VGC Short Course '4D change analysis of near-continuous LiDAR time series for applications in geomorphic monitoring'

Getting started with 4D change analysis in py4dgeo

See the full workflow in https://github.com/tum-rsa/vgc2023-shortcourse-4d/blob/main/course/practical/vgc2023_time_series_change_analysis.html

Katharina Anders
k.anders@tum.de

Technical University of Munich
TUM School of Engineering and Design
Professorship for Remote Sensing Applications

20 September 2023

The case for py4dgeo

4D → massive multitemporal point cloud data
- Automating change analysis
- Bundling state-of-the-art methods

Open-source Python library for change analysis in 4D point cloud data: https://github.com/3dgeo-heidelberg/py4dgeo

… providing 3D/4D methods:
- M3C2 (Lague et al., 2013)
- 4D objects-by-change (Anders et al., 2021)
- Correspondence-Driven Plane-Based M3C2 (Zahs et al., 2022)
- M3C2-EP (Winiwarter et al., 2021)
- …

… and a flexible interface for analysis workflows

See the basic usage tutorials: https://py4dgeo.readthedocs.io and materials in the E-TRAINEE online course

py4dgeo

Open source Python library for geographic change analysis in 4D point cloud data

Creating epochs from point cloud arrays:

```
print('[%s] Creating epoch 1...' % (datetime.now().strftime('%Y-%m-%d %H:%
epoch1 = py4dgeo.Epoch(coords1)

print('[%s] Creating epoch 2...' % (datetime.now().strftime('%Y-%m-%d %H:%
epoch2 = py4dgeo.Epoch(coords2)

print('[%s] Epochs successfully created, kd-trees built' % (datetime.now()
```

```
[2021-12-23 10:41:54] Creating epoch 1...
[2021-12-23 10:41:55] Creating epoch 2...
[2021-12-23 10:41:56] Epochs successfully created, kd-trees built
```

M3C2 distance calculation

Configuring and running M3C2:

```
print('[%s] Configuring M3C2...' % (datetime.now().strftime('%Y-%m-%d %H:%
m3c2 = py4dgeo.M3C2(epochs=(epoch1, epoch2), corepoints=corepoints, radii=
m3c2_multiscale = py4dgeo.M3C2(epochs=(epoch1, epoch2), corepoints=corepoi

print('[%s] Running M3C2...' % (datetime.now().strftime('%Y-%m-%d %H:%M:%S
distances, uncertainties = m3c2.run()
distances_multiscale, uncertainties_multiscale = m3c2_multiscale.run()
print('[%s] Calculation successful' % (datetime.now().strftime('%Y-%m-%d %
```

```
[2021-12-23 10:41:56] Configuring M3C2...
[2021-12-23 10:41:56] Running M3C2...
[2021-12-23 10:41:58] Calculation successful
```

Accessing all result information, i.e. `lodetection`, `num_samples`, and `stddev` in `uncert`

```
lodetection = uncertainties['lodetection']
stddev1 = uncertainties['stddev1']
numsamples1 = uncertainties['num_samples1']
stddev2 = uncertainties['stddev2']
numsamples2 = uncertainties['num_samples2']
normals = m3c2.directions()
```

Change Analysis in py4dgeo

class **Epoch**

- Data: coordinates (nx3 array)
- Property: metadata (dictionary), transformation
- Methods:
  - calculate_normals()
  - transform()
  - save/load()
- Created using py4dgeo.read_from_las() / py4dgeo.read_from_xyz()

```
# import the library for M3C2 distance calculation
import py4dgeo

# load point clouds as epoch objects
epoch_2009, epoch_2017 = py4dgeo.read_from_las(
    las_data2009_aligned, las_data2017
)
```

https://py4dgeo.readthedocs.io/en/latest/pythonapi.html#py4dgeo.Epoch

Bitemporal Change Analysis

… using the M3C2 algorithm (Lague et al., 2013)

Parameters:

- Normal radius (can be multiscale)
- Cylinder radius (‚projection scale‘ d/2)
- Max. search depth
- Registration error (for consideration of distance uncertainty)

Obtained variables:

- Distance
- Normal vector (direction)
- Num. samples n
- Spread σ
- Level of detection

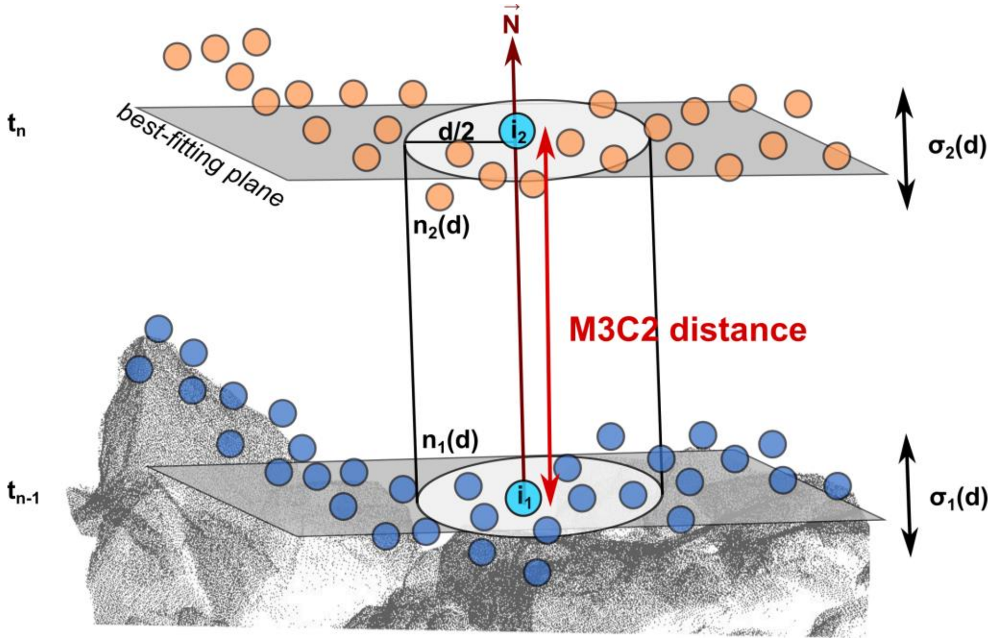


Figure by V. Zahs modified from Zahs et al. (2022)

Bitemporal Change Analysis

… using the M3C2 algorithm (Lague et al., 2013)

- Executed on **core points**
- Parameters are passed on instantiation of the algorithm
- Running the algorithm returns
  - **Distances**
  - **Uncertainty** data → structured array with
    - lodetection
    - num_samples[1/2]
    - spread [1/2]

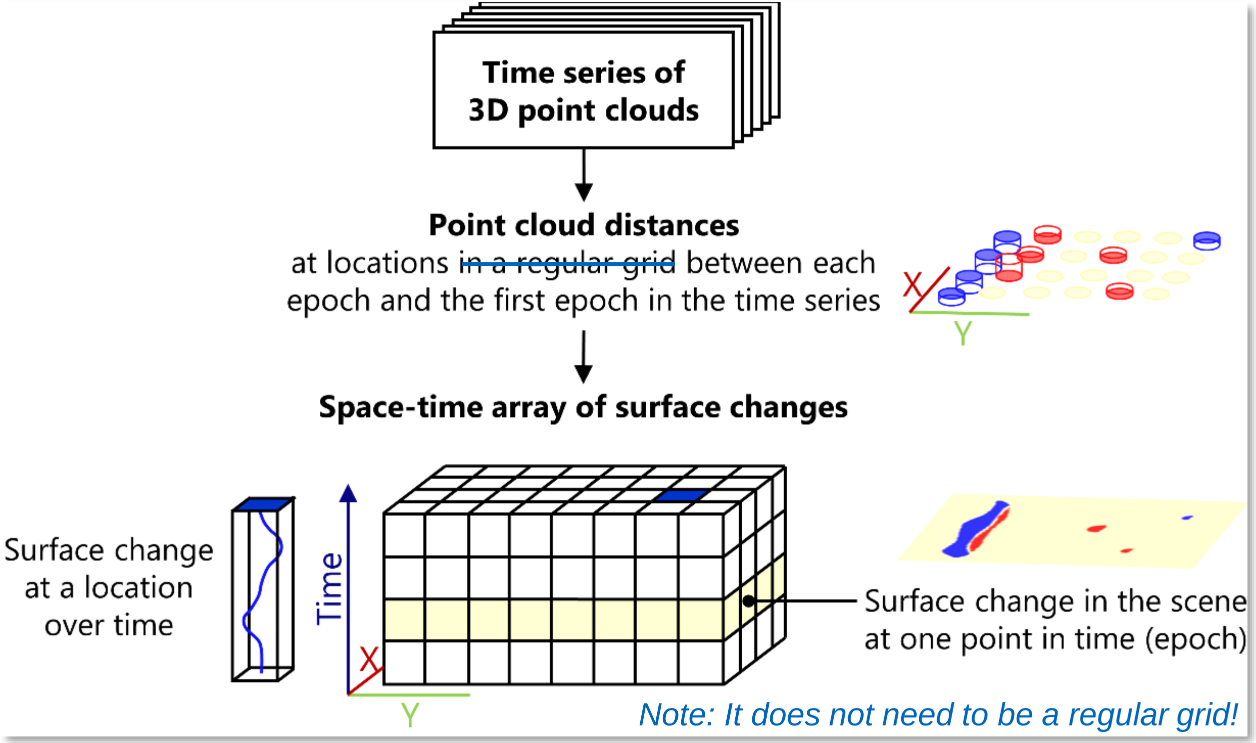
```python
# define a set of corepoints
corepoints = epoch_2009.cloud[::]
```

```python
m3c2 = py4dgeo.M3C2(
    epochs=(epoch_2009, epoch_2017),
    corepoints=corepoints,
    normal_radii=(2.0, 1.0, 8.0),
    cyl_radii=(2.0,),
    max_distance=(50.0),
    registration_error=(1.31)
)

m3c2_distances, uncertainties = m3c2.run()
```

Time Series Change Analysis in py4dgeo

class **SpatiotemporalAnalysis**

- **Data** (properties):
  - corepoints
  - reference_epoch
  - timedeltas
  - m3c2 → algorithm settings
  - **distances** (M2C2 result)
  - uncertainties (M2C2 result)
- Data is added via **add_epochs(*epochs)**



Note: It does not need to be a regular grid!

Anders et al. (2021)

https://py4dgeo.readthedocs.io/en/latest/pythonapi.html#py4dgeo.SpatiotemporalAnalysis

Creating a SpatiotemporalAnalysis (1/2)

```python
analysis = py4dgeo.SpatiotemporalAnalysis(f'{data_path}/kijkduin.zip', force=True)
```

```python
# specify the reference epoch
reference_epoch_file = os.path.join(pc_dir, pc_list[0])

# read the reference epoch and set the timestamp
reference_epoch = py4dgeo.read_from_las(reference_epoch_file)
reference_epoch.timestamp = timestamps[0]

# set the reference epoch in the spatiotemporal analysis object
analysis.reference_epoch = reference_epoch
```

```python
# Inherit from the M3C2 algorithm class to define a custom direction algorithm
class M3C2_Vertical(py4dgeo.M3C2):
    def directions(self):
        return np.array([0, 0, 1]) # vertical vector orientation

# specify corepoints, here all points of the reference epoch
analysis.corepoints = reference_epoch.cloud[::]

# specify M3C2 parameters for our custom algorithm class
analysis.m3c2 = M3C2_Vertical(cyl_radii=(1.0,), max_distance=10.0, registration_error = 0.019)
```

Creating a SpatiotemporalAnalysis (2/2)

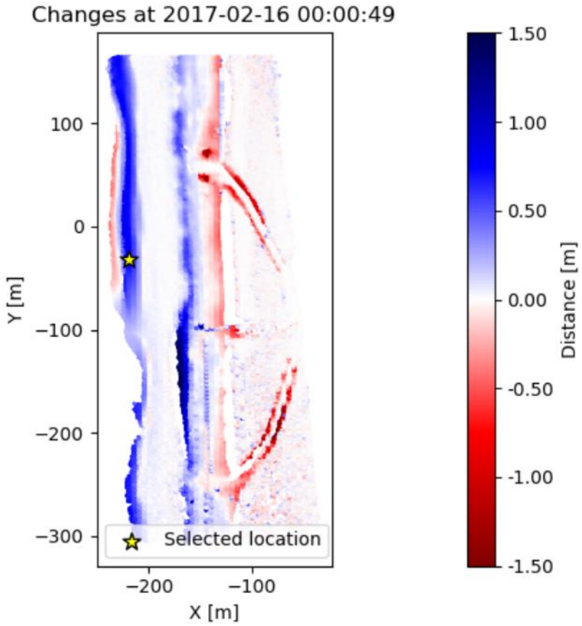
```
# create a list to collect epoch objects
epochs = []
for e, pc_file in enumerate(pc_list[1:]):
    epoch_file = os.path.join(pc_dir, pc_file)
    epoch = py4dgeo.read_from_las(epoch_file)
    epoch.timestamp = timestamps[e]
    epochs.append(epoch)

# add epoch objects to the spatiotemporal analysis object
analysis.add_epochs(*epochs)

# print the spatiotemporal analysis data for 3 corepoints and 5 epochs, respectively
print(f"Space-time distance array:\n{analysis.distances[:3,:5]}")
print(f"Uncertainties of M3C2 distance calculation:\n{analysis.uncertainties['lodetection'][:3, :5]}")
print(f"Timestamp deltas of analysis:\n{analysis.timedeltas[:5]}")
```

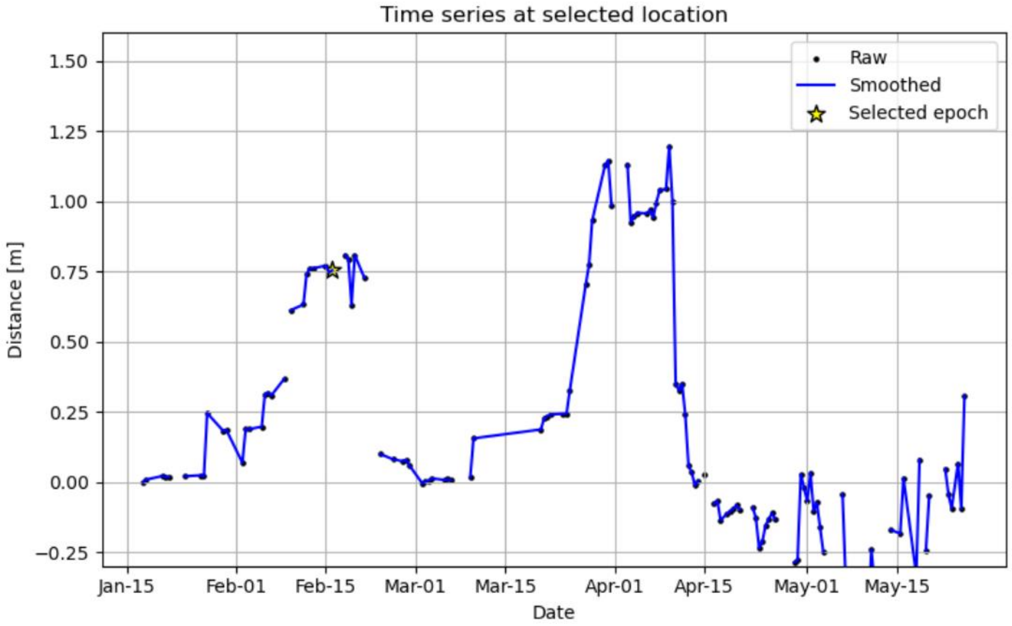
```
# get the corepoints
corepoints = analysis.corepoints.cloud
```

```
# get the list of timestamps from the reference epoch timestamp and timedeltas
timestamps = [t + analysis.reference_epoch.timestamp for t in analysis.timedeltas]
```

Making use of the SpatiotemporalAnalysis Object

```
cp_idx_sel = 15162 # selected core point index
epoch_idx_sel = 28 # selected epoch index
```

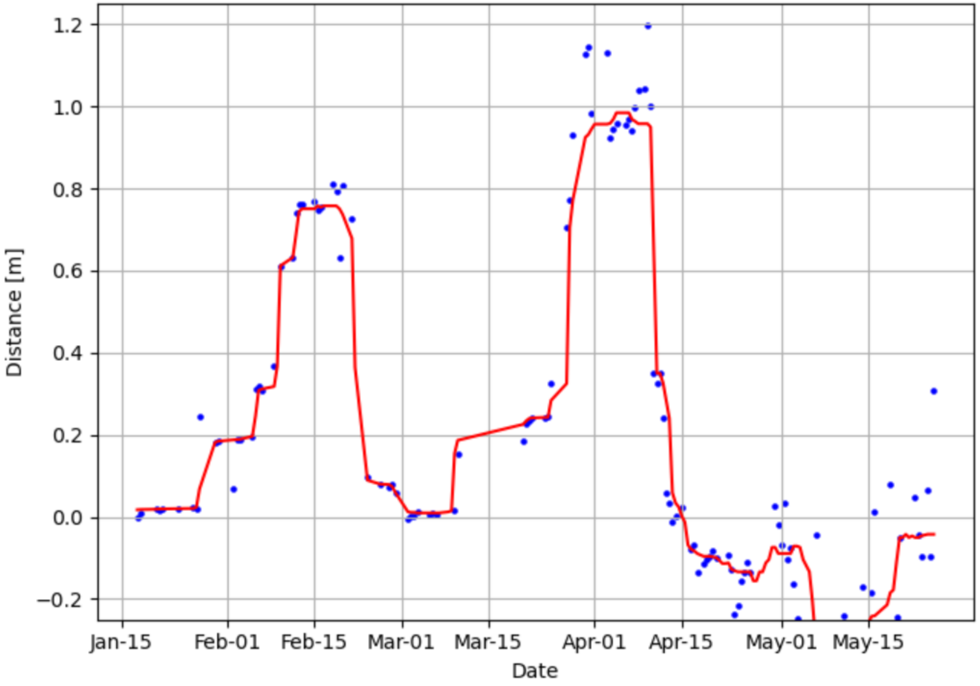




Temporal Smoothing

… following 4D filtering by Kromer et al. (2015)

```
analysis.smoothed_distances = py4dgeo.temporal_averaging(analysis.distances, smoothing_window=14)
```



References

- Anders, K., Winiwarter, L., Mara, H., Lindenbergh, R., Vos, S. E., & Höfle, B. (2021). Fully automatic spatiotemporal segmentation of 3D LiDAR time series for the extraction of natural surface changes. ISPRS Journal of Photogrammetry and Remote Sensing, 173, pp. 297-308. doi: 10.1016/j.isprsjprs.2021.01.015.
- Kromer, R., Abellán, A., Hutchinson, D., Lato, M., Edwards, T., & Jaboyedoff, M. (2015). A 4D Filtering and Calibration Technique for Small-Scale Point Cloud Change Detection with a Terrestrial Laser Scanner. Remote Sensing, 7 (10), pp. 13029-13052. doi: 10.3390/rs71013029.
- Kuschnerus, M., Lindenbergh, R., & Vos, S. (2021). Coastal change patterns from time series clustering of permanent laser scan data. Earth Surface Dynamics, 9 (1), pp. 89-103. doi: 10.5194/esurf-9-89-2021.
- Lague, D., Brodu, N., & Leroux, J. (2013). Accurate 3D comparison of complex topography with terrestrial laser scanner: Application to the Rangitikei canyon (N-Z). ISPRS Journal of Photogrammetry and Remote Sensing, 82, pp. 10-26. doi: 10.1016/j.isprsjprs.2013.04.009.
- Vos, S., Anders, K., Kuschnerus, M., Lindenberg, R., Höfle, B., Aarnikhof, S. & Vries, S. (2022). A high-resolution 4D terrestrial laser scan dataset of the Kijkduin beach-dune system, The Netherlands. Scientific Data, 9:191. doi: 10.1038/s41597-022-01291-9.
- Winiwarter, L., Anders, K., & Höfle, B. (2021). M3C2-EP: Pushing the limits of 3D topographic point cloud change detection by error propagation. ISPRS Journal of Photogrammetry and Remote Sensing, 178, pp. 240-258. doi: 10.1016/j.isprsjprs.2021.06.011.
- Zahs, V., Winiwarter, L., Anders,K. Williams, J. ,G., Rutzinger, M. & Höfle, B. (2022). Correspondence-driven plane-based M3C2 for lower uncertainty in 3D topographic change quantification. ISPRS Journal of Photogrammetry and Remote Sensing, 183, pp. 541 - 559. doi: 10.1016/j.isprsjprs.2021.11.018.