

3D_landslide_detection_using_CloudCompare

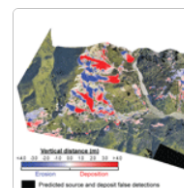
June 21, 2022

by *Thomas Bernard* (*thomas.bernard.rennes@gmail.com*)

Research article

26 Aug 2021

Beyond 2D landslide inventories and their rollover: synoptic 3D inventories and volume from repeat lidar data



Thomas G. Bernard, Dimitri Lague, and Philippe Steer 

Univ. Rennes, CNRS, Géosciences Rennes – UMR 6118, 35000 Rennes, France

Correspondence: Thomas G. Bernard (*thomas.bernard@univ-rennes1.fr*)

Received: 02 Sep 2020 – Discussion started: 15 Sep 2020 – Revised: 29 Jun 2021 – Accepted: 12 Jul 2021 – Published: 26 Aug 2021

Reference : Bernard et al., 2021 : <https://esurf.copernicus.org/articles/9/1013/2021/>

1 Presentation of the data

Location : The Towy-Robson catchment area

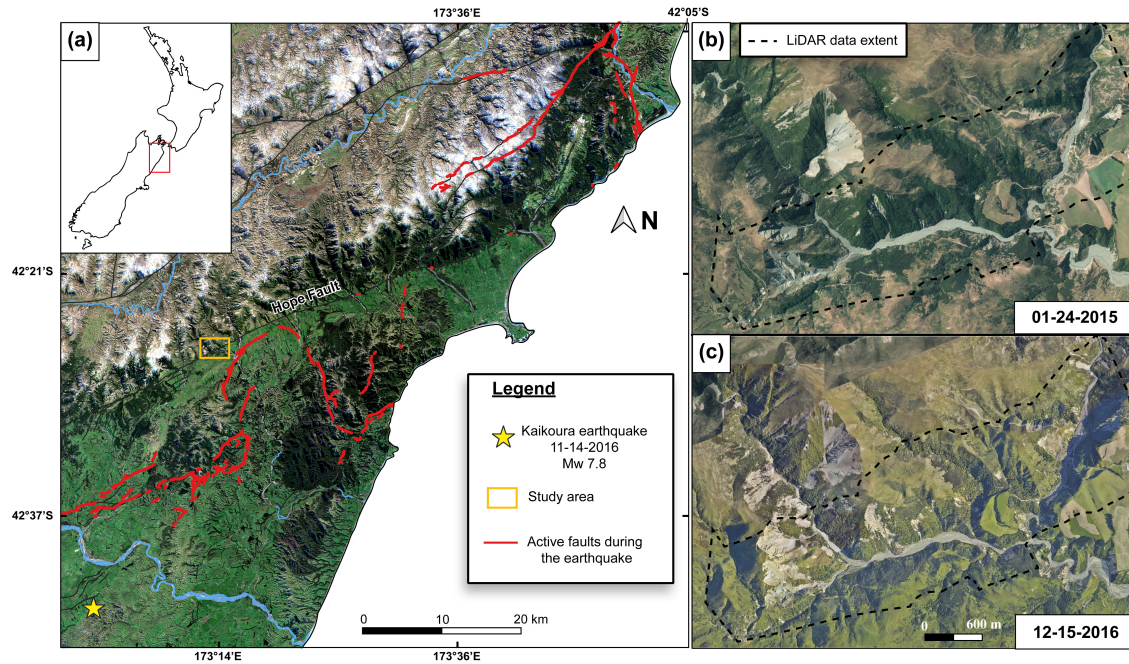


Table 1. Information on the lidar data used in this study.

	Pre-earthquake lidar	Post-earthquake lidar
Date of acquisition	13 Mar 2014–20 Mar 2014	3 Dec 2016–6 Jan 2017
Commissioned by/provided by	USC–UCLA–GNS science/NCALM	Land Information New Zealand/AAM NZ
Availability	https://doi.org/10.5069/G9G44N75	Upon request from https://canterburymaps.govt.nz/about/feedback/
Original point density (points m ⁻²)	9.02	19.2 ± 11.7
Number of ground points	10 660 089	63 729 096
Ground point density (points m ⁻²)	3.8 ± 2.1	11.5 ± 6.8
Vertical accuracy (m, as ±1 SD)	0.068–0.165	0.04
Study area (m ²)	5 253 133	5 253 133

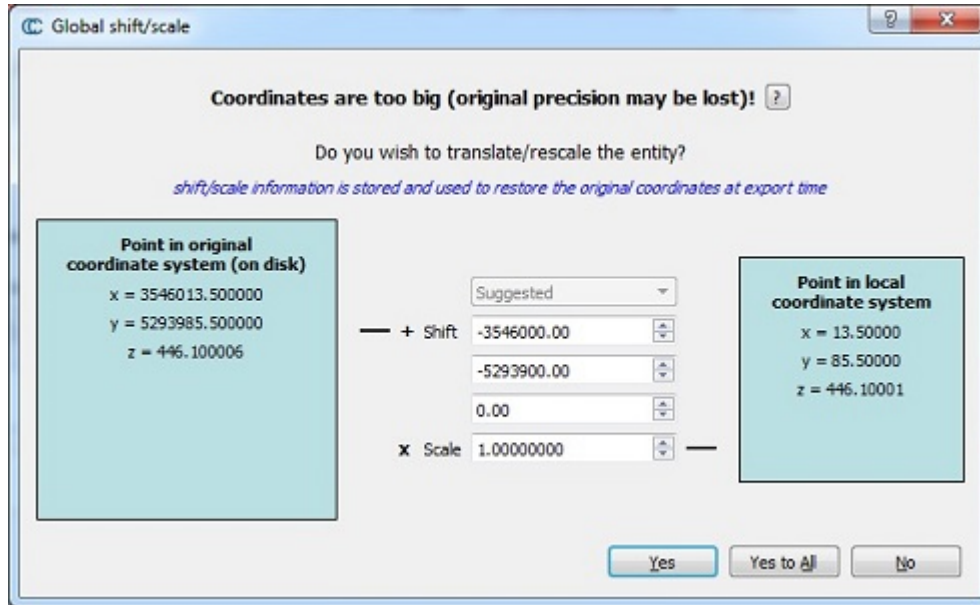
2 Before starting

2.1 Some good practice with CloudCompare:

1. Rename new point clouds with an explicit name
2. Clone point clouds before executing an irreversible operation
3. Save regularly

2.2 Set the global shift file

When loading (or generating) an entity with very big coordinates (typically greater than 105), CloudCompare will warn the user about this and suggest to shift (or rescale) the entity in order to work in a local coordinate system with smaller coordinates:



It is strongly advised to shift and scale the entities in this case. In “command line” mode, you have to modify the “global_shift_list.txt” file with your own shift generally located in ‘C:/Program Files/CloudCompare’.

The format is the following. All values are separated by a semicolon character (;): *name; shift(X); shift(Y); shift(Z); scale;*

3 Getting started

You can use your own data or download data from New Zealand here : <https://filesender.renater.fr/?s=download&token=b6cddecf-a810-4f01-a395-21ef1ba4e857>

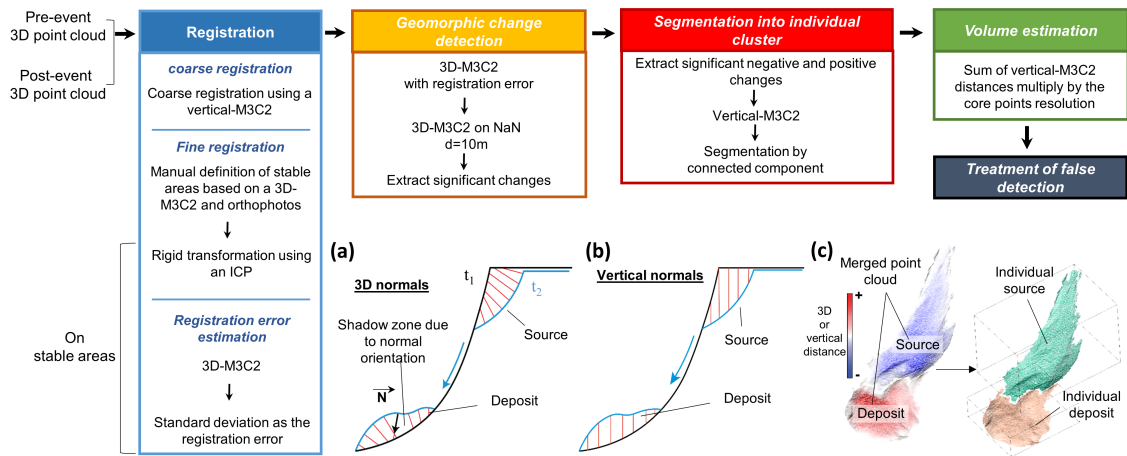
You can also download the orthophotos of the study area here (1.2 Go): <https://filesender.renater.fr/?s=download&token=70bc7a18-aaa9-4a0c-8fb9-6a93b4930bf1>

The folder contains:

- Pre_EQ_LiDAR_2014.laz : *LiDAR survey in 2014 along the hope fault collected by the National Center for Airborne Laser Mapping (available in Opentopography: <https://doi.org/10.5069/G9G44N75>)*
- Post_EQ_LiDAR_2016.laz : *LiDAR survey in 2017*
- Study_area_polyline : *polyline of the study area used to define the area of interest*
- river : *point cloud of the core points located in the river network.*
- core_points.laz : *the point cloud with the core points (but we will do our own core points)*
- GIS/:
 - . orthophotos/
 - . Kaikoura_EQ_Towy_river.qgz: *QGIS project with the orthophotos*

For what follows, I assume your data are already co-registered.


4 Workflow

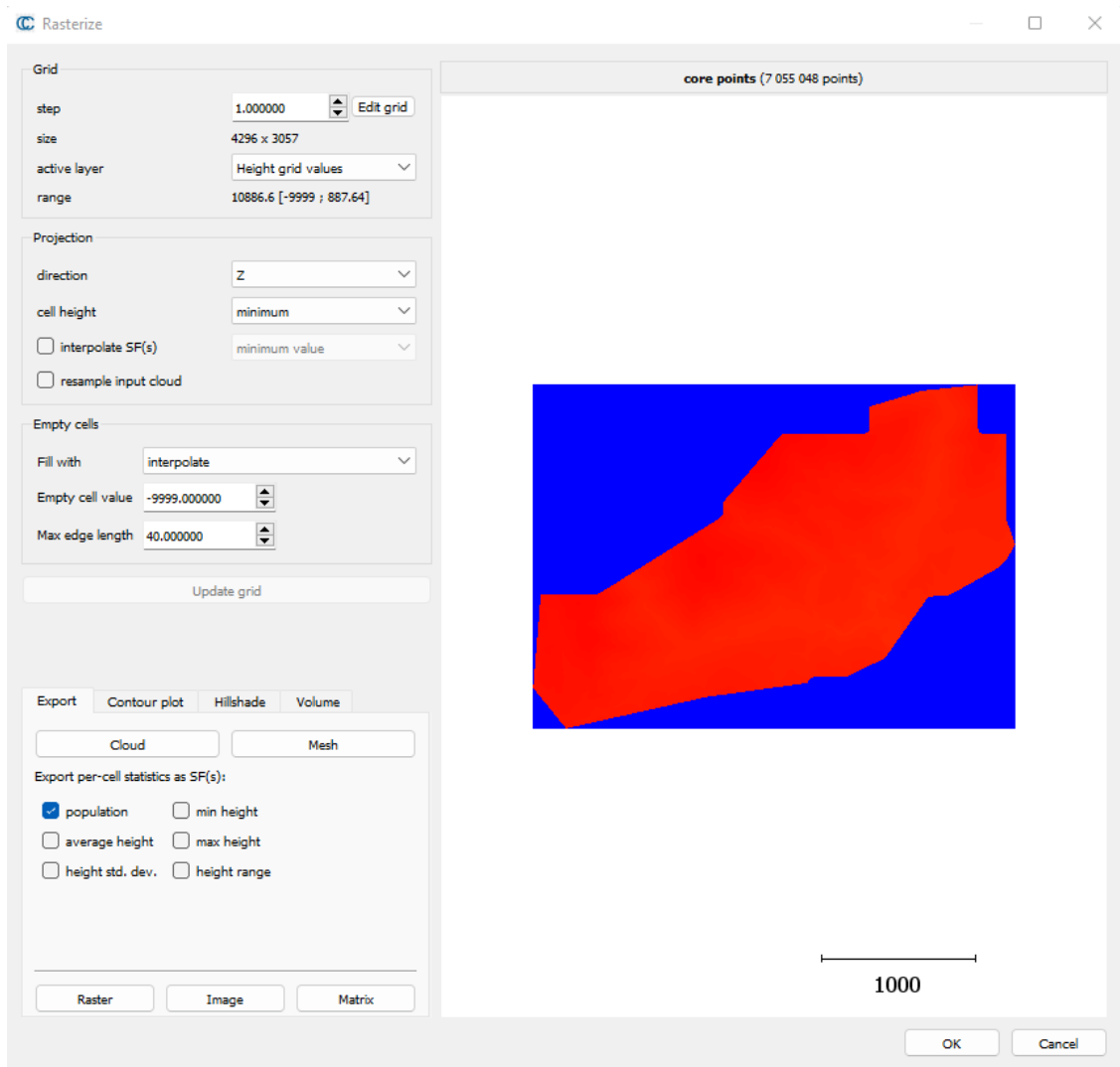


4.1 Creation of core points

While M3C2 can be applied on all points, the algorithm can use an accessory point cloud, called *core points*. In our case, core points constitute a regular grid with constant horizontal spacing generated by the rasterization of one of the two clouds.

Usually the core point should be created from the pre-EQ dataset to estimate normals. However, here we are going to create the core points from the pre-EQ dataset because the point density is much higher in the post-EQ point cloud and the classification is better.


Select the post-EQ point cloud and click on the Rasterize tool  and create a regular grid of 1 m spacing.

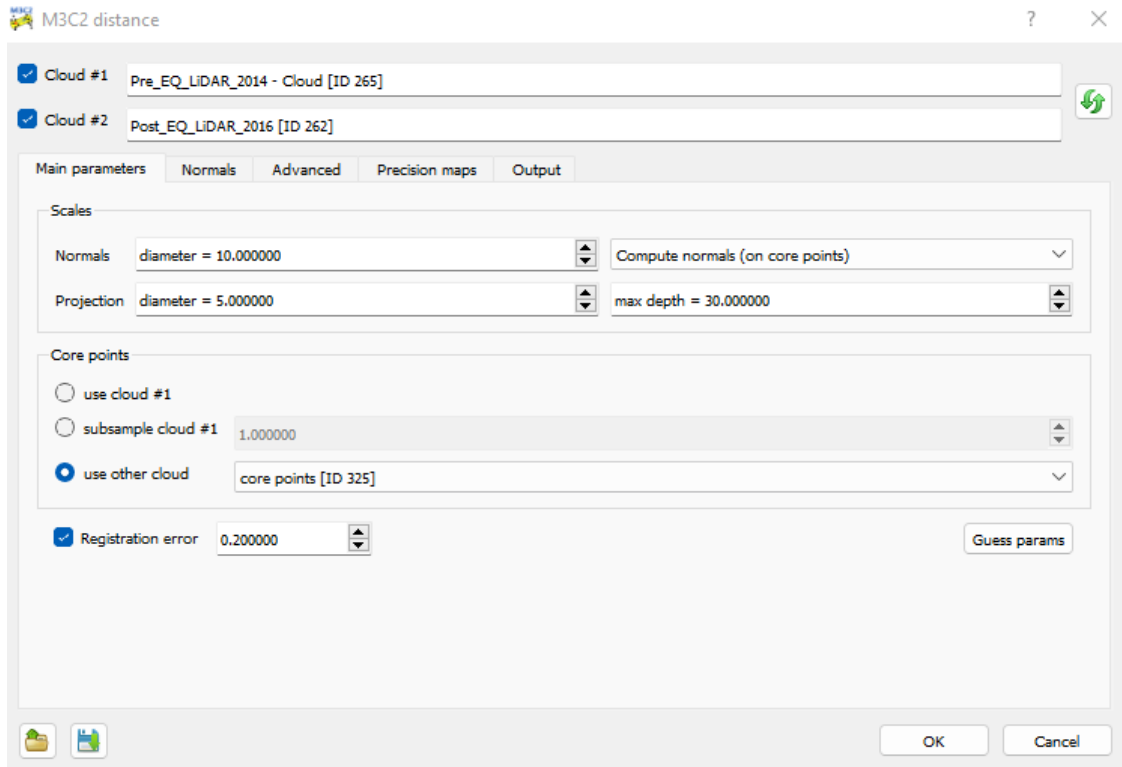


4.2 Geomorphic change detection

4.2.1 3D-M3C2 with the registration error

A bit of nicknames: M3C2 has the option of computing the distance vertically, which bypasses the normal calculation, and we use this option several times in the workflow. We use the abbreviation “*vertical-M3C2*” in these cases and “*3D-M3C2*” otherwise.


Click on the M3C2 plugin icon 




Estimation of the registration error → quality control on flight lines or standard deviation on stable areas

4.2.2 3D-M3C2 on NaN $d = 10\text{m}$

For core points in low-point-density areas, where a confidence interval could not be estimated due to insufficient points, a second application of 3D-M3C2 is performed at a larger projection scale $d = 10\text{ m}$. These core points generally correspond to ground points under canopy on steep slopes.

1. Display the SF “distance uncertainty” and click on the filtering tool icon .
2. Click on “Split”

It results in two point clouds : one with values and one with NaN.

3. Segment the NaN point cloud with the provided .shp file or with the segmentation tool  to delineate your area of interest.
4. Apply 3D-M3C2 with $d = 10\text{ m}$. The core points here is the NaN point cloud.

At this stage we have two point clouds: one with 3D-M3C2 results using a projection scale $d = 5\text{ m}$ and a second one with 3D-M3C2 results using $d = 10\text{ m}$.

4.2.3 (Optional) Applying multiple cylindre M3C2 : the approach

In M3C2, p_{max} must be chosen to be larger than the largest topographic change to be measured. Thus, p_{max} can be as large as several tens of meters in landslide inventories. This poses a potential issue in highly curved features of the landscape such as narrow ridges or gorges with steep flanks


where the cylinder can intercept the same point cloud twice (double-intercept issue), resulting in an incorrect distance calculation. Thus, we have modified the M3C2 algorithm to avoid the double-intercept issue. A new iterative procedure progressively increases the depth of the cylinder up to ***pmax***, by intervals of 1m for each core point, and checks for the stability of the measured distance: if the distance is stable for two successive iterations, it is considered as the final M3C2 distance for this core point.

The best approach is (a new version of M3C2 is in progress to do it automatically and will be available in Cloudcompare):

- While $p(i) < pmax$, $p(i+1) = 2 * p(i)$. Si $d5_p(i+1) = d5_p(i)$ -> add them to the final core points
- . Otherwise, we continue

However, it will be a bit long to do it in Cloucompare, so today we will apply a simpler approach which consists on the following step:

Steps in Cloudcompare:

1. Rename all SF with suffix `_p30`
2. 3D-M3C2 on $d=5m$ with $pmin=1.4m$ (keep normal with *'use core point normals'* and quote use original cloud)
3. Rename all newest SF with *'_p1.4'* at the end of the name
4. 3D-M3C2 on $d=5m$ with $pmin = 2.8m$ (keep normal with *'use core point normals'*)
5. Select the SF "distance_uncertainty" calculated with $p=2.8 m$ and split the point cloud using the filtering 
6. From the point cloud with NaN value, remove all SF except results from 3D-M3C2 with $p=30 m$ and rename the point cloud to *'3D-M3C2_p=30'*
7. On the other cloud: Edit > Scalar Field > Arithmetic -> $M3C2_distance(p=2.8m)/M3C_distance(p=1.4m)$
8. Extract all points with $M3C2_distance(p=2.8m)/M3C_distance(p=1.4m) \neq 1$
9. For the point cloud with $M3C2_distance(p=2.8m)/M3C_distance(p=1.4m) = 1$ -> M3C2 à 1.4m
10. All the other points -> 3D-M3C2 $pmin = 30 m$
11. Rename all SF
12. Merge all point clouds

Repeat these steps for 3D-M3C2 with $d = 10 m$.

4.2.4 (Optional) Re-calculate significant changes (available in the python version)

M3C2 provides the uncertainty in the computed distance at the 95% confidence level based on local roughness, point density and registration error as follows:

$$LoD_{95\%}(d) = \pm t(DF) \left(\sqrt{\frac{\sigma_1(d)^2}{n_1} + \frac{\sigma_2(d)^2}{n_2}} + reg \right)$$


with

$$DF = \frac{\left(\frac{\sigma_1(d)^2}{n_1} + \frac{\sigma_2(d)^2}{n_2} \right)^2}{\left(\frac{\sigma_1(d)^4}{n_1^2} + \frac{\sigma_2(d)^4}{n_2^2} \right)} \left(\frac{1}{(n_1-1)} + \frac{1}{(n_2-1)} \right)$$

where $LoD_{95\%}$ is the level of detection; t is the two-tailed t statistics with a confidence level of 95% and a degree of freedom DF (Borradaile, 2003); $\sigma_1(d)$ and $\sigma_2(d)$ are the standard deviation of distances of each cloud, at scale d ; measured along the normal direction; n_1 and n_2 are the number of points in each cloud at that scale; and reg is the co-registration error between the two epochs.

The original M3C2 algorithm uses a value of t equal to 1.96, the asymptotic value of the two-tailed t statistics when F is infinite, and the $LoD_{95\%}$ is only computed if $n_1 > 4$ and $n_2 > 4$. As will be shown later, the low point density of the pre-EQ data in forested areas resulted in values of n_1 varying between 5 and 15, in which case $t(F(n_1;n_2))$ is significantly larger than 1.96. For instance, when $n_1 = n_2 = 5$, $F = 4$ and $t = 2.776$. To avoid underpredicting $LoD_{95\%}$ in low-point-density areas, we choose to apply the strict two-tailed t statistics rather than the simplification used in the CloudCompare implementation of M3C2.



4.2.5 Extract significant changes

To extract significant changes, simply select the scalar field ‘Significant change’, then click on the filtering tool  and export all points with a value > 0 .

4.2.6 Extract topographic changes located in the river network


Changes located in the river bed, and likely specifically related to river dynamics and not to landslide deposits, are manually removed using the post-EQ orthoimage. Along with the selection of the stable areas, this is the only manual phase of the workflow.

2 possibilities:


1. Using the provided file “river.laz”: you can simply compute a cloud-to-cloud distance  for which the compared cloud is the ‘Significant changes’ and the reference cloud is the ‘river’ point cloud. Then remove all points with a Cloud-to-cloud distance value < 5 m
2. If you are working with your own dataset: Select the segmentation tool , delineate the river network and then extract the points located outside of the polygon you created.

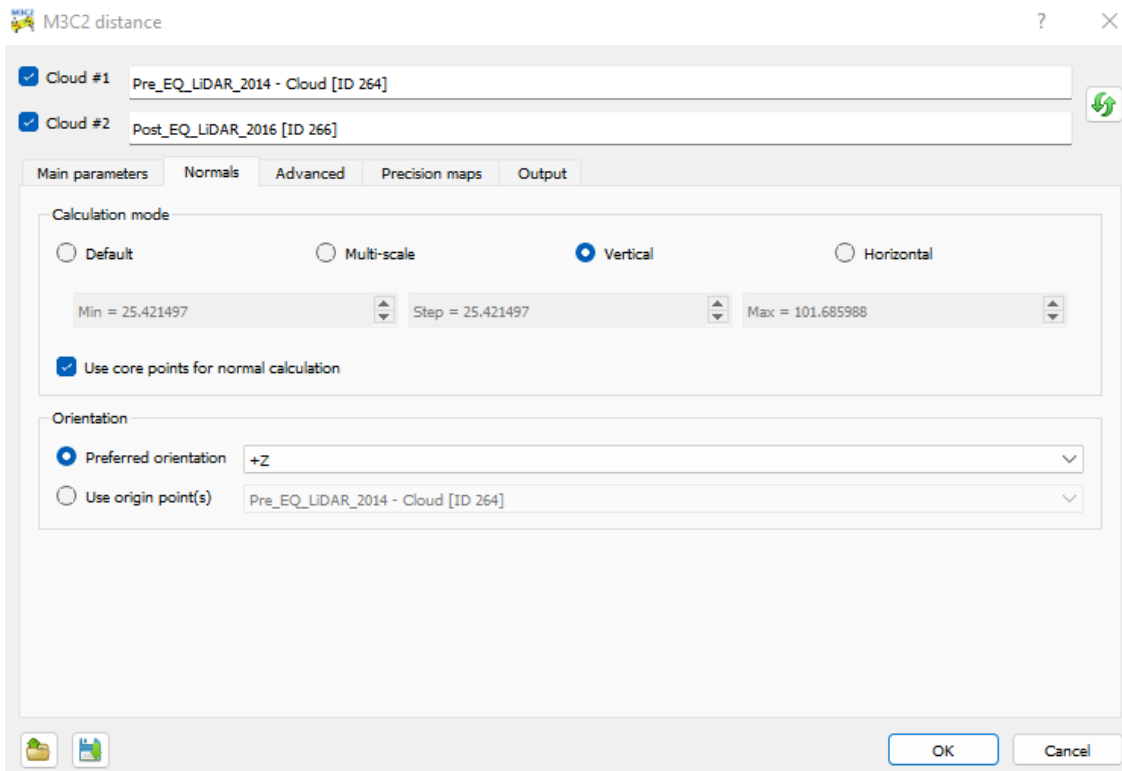
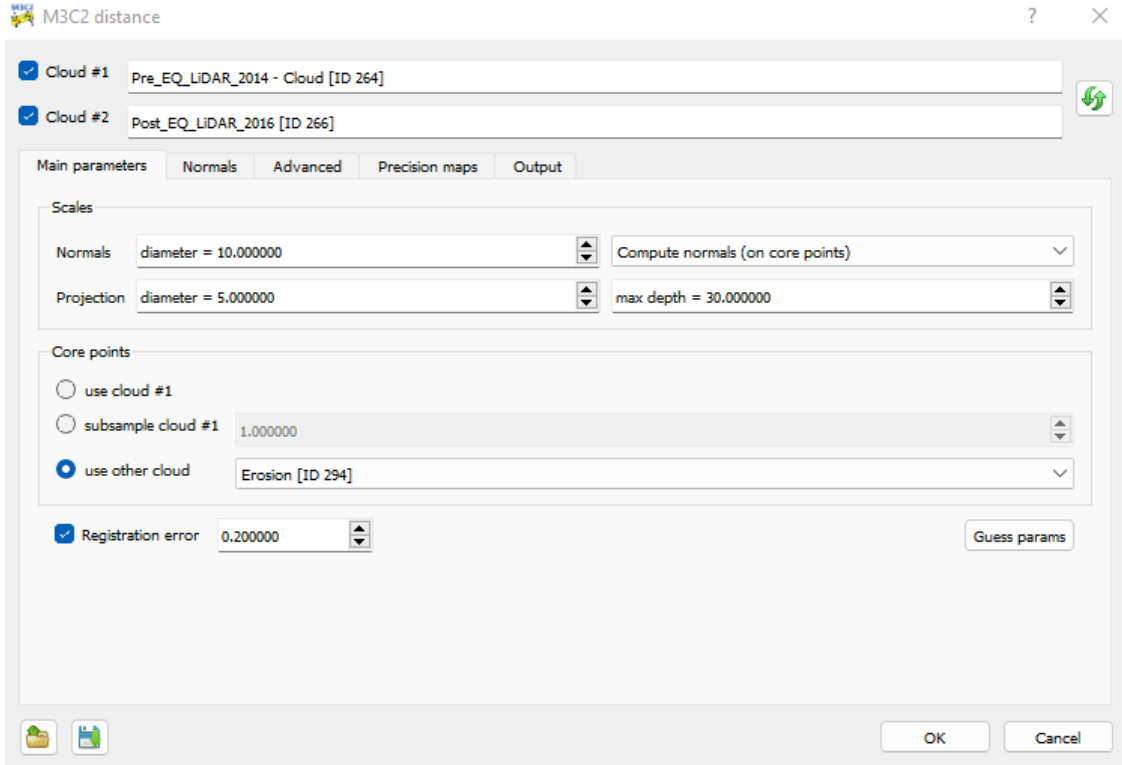
4.3 Segmentation into individual cluster

4.3.1 Extract significant negative and positive changes


Select the significant changes point cloud and active the ‘M3C2 distance’ scalar field. Then select the filtering tool , change the max value to zero and then click on ‘split’.

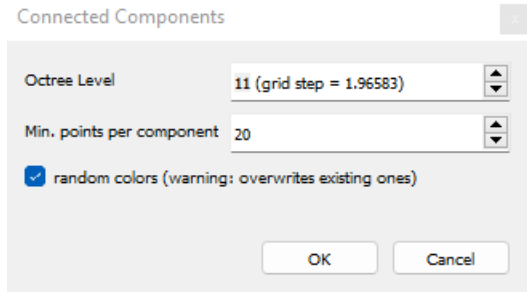
4.3.2 Vertical-M3C2

1. Rename all the scalar fields resulting from the 3D-M3C2 calculation (for example you can add ‘_3D’ at the end)
2. Select the Pre-EQ and Post-EQ point clouds and click on the M3C2 plugin icon . The core points are the point clouds with erosion areas and deposits areas. Do not forget to select the ‘vertical mode’ to compute normals.
3. Rename the newly created ‘M3C2 distance’ SF ‘M3C2_distance_vertical’.



4.3.3 Segmentation by connected component

1. Select the point cloud with negative M3C2 distance values and click on the Label connected components icon .



A_{min} was set to 20 m² to be consistent with the area of the projection cylinder used to average the point cloud position in the M3C2 distance calculation, $\pi * (d/2)^2 = 19.6$ m² with $d = 5$ m.

2. Repeat the previous step for positive M3C2 distance values.



Let's see the effect of the octree level (or minimum distance between two clusters) on the results! You can try with an octree level of 10 ($D_m \sim 4$ m) and of 9 ($D_m \sim 8$ m).

4.4 Volume estimation (using python/Matlab or excel)



1. Select all the segmented landslides
2. Tools > Batch export > Export cloud info or E

All information about the different landslides will be stored in a .csv file that can then be analyzed using python/Matlab or excel.

4.5 Export result to a GIS format

1. Select all the landslide source or deposit point clouds, clone  and Merge . While merging click on 'Yes' to generate a scalar field with the original cloud index.

Then we want to create a raster or a regular grid as we did to create core points. However, we want the points of the grid to be exactly in the same position than the core points.

2. Select the post-EQ_LiDAR point cloud (or the point cloud from which you created the core points) and select . Then click on 'Edit grid' and press 'OK', update the grid and click on 'OK'.
3. Select the point clouds where all the landslide sources (or deposits) are merged and select .
4. Go to 'Edit grid' and click on 'Last'. For empty cells 'leave empty' and update grid.

5. Click on raster and you can export your result in a .tif file format.

Now you can open this file in your favorite GIS software.

It is also possible to export the scalar fields!

4.6 Treatment of false detection (using python/Matlab or excel)

Owing to the simplified formulation of the $LoD_{95\%}$, it is possible for spatially correlated errors to create patches of statistically significant change that would appear after segmentation as false landslide detection.

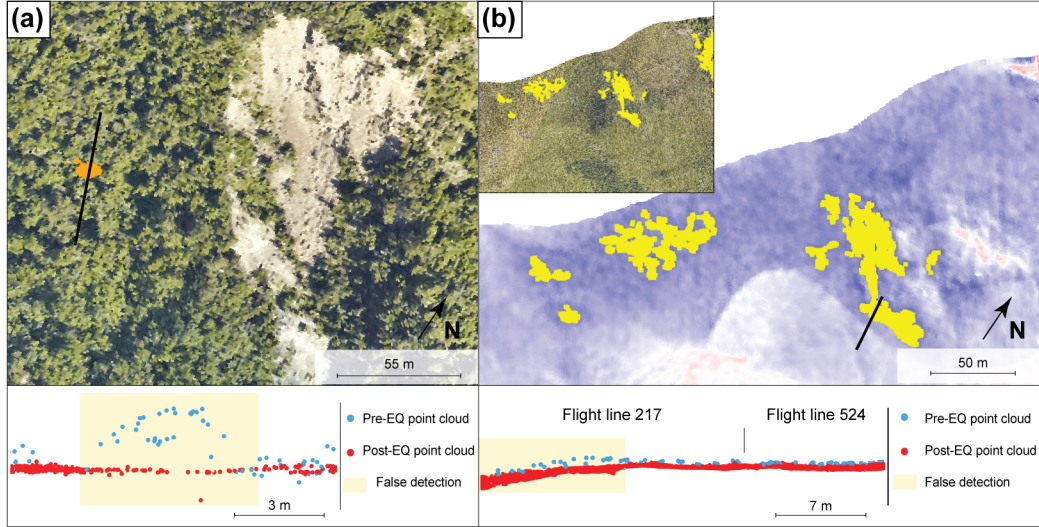


Illustration of two types of labeled false detections. (a) A false detection located in a forest due to vegetation incorrectly classified as ground in the pre-EQ point cloud. The apparent negative topographic change creates a source. (b) A false detection due to pre-EQ intra-survey registration errors. Yellow points on the post-EQ orthoimagery (Aerial Surveys, 2017) and the M3C2 distance field indicate patches of significant change that are a false detection. They occur due to a complex combination of intra-line errors related to time-dependent attitude and position errors and intra-survey flight-line registration error.

As false detections mainly emerge from the errors in the data in relation to the amplitude of a real topographic change that we aim to capture, we first choose to analyze three metrics based on the 3D-M3C2 calculation:

- (1) the maximum 3Ddistance,
- (2) the mean $LoD_{95\%}$ and
- (3) the mean signal-to-noise ratio (SNR).

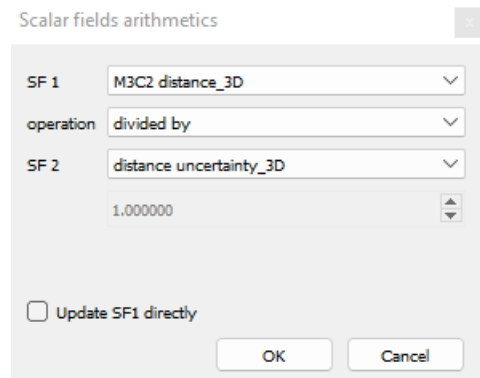
As the $LoD_{95\%}$ is a direct measure of the quality of the data (point density and roughness), classification errors of vegetation should be characterized by a significantly higher mean $LoD_{95\%}$ than actual landslide sources. The SNR is defined as the ratio between the 3D-M3C2 distance and the associated $LoD_{95\%}$ for each core point. This measure can be used as a confidence metric for each source.

$$SNR = \frac{3D \text{ M3C2 distance}}{LoD_{95\%}}$$

This ratio is defined for each core point and the mean is used at the scale of each landslide to filter artefacts.

In practice:

1. Select the point cloud with negative M3C2 distance values and go to Edit > Scalar fields > Arithmetic
2. Compute the SNR ratio



3. Rename the newly created scalar field '*SNR_ratio*'

We also choose to take advantage of the ability of the 3D differencing approach to detect deposit areas to analyze the closest deposit distance (CDD). The CDD is defined for each provisional source as the closest downslope distance to a provisional deposit along the flow path using a D8 (deterministic eight-node) algorithm (Fairfield and Leymarie, 1991). This distance is calculated from the post-EQ DEM with the MATLAB-based TopoToolbox software (Schwanghart and Scherler, 2014).

5 A bit of Python

These are all the libraries you will need.

These are libraries you probably already have.

- numpy
- matplotlib
- os
- subprocess

These are those you probably have to install :

- pandas : a fast, powerful, flexible and easy to use open source data analysis and manipulation too
Installation with pip: '*pip install pandas*' in your command prompt
- laspy : Python library for lidar LAS/LAZ
Installation with pip: '*pip install laspy*' in your command prompt
- scikit-learn : Simple and efficient tools for predictive data analysis
Installation : '*pip install -U scikit-learn*' in your command prompt

5.1 Filtering of false detection using the SNR ratio

As a reminder, the SNR is defined as the ratio between the 3D-M3C2 distance and the associated $LoD_{95\%}$ for each core point. This measure can be used as a confidence metric for each source.

```
[5]: # Definition of the functions
def do_compression(filename):
    """
    This function allows to compress a ".las" file to a ".laz" file.
    filename : path + filename
    """
    query="laszip -i "+filename+" -olaz"
    subprocess.run(query)
    os.remove(filename)

def write_new_SF(path,filename,SF_name,SF_array):
    """
    This function allows to write a new scalar field from a LAS file
    SF_name: Name of the new scalar field
    SF_array : Values of this new scalar field
    """
    import copy
    import os
    # rename the input file
    os.rename(path+filename+'.laz', path+filename+'_origin+'.laz')
    # Set up our input and output files.
    inFile = laspy.file.File(path+filename+'_origin.laz', mode = "r")
    outFile = laspy.file.File(path+filename+'.las', mode = "w",
        header = inFile.header)
    # Define our new dimension. Note, this must be done before giving
    # the output file point records.
    outFile.define_new_dimension(name = SF_name,
        data_type = 9, description = "Test Dimension")

    # Lets go ahead and copy all the existing data from inFile:
    for dimension in inFile.point_format:
        dat = inFile.reader.get_dimension(dimension.name)
        outFile.writer.set_dimension(dimension.name, dat)

    # Now lets put data in our new dimension
    # (though we could have done this first)

    # Note that the data type 5 refers to a long integer
    outFile.points['point'][SF_name] = SF_array
    inFile.close()
    outFile.close()
    do_compression(path+filename+'.las')
```

```
os.remove(path+filename+'_origin'+'.laz')
```

```
[6]: # Import libraries
import laspy
from laspy.file import File
import numpy as np
import subprocess
import os
```

```
[3]: # Read the .laz file and the scalar field 'id' corresponding to the id of each
↳ landslide sources
path = 'D:/Post_doc/Summer_school_ptcloud_2022/3D_landslide_detection_python/
↳ Landslide_detection/res/'
filename = 'landslide_sources'

inFile = File(path+filename+'.laz', mode='r')
id_landslide = inFile.points['point']['id']
id_landslide_unique = np.unique(id_landslide) # create an array with all the
↳ landslide id
SNR_ratio = inFile.points['point']['SNR_ratio']
inFile.close()

SNR_threshold = 1.45 # the value of the mean SNR ratio below which the
↳ landslide sources are considered as false detection.
id_SNR = np.copy(SNR_ratio)

for i in id_landslide_unique:
    SNR_ratio_ls = SNR_ratio[id_landslide == i] # select the SNR values for the
↳ landslide source with the id i
    mean_SNR = np.mean(SNR_ratio_ls) # Compute the mean
    if mean_SNR < 1.45 :
        id_SNR[id_landslide==i] = -9999 # we define -9999 as a new id for the
↳ false detection to be easily removed later in CloudCompare
    else:
        id_SNR[id_landslide==i] = i # we keep the same id otherwise

write_new_SF(path,filename,'id_landslide',id_SNR) # write a new scalar_field
```

Now you can open the newly created .laz file with Cloudcompare and see which landslide sources have been filtered out!

You can split the point cloud in function of the 'id' scalar field : *Edit > Scalar fields > Split cloud (integer values)*

5.2 Landslide analysis

5.2.1 Read the .csv file with pandas

```
[7]: # Import libraries
import pandas as pd

[8]: # import data into a pandas dataframe
path = 'D:/Post_doc/Summer_school_ptcloud_2022/3D_landslide_detection_python/
↳Landslide_detection/res/'
file1 = 'Landslide_source_infos'
file2 = 'Landslide_deposits_infos'
# Open the dataset into a pandas dataframe
df_source= pd.read_csv(path+file1+'.csv',sep=';')
df_deposits= pd.read_csv(path+file2+'.csv',sep=';')
```

5.2.2 Display some landslide general information

```
[9]: def read_landslide_prop(path, filenames):
    """
    This function allows to read and display some basic landslide properties.
    path: path to the landslide results
    filenames: dictionary with the sources and deposits filenames.
    """
    # Read .csv files into a dataframe
    dataframes={'Sources': pd.read_csv(path+filenames['Sources'],sep=';
↳'), 'Deposits': pd.read_csv(path+filenames['Deposits'],sep=';')}
    # Create empty list to add values after
    number_of_clouds = []
    area_range = []
    Total_area = []
    volume_range = []
    Total_volume = []
    uncertainty_volume_range = []
    Uncertainty_Total_volume = []
    # Select Area, volume and uncertainty
    ## define the name of the scalar fields corresponding to the vertical-M3C2_
↳calculation
    ## and the 3D distance uncertainty
    vertical_M3C2 = 'Sum_M3C2_distance_vertical'
    dist_unc = 'Sum_3D_distance_uncertainty'
    for i in range(0,len(dataframes)):
        number_of_clouds.append(len(list(dataframes.values())[i]))
        area_range.append([list(dataframes.values())[i]['Points'].
↳min(),list(dataframes.values())[i]['Points'].max()])
        Total_area.append(list(dataframes.values())[i]['Points'].sum())
```

```

        volume_range.append([np.around(list(dataframes.
↪values())[i][vertical_M3C2].abs().min(),2),np.around(list(dataframes.
↪values())[i][vertical_M3C2].abs().max(),2)])
        Total_volume.append(np.around(list(dataframes.values())[i][dist_unc].
↪abs().sum(),0))
        uncertainty_volume_range.append([list(dataframes.values())[i][dist_unc].
↪min(),list(dataframes.values())[i][dist_unc].max()])
        Uncertainty_Total_volume.append(np.around(list(dataframes.
↪values())[i][dist_unc].sum(),0))

# Range data into a dataframe
        df_dico={'Number of sources/deposits':number_of_clouds,'Area range (m2 ;
↪[min,max]':area_range,'Total area (m2)':Total_area,
        'Volume range (m3)':volume_range,'Total volume (m3)':
↪Total_volume, '3D uncertainty on total volume':Uncertainty_Total_volume}
        Dataframe = pd.DataFrame(df_dico,index=['Sources','Deposits'])

    return Dataframe

```

```

[10]: path = 'D:/Post_doc/Summer_school_ptcloud_2022/3D_landslide_detection_python/
↪Landslide_detection/res/'
filenames = {'Sources':'Landslide_source_infos.csv','Deposits':
↪'Landslide_deposits_infos.csv'}
read_landslide_prop(path, filenames)

```

```

[10]:
      Number of sources/deposits Area range (m2 ; [min,max]) \
Sources                1163                [20.0, 40444.0]
Deposits                741                [20.0, 27894.0]

      Total area (m2) Volume range (m3) Total volume (m3) \
Sources          320589.0    [1.01, 169710.32]          182065.0
Deposits          315780.0    [0.03, 151612.67]          174306.0

      3D uncertainty on total volume
Sources                182065.0
Deposits                174306.0

```

5.2.3 Landslide source distribution

```

[17]: def distribution_log_bin(dataset, N_bin = 100, density = True):
      '''
      Compute dataset distribution with a logarithmic binning
      '''
      dataset = dataset[~np.isnan(dataset)]

      #binning creation

```



```

max_data = np.max(dataset)
min_data = np.min(dataset)
bins = np.logspace(np.log10(min_data), np.log10(max_data), num=N_bin) #
↳ create bin in a logspace

#distribution
bin_min = bins[:-1]
bin_max = bins[1:]
N = np.zeros(len(bin_min))
N2 = np.zeros(len(bin_min))
X = np.zeros(len(bin_min))

for i in range(len(bin_min)):
    index = [(dataset>=bin_min[i]) & (dataset<bin_max[i])]
    if i == len(bin_min):
        index = [(dataset>=bin_min[i]) & (dataset<=bin_max[i])]
    N[i] = dataset[index].shape[0]
    X[i] = (bin_min[i] + bin_max[i])/2 # take the center of the bin

if density == True:
    N = N/(bin_max-bin_min)
    N = N/dataset.shape[0]
    return X, N
else:
    return X, N

def linear_fit(X,Y):
    '''
    Function that compute a log-transformed linear fit model
    '''

    x_test = X==0
    x_test = np.max(x_test)
    y_test = Y==0
    y_test = np.max(y_test)
    if x_test == True:
        Y = Y[X!=0]
        X = X[X!=0]
    if y_test == True:
        X = X[Y!=0]
        Y = Y[Y!=0]
    p, v = np.polyfit(np.log10(X), np.log10(Y), deg=1, cov=True)
    std = np.sqrt(np.diag(v))
    print(p, std)

    ypred = 10**(p[1])*X**p[0]

```

```

from sklearn.metrics import r2_score
print('R2 = ', r2_score(np.log10(Y),np.log10(ypred)))

return p, v, std, ypred, X,Y

```

```
[36]: import matplotlib.pyplot as plt
```

```
[41]: # Compute distribution
dataset = df_source['Points'].to_numpy() # here we consider that the number of
↳point is approximately the area, you can choose the landslide volume as well
X, N = distribution_log_bin(dataset, N_bin = 20,density = True)
# Compute linear fit
p,v,std, y_fit, X_fit,_ = linear_fit(X,N)

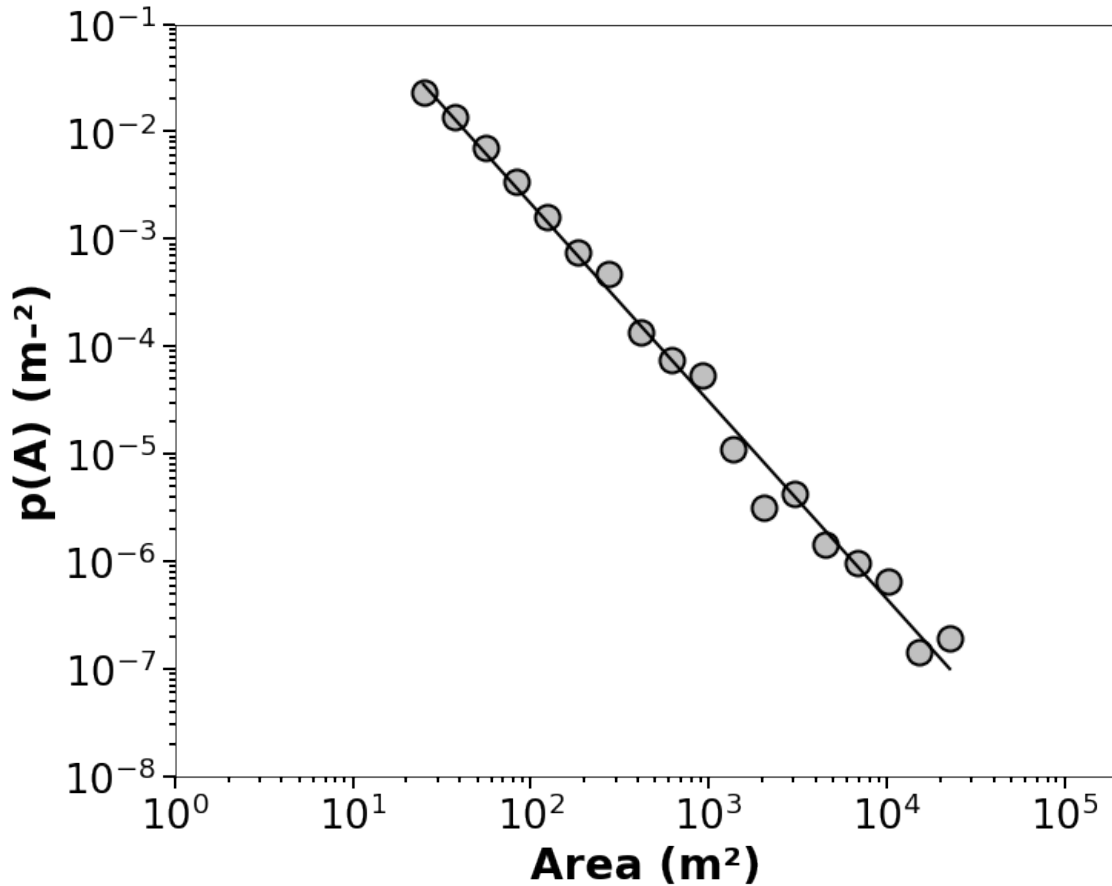
# Plot results
ymin, ymax = 1e-8, 1e-1
xmin, xmax = 1, 2e5
plt.figure(figsize=(11,9))
# plot the data
plt.plot(X,N,'o', markersize = 16, mfc='silver', mec = 'k',mew=2, label = '3D
↳inventory',zorder=4)
plt.plot(X_fit,y_fit,'k-',zorder=4,linewidth=2)
# plot parameters
plt.tick_params(axis='both',which='major',length=8,width=2)
plt.tick_params(axis='both',which='minor',length=4,width=2)
plt.ylim(ymin,ymax)
plt.xlim(xmin,xmax)
plt.xscale('log')
plt.yscale('log')
plt.xticks(fontsize=26)
plt.yticks(fontsize=26)
plt.xlabel('Area (m2)', fontsize=28, fontweight='bold')
plt.ylabel('p(A) (m-2)', fontsize=28, fontweight='bold')

```

```
[-1.83883188  1.00709287] [0.04318124  0.13014635]
R2 =  0.9912539680932435
```

C:\Users\tbernard\Anaconda3-64bit\envs\papier_landslide\lib\site-packages\ipykernel_launcher.py:24: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.

```
[41]: Text(0, 0.5, 'p(A) (m-2)')
```



6 References

- Bernard, T.G., Lague, D., Steer, P. (2021) Beyond 2D landslide inventories and their rollover: synoptic 3D inventories and volume from repeat LiDAR data. *Earth Surface Dynamics*. 9(4), 1013-1044.
- Borradaile, G. J.: *Statistics of earth science data: their distribution in space, time, and orientation*, Springer-Verlag, New York, 2003
- Fairfield, J. and Leymarie, P.: Drainage networks from grid digital elevation models, *Water Resour. Res.*, 27, 709–717, <https://doi.org/10.1029/90WR02658>, 1991.
- Schwanghart, W. and Scherler, D.: Short Communication: Topo-Toolbox 2 – MATLAB-based software for topographic analysis and modeling in Earth surface sciences, *Earth Surf. Dynam.*, 2, 1–7, <https://doi.org/10.5194/esurf-2-1-2014>, 2014.

[]: