tile first to verify the `lasground` parameters:

```
mkdir tiles_200m_ground
wine /opt/LAStools/bin/lasground64.exe -by_flightline -extra_fine -cores 12 -i  \
    tiles_200m_noise/683400_5297800_n.laz -olaz -odir tiles_200m_ground -odix g -step  \
    10 -offset 0.2
```

Specifically look into: `-not_airborne`, `-no_stddev`, or `-no_bulge`

You can fine tune with:

- `-spike 0.5` will remove up-spikes above 50 centimeter and down-spikes below 5 meters in the coarsest TIN)

- `-bulge 1.0` this parameter specifies how much the TIN is allowed to bulge up when including points as it is getting refined. The default bulge is one tenth of the step for step sizes larger than 5 meters and one fifth of the step otherwise. Especially for ground-classification of non-LiDAR points such as dense-matching or photogrammetry output by Agisoft of Pix4D the fine-tuning of this parameter can produce great results.
- `-stddev 10` describes the maximal standard deviation for planar patches in centimeter

By default the tool only considers the last return. Earlier returns are considered non-ground. You can turn this off by requesting '-all_returns'.

View interpolated hillshade We can look at the interpolated grid - use hillshade! - of this tile:

```
mkdir tiles_200m_ground_hs_tif
wine /opt/LAStools/bin/blast2dem.exe -keep_class 2 -drop_withheld -hillshade -odir  \
    tiles_200m_ground_hs_tif -i tiles_200m_ground/683400_5297800_ng.laz -otif
```

If the parameters are fine, let's ground-classify all files on multiple cores:

```
wine /opt/LAStools/bin/lasground.exe -by_flightline -extra_fine -cores 12 -i  \
    tiles_200m_noise/*.laz -olaz -odir tiles_200m_ground -odix g -step 10 -offset 0.2
```

2.1.4 Merge Files

Next, you can create a merged laz file only containing the ground-classified points:

```
wine /opt/LAStools/bin/las2las.exe -i tiles_200m_ground/*ng.laz -olaz -merged  \
    -keep_class 2 -o Lidar_HopeFault2014_Opentopo_groundonly.laz -drop_withheld
```

Or create a file with all points:

```
wine /opt/LAStools/bin/las2las.exe -i tiles_200m_ground/*ng.laz -olaz -merged -o  \
    Lidar_HopeFault2014_Opentopo_ground.laz -drop_withheld
```

2.1.5 Generate hillshade and DEM

```
wine /opt/LAStools/bin/blast2dem.exe -hillshade -i  \
    Lidar_HopeFault2014_Opentopo_groundonly.laz -o  \
    Lidar_HopeFault2014_Opentopo_ground_UTM59S_hs.tif

wine /opt/LAStools/bin/blast2dem.exe -i Lidar_HopeFault2014_Opentopo_groundonly.laz -o  \
    Lidar_HopeFault2014_Opentopo_ground_UTM59S.tif
```

2.1.6 Alternative approach using only one file

If you already know the classification parameters, you can run this in one step. *But beware*: This is much slower, because it runs only on one core. It is always better to tile the data and use a multi-core approach. This also allows to easier indicate problematic areas.

```
wine /opt/LAStools/bin/lasnoise.exe -classify_as 7 -i  \
    Lidar_HopeFault2014_Opentopo_cl0.laz -olaz -o Lidar_HopeFault2014_Opentopo_cl0n.laz
wine /opt/LAStools/bin/lasground.exe -ignore_class 7 -by_flightline -extra_fine -i  \
    Lidar_HopeFault2014_Opentopo_cl0n.laz -o Lidar_HopeFault2014_Opentopo_cln2.laz  \
    -olaz -step 10 -offset 0.2

wine /opt/LAStools/bin/blast2dem.exe -keep_class 2 -hillshade -i  \
    Lidar_HopeFault2014_Opentopo_cln2.laz -o  \
    Lidar_HopeFault2014_Opentopo_ground_UTM59S_hs.tif

wine /opt/LAStools/bin/blast2dem.exe -keep_class 2 -i  \
    Lidar_HopeFault2014_Opentopo_cln2.laz -o  \
    Lidar_HopeFault2014_Opentopo_ground_UTM59S.tif
```

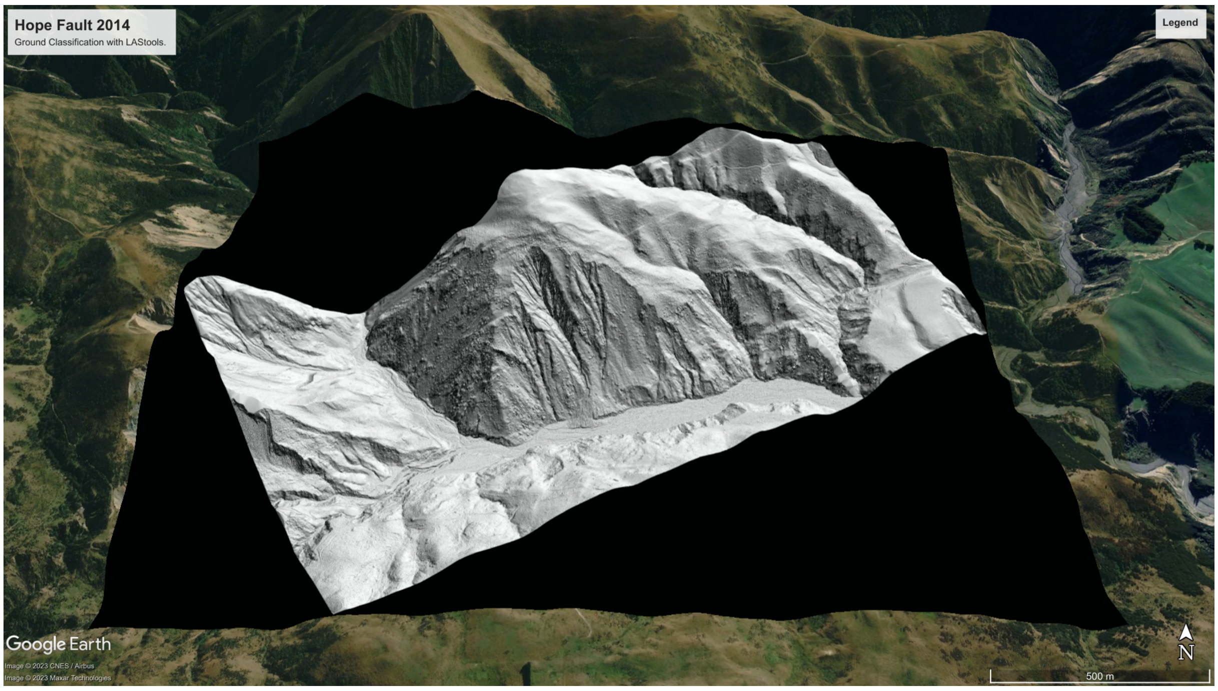
Hope Fault 2014
Ground Classification with LAStools.
Legend
Google Earth
Image © 2023 CNES / Airbus
Image © 2023 Maxar Technologies
N
500 m

Figure 1: Hillshade of LAStools Classification in Google Earth.

2.1.7 Additional processing: height above ground

It is helpful to calculate the height above ground to better identify problematic areas. We use lasheight for this that stores these data in the userdata field (scaled with a factor of 10.0, quantized and clamped into an unsigned char between 0 and 255).

```
wine /opt/LAStools/bin/lasheight.exe -cores 12 -i tiles_200m_ground/*ng.laz -olaz  \
    -odix h -odir tiles_200m_ground
```

If needed, you can create a normalized file by replacing the actual height in z with the height above ground:

```
wine /opt/LAStools/bin/lasheight.exe -replace_z -cores 12 -i tiles_200m_ground/*ng.laz  \
    -olaz -odix hn -odir tiles_200m_ground
```

and merge into one files:

```
wine /opt/LAStools/bin/las2las.exe -i tiles_200m_ground/*ngh.laz -olaz -merged -o  \
    Lidar_HopeFault2014_Opentopo_ground_hag.laz -drop_withheld

wine /opt/LAStools/bin/las2las.exe -i tiles_200m_ground/*nghn.laz -olaz -merged -o  \
    Lidar_HopeFault2014_Opentopo_ground_hagn.laz -drop_withheld
```

Or (probably most useful) store the height above ground in an extra byte in cm resolution (0.01 m):

```
wine /opt/LAStools/bin/lasheight.exe -store_as_extra_bytes -cores 12 -i  \
    tiles_200m_ground/*ng.laz -olaz -odix hne -odir tiles_200m_ground

wine /opt/LAStools/bin/las2las.exe -i tiles_200m_ground/*nghne.laz -olaz -merged -o  \
    Lidar_HopeFault2014_Opentopo_ground_hag_extrabyte.laz -drop_withheld
```

You will see the following field `attribute0`:

```
 data type: 4 (short), name "height above ground", description: "vertical point to TIN  \
     distance", scale: 0.01, offset: 250
```

If you want to run this on a single file, use:

```
wine /opt/LAStools/bin/lasheight.exe -ignore_class 7 -store_as_extra_bytes -i  \
    Lidar_HopeFault2014_Opentopo_cln2.laz -olaz -o  \
    Lidar_HopeFault2014_Opentopo_cln2_hag_extrabyte.laz
```

2.1.8 Height above ground (HAG) processing with PDAL

Important The height above ground calculated in LAStools is different than the HAG computed with the PDAL HAG filter. We will use the PDAL processing steps to make the results comparable. Alternatively, we can calculate HAG with LAStools, but it will be more straight forward using an open-source software system. Calculating HAG with PDAL can be done on the command line.

Note LAStools often adds a synthetic flag to the data and this is not understood by PDAL. It is best to first remove this flag (`-set_synthetic_flag 0`).

```
wine /opt/LAStools/bin/las2las.exe -set_synthetic_flag 0 -i  \
    Lidar_HopeFault2014_Opentopo_cln2.laz -olaz -o  \
    Lidar_HopeFault2014_Opentopo_cln2_nosynthetic.laz
pdal translate Lidar_HopeFault2014_Opentopo_cln2_nosynthetic.laz  \
    Lidar_HopeFault2014_Opentopo_cln2_hag_pdal.laz hag_delaunay  \
    --writers.las.extra_dims="HeightAboveGround=float32"
```

2.1.9 Generate interpolates views of Lidar attributes

The gridding tool `las2dem` comes in handy to interpolate lidar attributes to a grid. For example, generate a view of intensity:

```
wine /opt/LAStools/bin/las2dem64.exe -i  \
    Lidar_HopeFault2014_Opentopo_cln2_hag_extrabyte.laz -o  \
    Lidar_HopeFault2014_Opentopo_intensity_UTM59S.tif -intensity -step 1  \
    -compute_min_max -gray
```

or the height above ground stored in attribute 0:

```
wine /opt/LAStools/bin/las2dem64.exe -i  \
    Lidar_HopeFault2014_Opentopo_cln2_hag_extrabyte.laz -o  \
    Lidar_HopeFault2014_Opentopo_HAG_UTM59S.tif -attribute 0 -step 1 -set_min_max 0 20  \
    -false
```

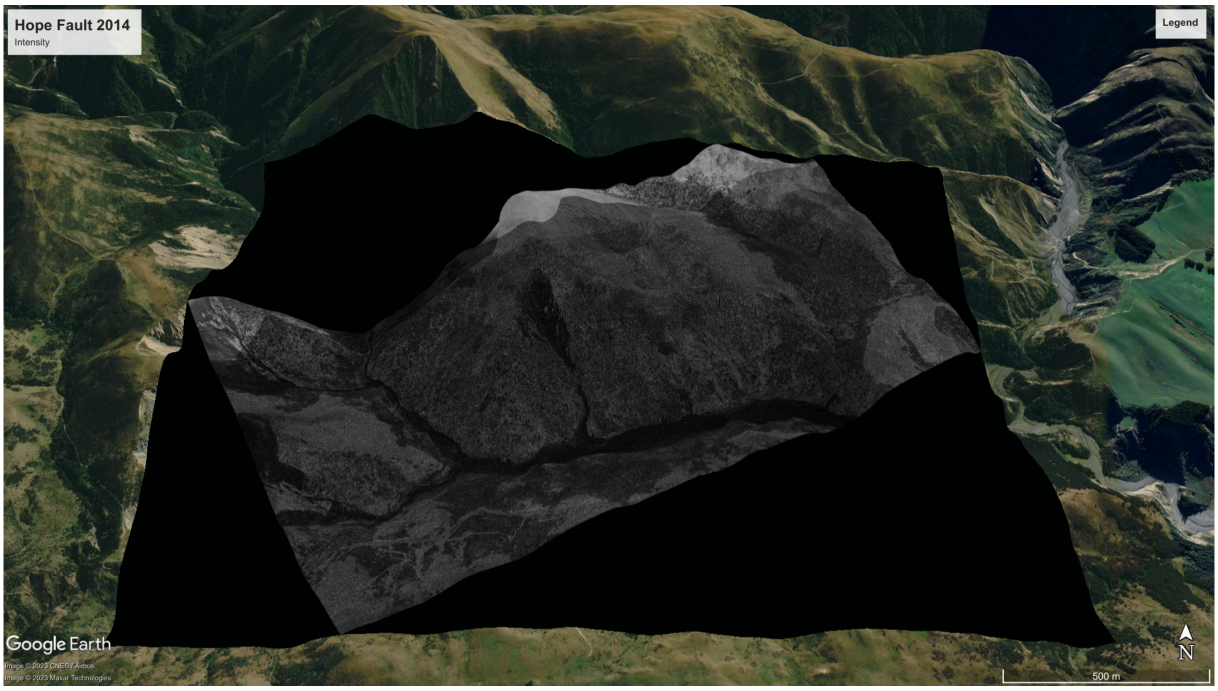

Figure 2: Intensity interpolation created with las2dem.exe in LAStools. View in Google Earth.

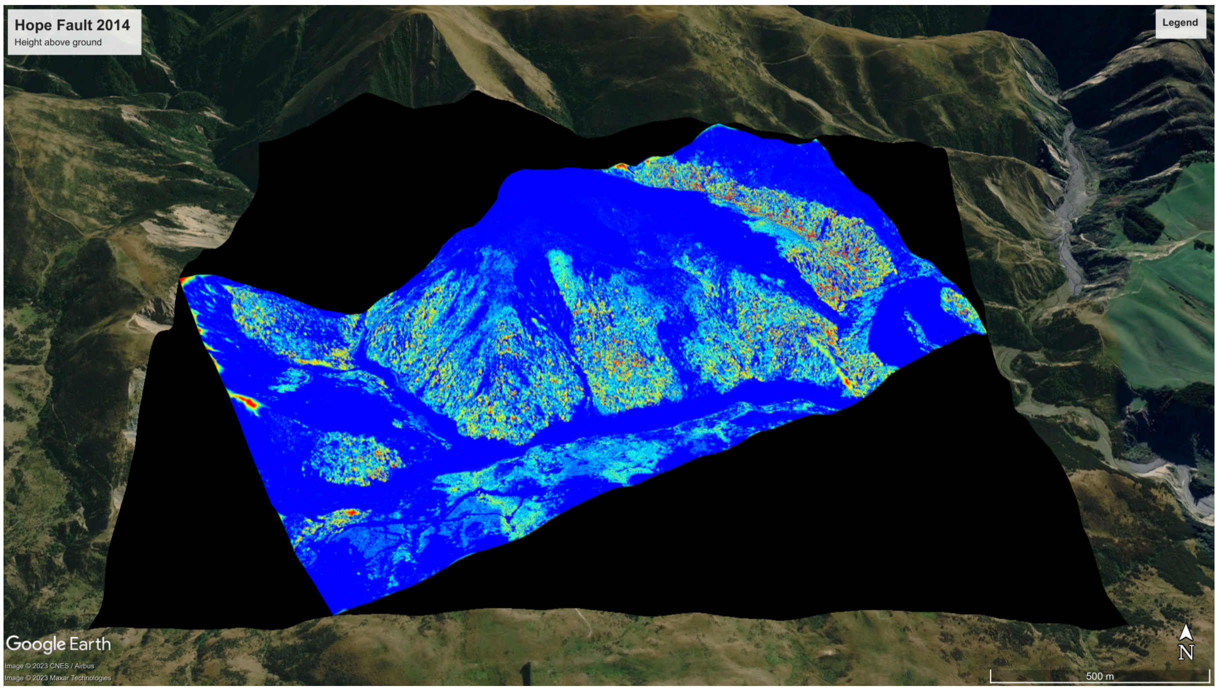

Figure 3: Height above ground (attribute 0) interpolation created with las2dem.exe in LAStools. View in Google Earth.

2.2 PDAL

2.2.1 Simple Morphological Filter

If you want a strong outlier filtering, you may want to apply the Extended Local Minimum (ELM) filter to mark low points as noise. In this case, we will only apply a statistical outlier filter. We also apply a height above ground calculation. This relies on the ground points generated by the SMRF step, performs a triangulation and calculates the height above this surface. The height is assigned to every point not classified as ground. Here we write the height above ground into an extra dimension and the option is given on the command line (could also be included in the .json command file).

```
pdal translate Lidar_HopeFault2014_Opentopo.laz Lidar_HopeFault2014_Opentopo_smrf.laz  \
    --json Lidar_HopeFault2014_Opentopo_smrf.json  \
    --writers.las.extra_dims="HeightAboveGround=float32"
```

The height above ground step can also be invoked on the command line - but it is generally more efficient to have one `.json` file that combines all steps

```
pdal translate Lidar_HopeFault2014_Opentopo_withground.laz  \
    Lidar_HopeFault2014_Opentopo_withground_hag_delaunay.laz hag_delaunay \
    --writers.las.extra_dims="HeightAboveGround=float32"
```

with Simple Morphological Filter file `Lidar_HopeFault2014_Opentopo_smrf.json`:

```
{
  "pipeline":[
    {
      "type":"filters.assign",
      "assignment":"Classification[:]=0"
    },
    {
      "type":"filters.outlier"
    },
    {
      "type":"filters.smrf",
      "ignore":"Classification[7:7]",
      "slope":0.2,
      "window":16,
      "threshold":0.45,
      "scalar":1.2
    },
    {
    "type":"filters.hag_delaunay"
    }
  ]
}
```

Additional Notes Add the following field, if you want to do more outlier filtering.

```
    {
      "type":"filters.elm"
    },
```

If you only want to write ground points, add the following snippet to the end:

```
    {
      "type":"filters.range",
      "limits":"Classification[2:2]"
    }
```

2.2.2 Progressive Morphological Filter

The PMF filter is faster than SMRF.

```
pdal translate Lidar_HopeFault2014_Opentopo.laz Lidar_HopeFault2014_Opentopo_pmf.laz  \
    --json Lidar_HopeFault2014_Opentopo_pmf.json  \
    --writers.las.extra_dims="HeightAboveGround=float32"
```

with Progressive Morphological Filter file `Lidar_HopeFault2014_Opentopo_pmf.json`:

```
{
  "pipeline":[
    {
      "type":"filters.assign",
      "assignment":"Classification[:]=0"
    },
    {
      "type":"filters.outlier"
    },
    {
      "type":"filters.pmf",
      "ignore":"Classification[7:7]",
      "cell_size": 1,
      "returns": "last"
    },
    {
    "type":"filters.hag_delaunay"
    }
  ]
}
```

3 SfM-UAV Classification

Structure from Motion data will require significant filtering. It is most useful to do some of this filtering within the processing software you are using to generate the point clouds (e.g., Metashape, OpenDrone Mapper). For example, filtering by number of depth pairs (called confidence interval in Metashape) is a useful approach: Only use points that have at least 5 depth pairs.

If you have generated a LAZ file with an attribute 'confidence' (several software packages allow to save this, for example Metashape), you can use this to filter your data.

We can first look at the 2D projected confidence values using `las2dem`:

```
wine /opt/LAStools/bin/las2dem64.exe -i Chattisgarh_Sakri_UAV_clip.laz -o  \
    Chattisgarh_Sakri_UAV_clip_confidence_UTM44N.tif -attribute 3 -step 1  \
    -compute_min_max -false
```

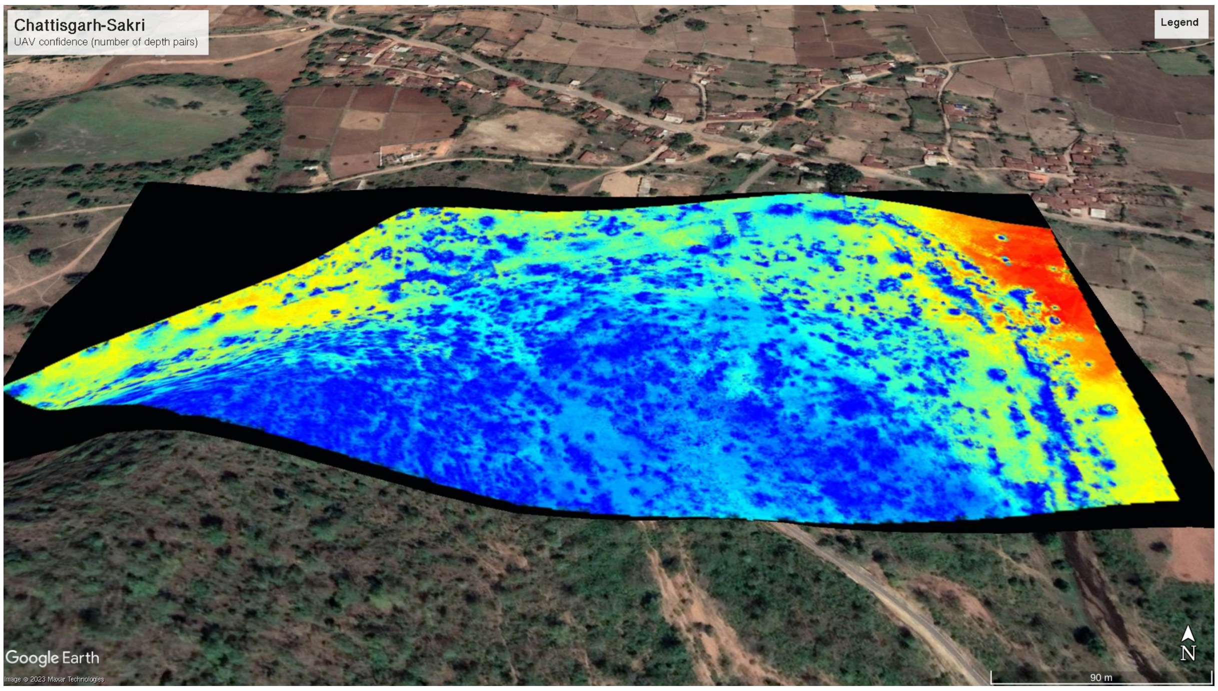

Figure 4: Confidence value for Sakri in Chattisgarh. Output from las2dem

If you have not performed the confidence filtering in your processing software, we can use `las2las.exe` to do this filtering step:

```
wine /opt/LAStools/bin/las2las.exe -drop_attribute_below 3 5 -i  \
    Chattisgarh_Sakri_UAV_clip.laz -olaz -o Chattisgarh_Sakri_UAV_clip_cf5.laz
```

Sometimes you will need to use a stronger (higher) filtering value:

```
wine /opt/LAStools/bin/las2las.exe -drop_attribute_below 3 10 -i  \
    Chattisgarh_Sakri_UAV_clip.laz -olaz -o Chattisgarh_Sakri_UAV_clip_cf10.laz
```

Now follow the standard classification steps as described above for the airborne lidar data. You may want to use the `-not_airborne` flag for lasground.

```
wine /opt/LAStools/bin/lasground.exe -i Chattisgarh_Sakri_UAV_clip_cf5.laz -olaz -o  \
    Chattisgarh_Sakri_UAV_clip_cf5_ground.laz
```