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

4D Objects-By-Change

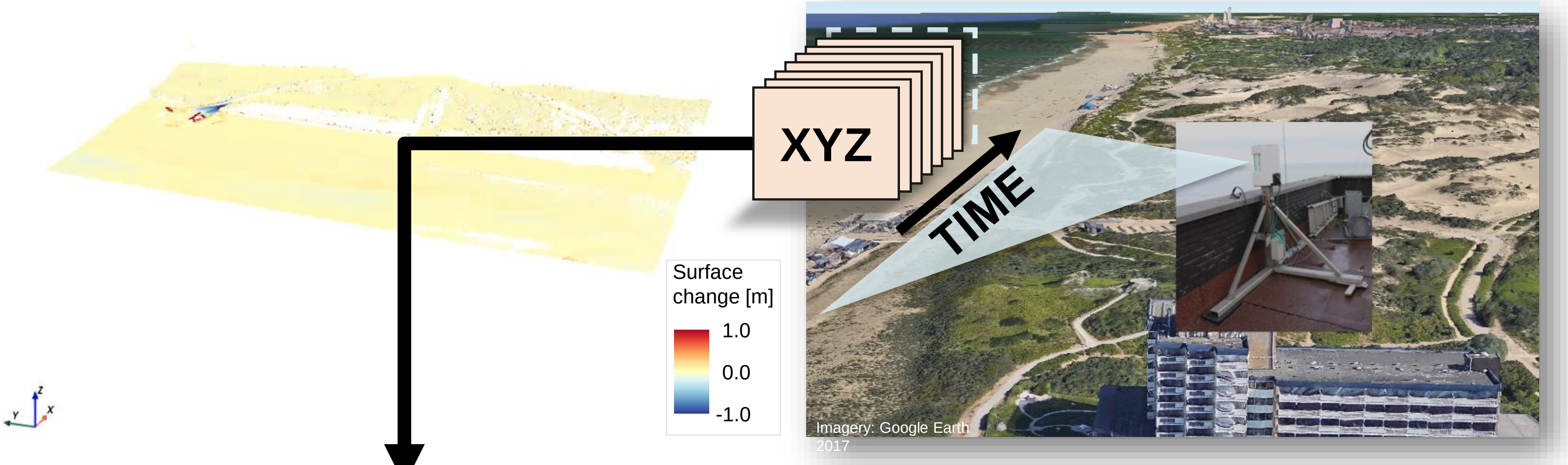
See the full workflow in https://github.com/tum-rsa/vgc2023-shortcourse-4d/blob/main/course/practical/vgc2023_time_series_change_analysis.html

Katharina Anders
k.anders@tum.de

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

20 September 2023

Near-Continuous Coastal Monitoring

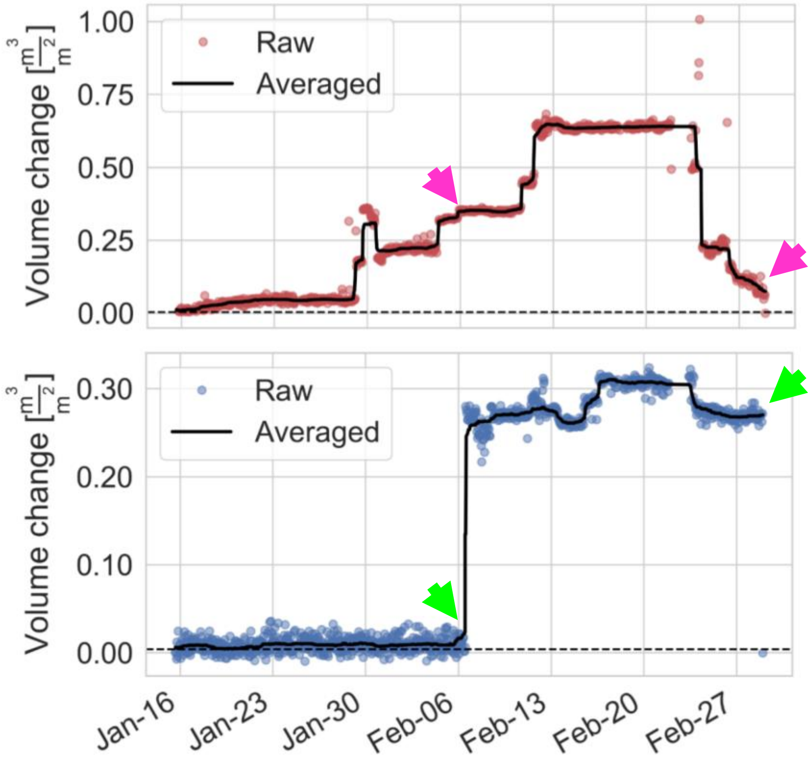


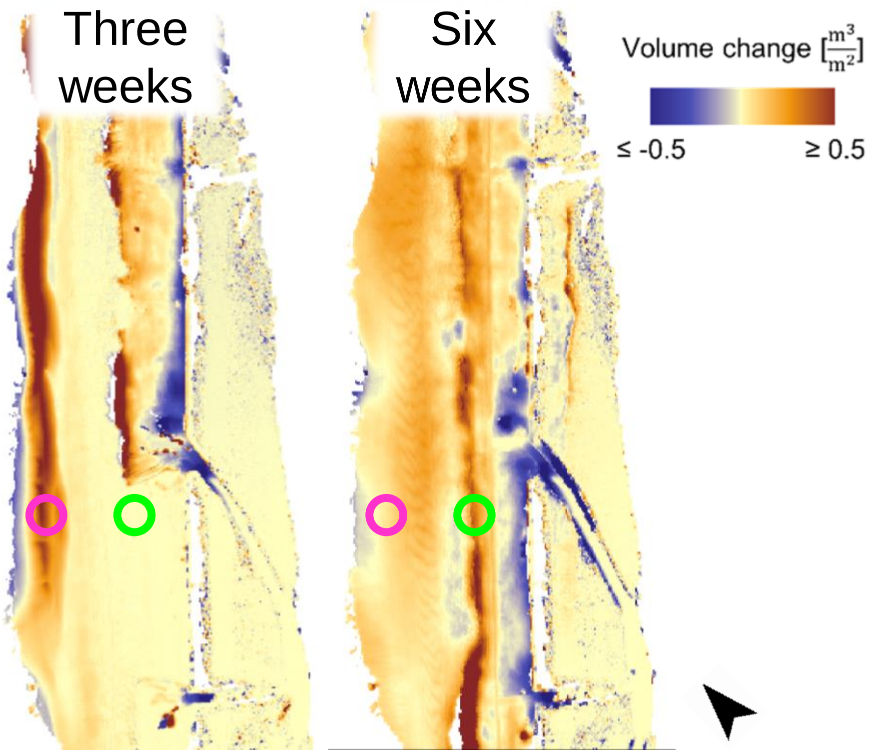
Surface activities

Hourly TLS from January to May 2017

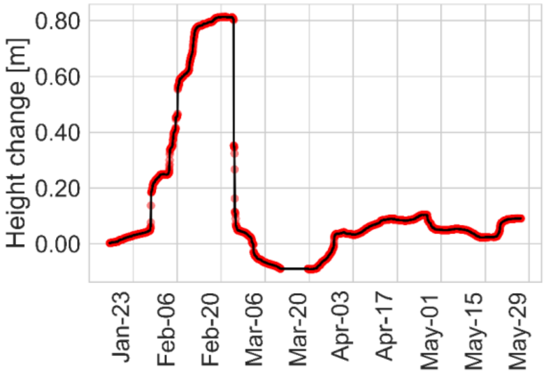
Anders et al. (2019)
Vos et al. (2022)

Time Series of Change





Time Series-Based Surface Change Analysis

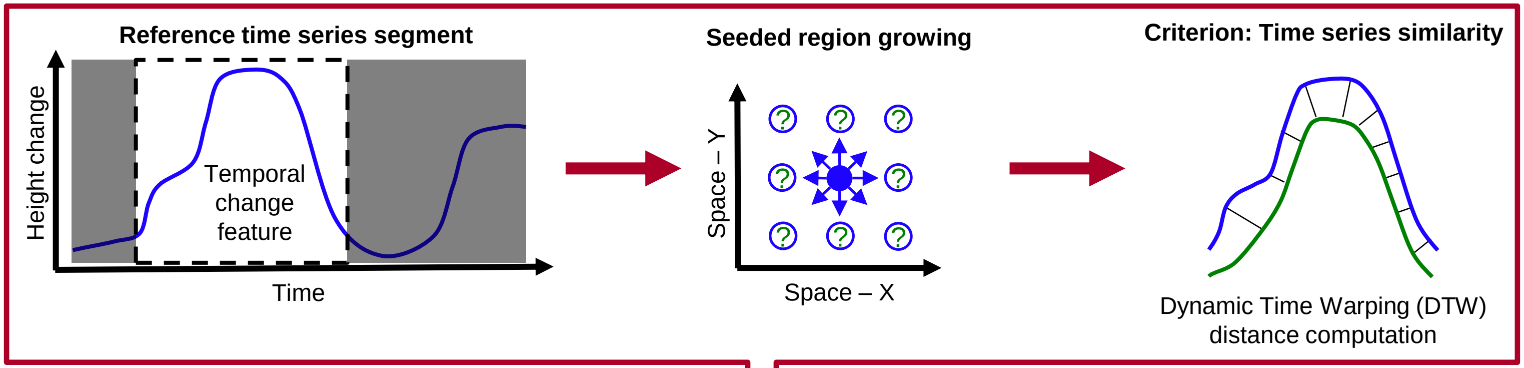


Spatiotemporal segmentation of surface changes for the extraction of ***4D Objects-by-change***

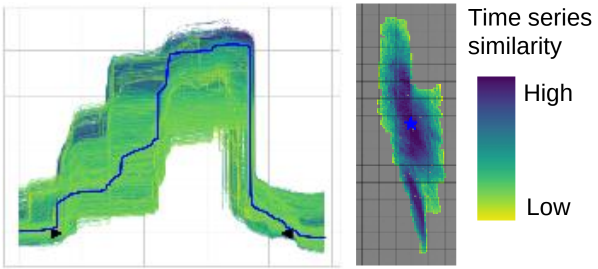
→**Object delineation as areas of similar surface change histories**

Anders et al. (2020)

Spatiotemporal Segmentation

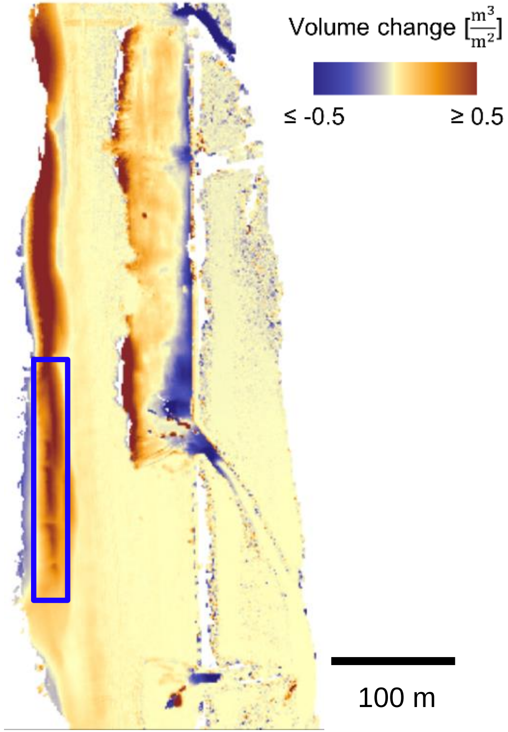


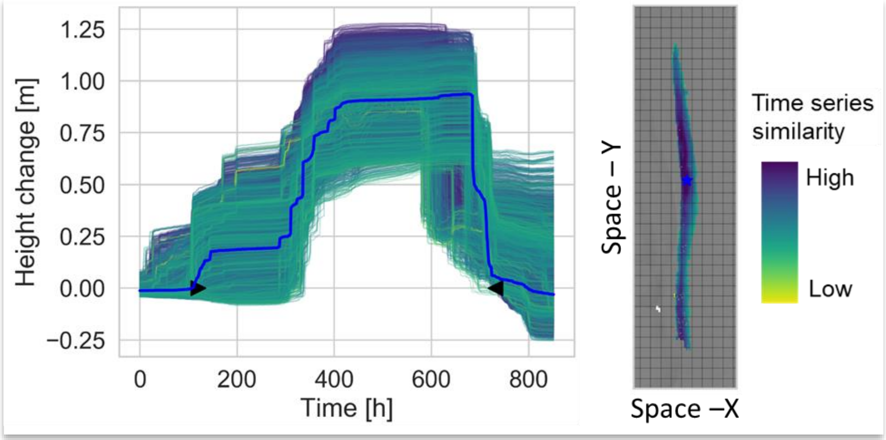
4D Object-By-Change



Anders et al. (2020)

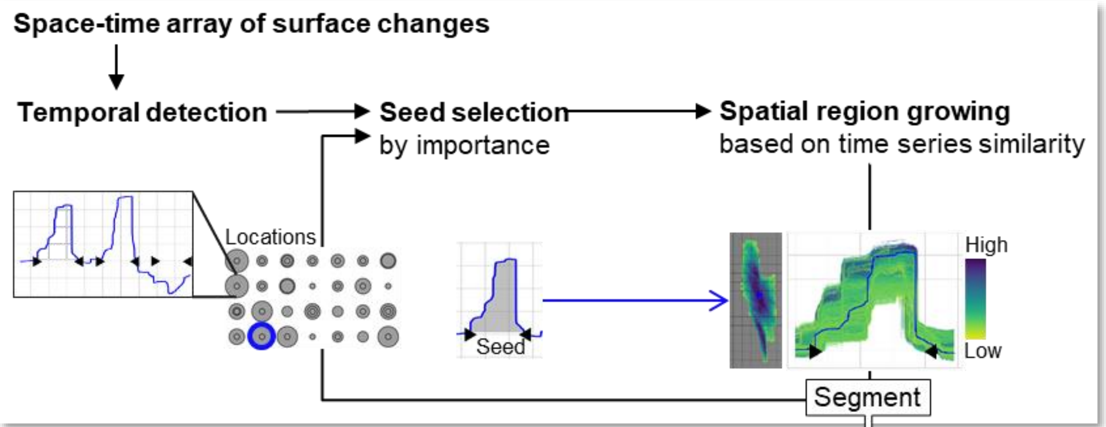
4D Object-By-Change



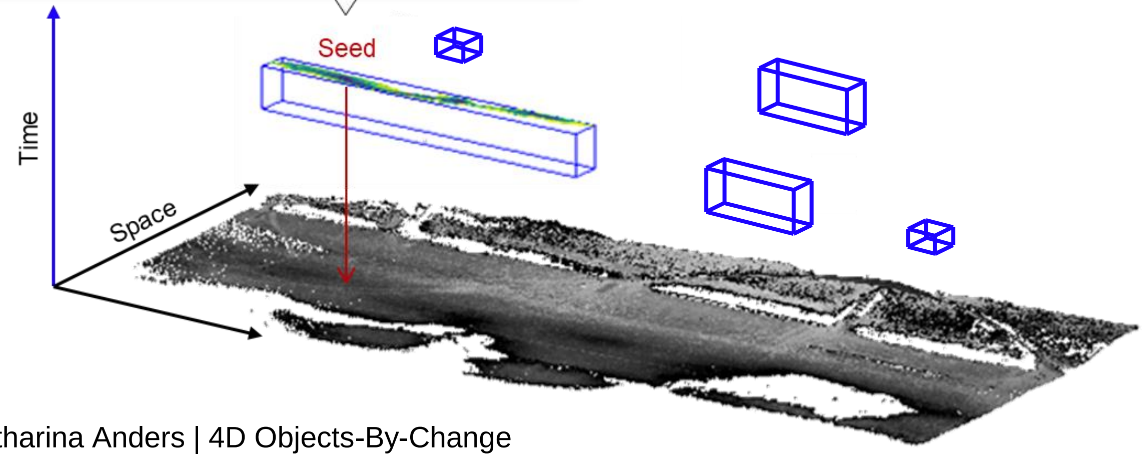


Anders et al. (2021)

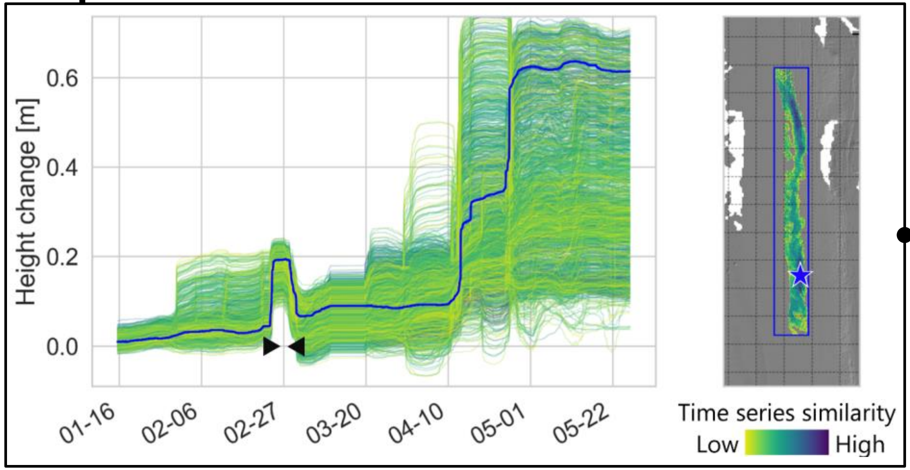
Fully Automatic Spatiotemporal Segmentation



Anders et al. (2021)

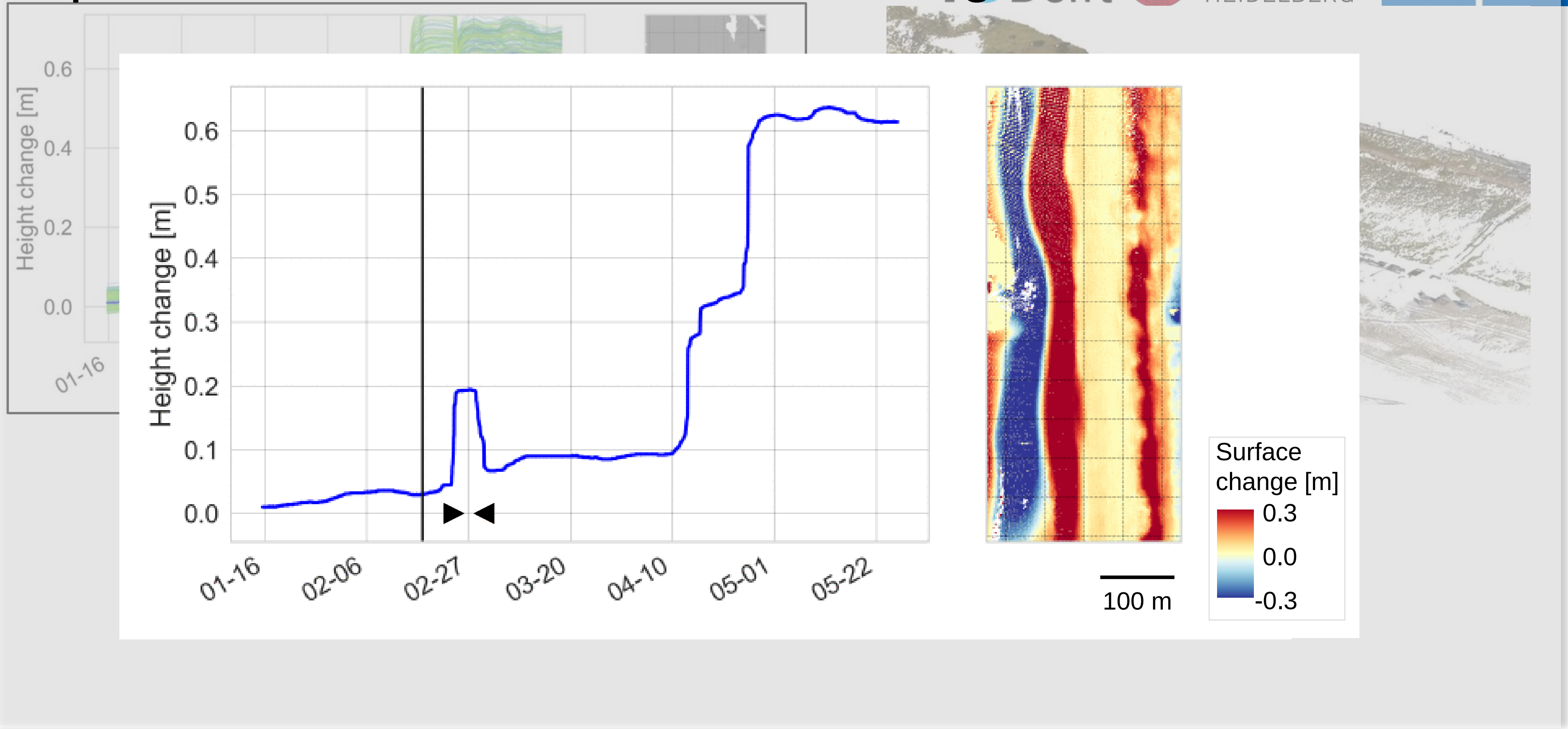


Improved Detection of Surface Changes



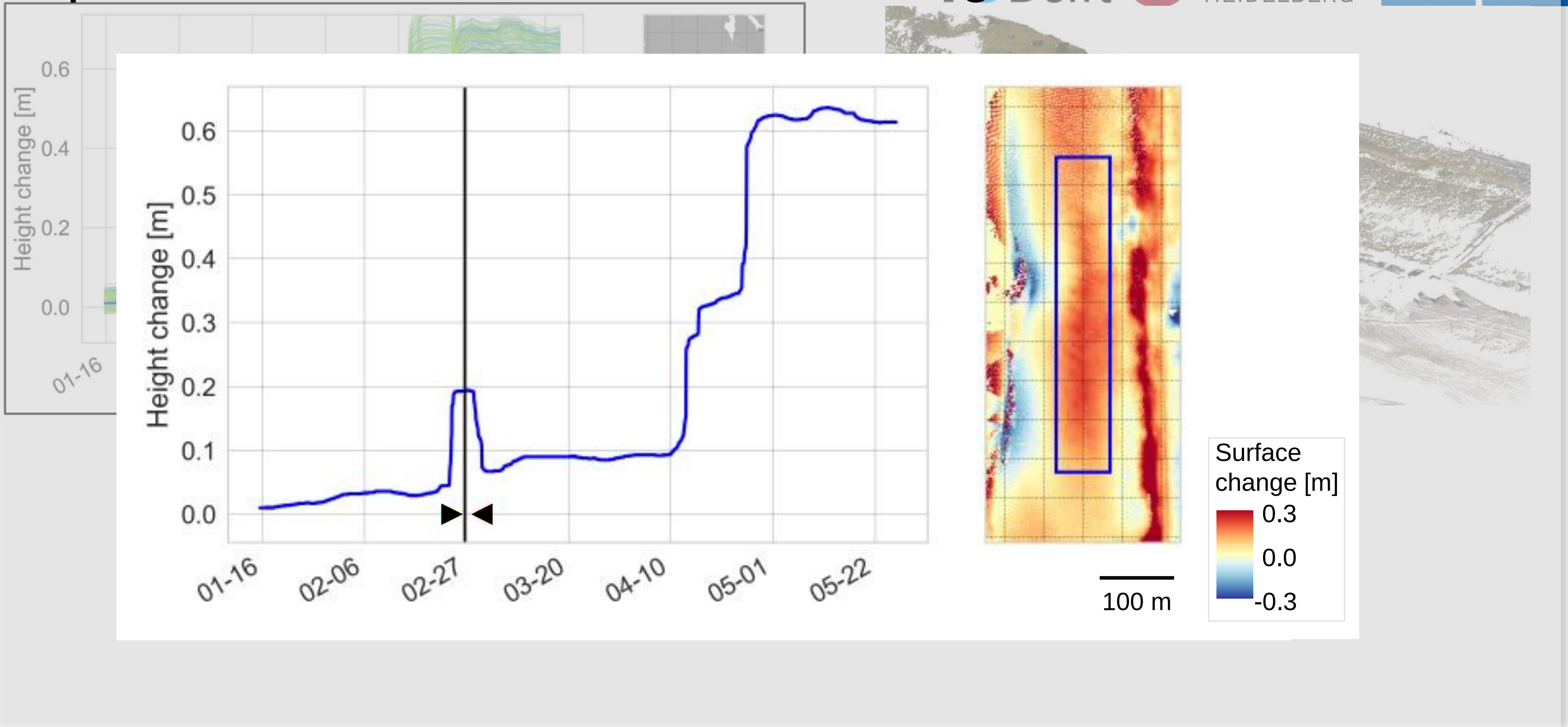
Anders et al. (2020)

Improved Detection of Surface Changes



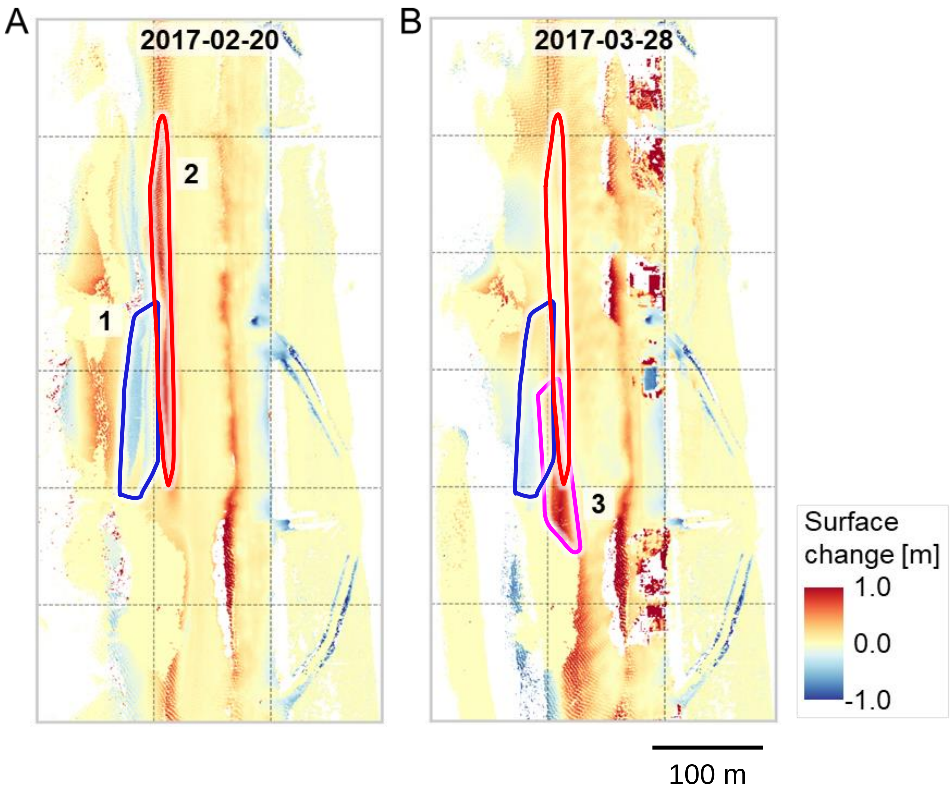
Anders et al. (2020)

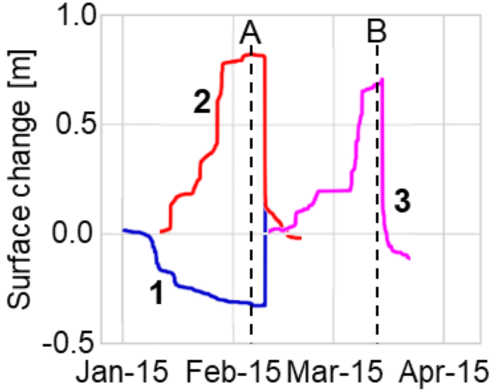
Improved Detection of Surface Changes



Anders et al. (2020)

Separating Overlapping Changes





Performance:

- 95 % completeness
- >80 % correctness

Anders et al. (2021)

From theory to practice: **py4dgeo**

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

```
# 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 =
```

```
analysis.smoothed_distances = py4dgeo.temporal_averaging(analysis.distances, smoothing_w
```

```
# parametrize the 4D-OBC extraction
algo = py4dgeo.RegionGrowingAlgorithm(window_width=14,
                                      minperiod=2,
                                      height_threshold=0.05,
                                      neighborhood_radius=1.0,
                                      min_segments=10,
                                      thresholds=[0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9],
                                      seed_candidates=[cp_idx_sel])

# run the algorithm
analysis.invalidate_results(seeds=True, objects=True, smoothed_distances=False) # only r
objects = algo.run(analysis)
```

From theory to practice: **py4dgeo**

```
ar                                                   f'{data_path}/kijkduin.zip', force=True)
```

```
#                                                    def
cl
                                                     cal

#                                                    he r
ar                                                   d[::

# specify M3C2 parameters for our custom algor
analysis.m3c2 = M3C2_Vertical(cyl_radii=(1.0,)
```

```
analysis.smoothed_distances = py4dgeo.temporal
```

```
# parametrize the 4D-OBC extraction
algo = py4dgeo.RegionGrowingAlgorithm(window_w
                                      minperio
                                      height_t
                                      neighbor
                                      min_segm
                                      threshol
                                      seed_can

# run the algorithm
analysis.invalidate_results(seeds=True, object
objects = algo.run(analysis)
```

Hint: Explore the possibilities of py4dgeo with in-notebook help

```
?py4dgeo.SpatiotemporalAnalysis
```

```
Init signature:
py4dgeo.SpatiotemporalAnalysis(
    filename,
    compress=True,
    allow_pickle=True,
    force=False,
)
Docstring:      <no docstring>
Init docstring:
Construct a spatiotemporal segmentation object

This is the basic data structure for the 4D objects by change algorithm
and its derived variants. It manages storage of M3C2 distances and other
intermediate results for a time series of epochs. The original point clouds
themselves are not needed after initial distance calculation and additional
epochs can be added to an existing analysis. The class uses a disk backend
to store information and allows lazy loading of additional data like e.g.
M3C2 uncertainty values for postprocessing.

:param filename:
    The filename used for this analysis. If it does not exist on the file
    system, a new analysis is created. Otherwise, the data is loaded from the existent file.
```