CHANGE DETECTION AND DEFORMATION ANALYSIS BASED ON MOBILE LASER SCANNING DATA OF URBAN AREAS

Joachim Gehrung[1,2,*], Marcus Hebel[1], Michael Arens[1], Uwe Stilla[2]

[1] Fraunhofer IOSB, Ettlingen, Fraunhofer Institute of Optronics, System Technologies and Image Exploitation, 76275 Ettlingen, Germany - (joachim.gehrung, marcus.hebel, michael.arens)@iosb.fraunhofer.de
[2] Photogrammetry and Remote Sensing, Technische Universitaet Muenchen, 80333 Muenchen, Germany - stilla@tum.de

Commission II, WG II/10

KEY WORDS: Mobile Laser Scanning, Change Detection, Deformation Analysis, Occupancy Grid

ABSTRACT:

Change detection is an important tool for processing multiple epochs of mobile LiDAR data in an efficient manner, since it allows to cope with an otherwise time-consuming operation by focusing on regions of interest. State-of-the-art approaches usually either do not handle the case of incomplete observations or are computationally expensive. We present a novel method based on a combination of point clouds and voxels that is able to handle said case, thereby being computationally less expensive than comparable approaches. Furthermore, our method is able to identify special classes of changes such as partially moved, fully moved and deformed objects in addition to the appeared and disappeared objects recognized by conventional approaches. The performance of our method is evaluated using the publicly available TUM City Campus datasets, showing an overall accuracy of 88 %.

1. INTRODUCTION

Mobile laser scanning (MLS) is a fast and efficient way to collect high resolution 3D measurements of extensive areas. The possible benefits of such data are numerous. It can, for example, be used for disaster assessment, to extract street sign information or for the generation of city models. Regardless of the actual use case, the knowledge and products derived from the data needs to be kept up to date. A re-evaluation of all data once new measurements are available is inefficient. The active search for changed areas within the dataset is a much more suitable approach. In literature, this process is referred to as *automatic change detection*.

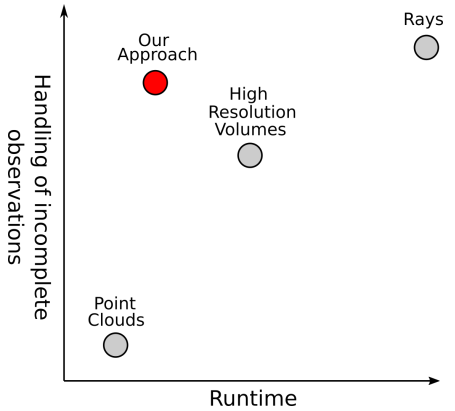

Figure 1. Our approch in context of change detection research.

The big challenge regarding automatic change detection is the incomplete observation of the environment. While some scenarios allow a complete observation of the area-of-interest, most real-world datasets suffer from occlusions or the fact that some areas are only observed in one epoch, but not in the other. Is a method unaware that some parts of the dataset have not been observed in both epochs, then it will accidently flag objects within these areas as changed. For this reason, it is necessary that a change detection method is able to handle such incomplete observations. One way to do this is to consider the occupancy information embedded within the range measurements.

There are several categories of change detection algorithms that can be classified by *runtime* and their *capability to handle incomplete observations*. An illustration can be seen in Figure 1. Approaches solely based on point clouds are not able to handle incomplete observations, but are usually quite fast. Less fast are volumetric methods such as occupancy grids, since they are usually generated using raycasting-like algorithms. They are able to recognize incomplete observations, however, at the cost of discretization errors. Ray-based methods are the most precise since they use all of the available information. Unfortunately, they are required to have a quadratic runtime complexity and therefore are much slower than algorithms based on volumes and point clouds.

Our approach combines the best of two worlds. We use point clouds in order to get a high spatial resolution and apply information extracted from a low-resolution occupancy grid to include occupancy information. Therefore we are able to get fast runtimes in combination with the capability to handle incomplete observations. In addition, our approach allows to easily identify special cases such as partially moved, fully moved and deformed objects, whereas other approaches usually only identify appeared and disappeared objects.

Section 2 gives an overview over the state-of-the-art of occupancy grids, change detection and deformation analysis. In Section 3, we give a short definition of what we define as *changes* and describe the different classes of changes identified by our approach. A detailed explanation of our method follows in Section 4. The capabilities of our approach are evaluated in Section 5 based on a synthetic dataset especially created for this purpose, as well as the TUM City Campus datasets.

*Corresponding author

2. RELATED WORK

2.1 Occupancy grids

Occupancy grids are a memory-friendly way to store information about free, occupied and unobserved areas. An approach for indoor mapping proposed by (Moravec, Elfes, 1985) utilizes a 2D grid with occupancy information. The latter one was represented by a probability stored in each of the grid cells, implying either free, occupied or unseen space. The grid has been generated using ultrasonic sensors. Although the aperture angle of each sensor was wide, the resulting grid turned out to be a quite accurate representation of the environment.

An octree-based approach to represent arbitrary 3D structures has been presented by (Meagher, 1982). Initially only binary occupancy information had been considered, but later probabilistic occupancy information has been added by (Payeur et al., 1997). (Hornung et al., 2013) introduced propability clamping in order to ensure fast adjustment of the representation to a dynamic environment. This also enabled a nearly lossless compression strategy. The framework implemented by the authors has gained popularity within the robotics community and is known by the name *OctoMap*.

Based on the theoretical foundation provided by this work, we proposed a concept for occupancy representation on a global scale (Gehrung et al., 2016). The problem of artifacts caused by the discretization inherent to occupancy grids has been solved using an iterative refinement algorithm for occupancy octree generation (Gehrung et al., 2018).

2.2 Change detection

The meaning of the term *change detection* depends on the context and the field of research in which it is used. It plays an important role in photogrammetry and remote sensing, where it is used for tasks such as *urban area monitoring* and *updating geo-databases* (Du et al., 2016). Approaches similar in effect, but different in name can be found in the field of 3D computer vision. Here it is applied to tasks such as the *Detection and Tracking of moving Objects (DATMO)* (Litomisky, Bhanu, 2013) or for the *distinction of dynamic objects and static background* (Azim, Aycard, 2012). However, a far better categorization of methods can be made on the basis of the sensor system used, more precisely on the processing of the measurements generated by it. Since this work deals with mobile laser scanning, only approaches that use 3D data are considered.

Surface points. Point clouds are a common representation for 3D measurements, so using them in change detection approaches is straightforward. (Girardeau-Montaut et al., 2005) proposed several simple cloud-to-cloud comparison algorithms, organizing the point clouds using an octree in order to increase performance. (Zeibak, Filin, 2008) converted point clouds from a stationary terrestrial laserscanner into depth images in order to compare multiple epochs with each other. The downside of these approaches is that they can't distinguish between changes and incomplete observations.

Rays. In addition to the surface points, 3D measurements usually provide information about the free space traversed by the measurement. This information is critical in order to distinguish changes and incomplete observations. A popular approach utilizing this was proposed by (Underwood et al., 2013). A ray comparison strategy based on a spherical grid is used in order to compare point clouds from different epochs or view points. (Hebel et al., 2013) apply the Dempster-Shafer theory of evidence to derive changes based on rays originating from airborne laser scanning. A voxel grid is used for performance reasons. (Xiao et al., 2015) proposed a similar approach for mobile laser scanning data. Methods of this category allow accurate conclusion, but require the evaluation of all rays within the region of interest and are therefore computationally very expensive.

Cluster based. Some approaches utilize segmentation techniques in order to extract clusters, which in turn are used for the actual change detection. This requires well working segmentation algorithms or classifiers for cluster extraction. (Schachtschneider et al., 2017) use an occupancy grid in order to assess the temporal behavior of clusters extracted from point clouds of urban environments. (Aijazi et al., 2013) proposed an approach that classifies clusters into known permanent and temporary classes. A similarity map derived from an evidence grid is used, inter alia, for change detection.

Occupancy grids. Grid based occupancy representations provide similar advantages in terms of information as the ray-based ones, albeit in a condensed form. (Pagac et al., 1996) utilized the Dempster-Shafer theory of evidence to generate a 2D occupancy grid that serves as environment representation for autonomous driving. (Wolf, Sukhatme, 2004) use a similar approach for robot navigation that utilizes two grids to determine static and dynamic parts of a scene. (Azim, Aycard, 2012) also divide measurements of an environment into static and dynamic elements by utilizing conflict search on an occupancy grid based on the Octomap framework. We have successfully applied probabilistic occupancy grids for short-term change detection in order to detect and extract moving objects (Gehrung et al., 2017). Additionally we proposed a fast indicator for detecting changes (Gehrung et al., 2019). An octree-based structure utilizing Gaussian kernels has been used in order to represent the occupancy state of the environments. Thereby, we also coined the term *delta octree*, which describes the differences of occupancy and observation between two epochs.

Most of the approaches mentioned above do not scale to large amounts of sensor data, such as the one generated by a state-of-the-art mobile laser scanning system. Surface point based approaches as well as cluster-based ones are not able to handle occlusions, ray-based approaches encounter performance issues. Approaches based on occupancy grids either suffer from information loss due to crude resolutions or require great effort in order to generate high spatial resolutions. Also, none of the proposed techniques can correctly recognize partly moved objects out-of-the-box.

2.3 Deformation analysis

(Lindenbergh, 2010) distinguishes between change detection and deformation analysis. In contrast to change detection, deformation analysis does not determine a purely binary change, but rather the degree of change. There are different classes of approaches, each one of them working on a different abstraction level. Point-by-point deformation analysis uses individual measurement points or the corner points of a regular grid, like the approach proposed by (Lague et al., 2013). Object-oriented deformation exploits some kind of object representation, such as geometric primitives (Xu et al., 2013) or building footprints (Rutzinger et al., 2010). A middleway between both classes of approaches are morphological maps, where the distances between points and a geometric representation of the object are determined (Pesci et al., 2013).

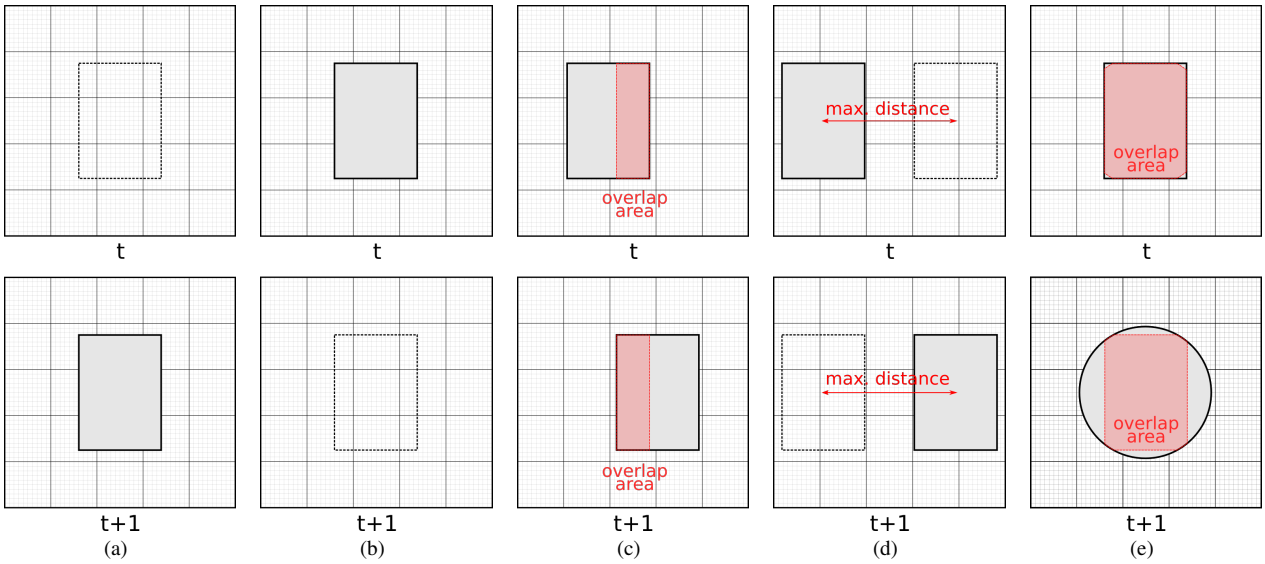

Figure 2. The change detection classes, (a) *appeared*, (b) *disappeared*, (c) *partially moved*, (d) *fully moved* and (e) *deformed*.

3. THEORETICAL PRIMER TO CHANGE DETECTION

3.1 About the term *changes*

The scope of this work is on long-term changes that typically appear when comparing epochs that have a timespan of days, weeks or month in between. Short-term changes such as moving objects are not considered. Furthermore, we define *change* as the result of a semantically meaningful action that is applied to some kind of object instance. Examples for this would be moving a car or breaking a hole into a wall.

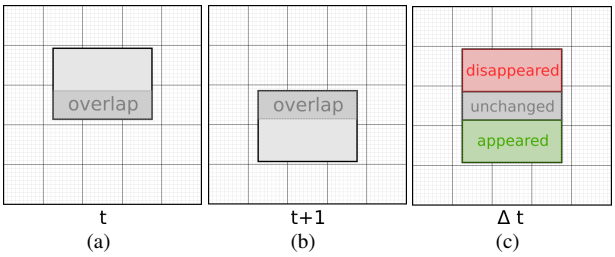

Figure 3. An object partially moved between (a) epoch *t* and (b) epoch *t+1* is usually recognized as (c) an appeared and a disappeared cluster with an unchanged area in between.

Usually change detection is limited to determining appeared and disappeared areas by comparing two or more epochs. This leads to the following problems. Whenever an object has been partially moved, it is broken down into an appeared, a disappeared and an unchanged part instead of being recognized as the same object at different locations (see Figure 3). Also, when an object is replaced by another one of about the same size, detecting a change by comparing surface points or occupancy information only could lead to partial or no detection at all.

3.2 Class definitions

In order to deal with the problems mentioned above, the change detection approach presented in this work classifies changes into five classes and a residue class. An illustration can be seen in Figure 2. The classes are defined as follows.

Appeared. An object instance appeared. The space encompassed by it is free in epoch t but occupied in epoch $t+1$.

Disappeared. An object instance disappeared. The space encompassed by it is occupied in epoch t but free in epoch $t+1$.

Partially moved. An object instance has been moved between epoch t and $t+1$, but in a way that it overlaps when comparing both epochs with each other.

Fully moved. An object instance has been moved between epoch t and $t+1$, but in a way that it does not overlap when comparing both epochs with each other. The object instance has the same appearance in both epochs and the distance between its center points is less than a predefined threshold.

Deformed. An object instance encompasses the same space in both epochs t and $t+1$ and has approximately the same dimensions, but a different form. If an object instance exists in both epochs, but with a significantly different size, the object instance is considered to be disappeared and another one to be appeared.

Unchanged Residue class which is applied to object instances that are considered to be unchanged. It is used in case no class of the above applies.

Appeared and *disappeared* are the classes usually used in change detection. The *partially moved* class solves the typical change detection related issue mentioned above and is a valuable source of information, as the class *fully moved* is. The *deformed* class is required to denote object instances that have significantly changed their appearance or have been replaced by instances of other objects.

3.3 Reasons for combining voxels and point clouds

As mentioned in Section 2.2, change detection methods either utilize volumes or some variation of point clouds. An advantage of point clouds is their ability to represent the environment with the highest resolution available, given the sensor measurements. However, point clouds gained from mobile laser scans do only provide occupancy information if the latter ones are processed as rays, but the evaluation of all available rays within an area

is a time-consuming process, especially when considering large areas or huge amounts of measurements common for mobile laser scanning.

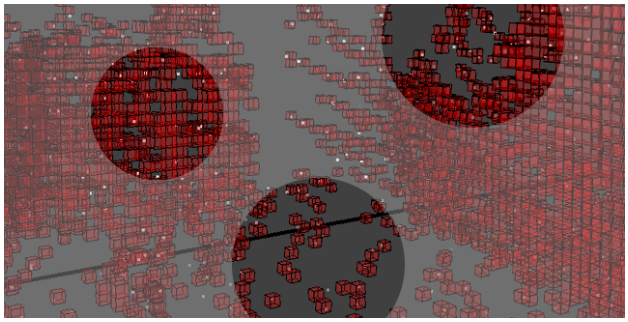

Figure 4. Whenever the environment is insufficiently sampled by the laserscanner, holes appear in occupancy grids of high resolution.

Volume-based data structures like octrees or occupancy grids have the inherent advantage that they are able to represent occupancy as well as explorational information. They enable the derivation of a so called *delta octree* that encodes information about changes between epochs (Gehrung et al., 2019). It is also able to mark areas for which no statement about a change of occupancy can be made. The challenge with volume-based technologies is that the maximum resolution is limited. For one thing, this is due to performance issues regarding the voxel generation. Even fast algorithms such as the one using iterative refinement previously proposed by us (Gehrung et al., 2018) require large runtimes for resolutions of about 10 cm.

Another limitation is the well known *Nyquist-Shannon sampling theorem* that describes the condition required for the sampling of a continuous signal. In this case it can be used as an analogy to understand the effects that take place when an environment is reconstructed from laser scanning data. The generation of a voxel grid from laser scans represents such a reconstruction. Figure 4 shows the effects appearing when a voxel grid is constructed using a resolution too fine for the available laser scans. A fragmentation occurs in a way that the comparison of two epochs would compare most voxels with unseen areas. The reason for this is that the laser scanner sampled the environment in a frequence too low to allow for its reconstruction at the given resolution. The correct frequency would be a function of the scanner mount angle, the distance to the surface and the speed of the mobile mapping system. As a result, the actual sampling frequency varies greatly in practice.

This work attempts to combine the advantages of both volumes and point clouds. The goal is to propose a change detection algorithm that is able to utilize occupancy and explorational information from a voxel-grid with a coarse resolution in order to detect and evaluate candidates selected from a point cloud with a higher resolution. This approach is also intended to solve the problems regarding partially moved and deformed objects described in Section 3.1.

4. VOXEL-ENHANCED CHANGE DETECTION

4.1 Overview

This section provides a quick overview of the approach presented in this paper. A full explanation is given in the following sections. A flow chart illustrating the process can be seen in Figure 5. The input consists of the point clouds of both epochs as well as the delta octree. In a first step, candidates are selected from the point clouds of both epochs. Clusters representing appeared and disappeared volumes are extracted from the delta octree and intersected with the candidates. This results in what we refer to as *basic instances*. These represent object instances that either appeared or disappeared. This term has been chosen in order to describe the kind of information usually generated by state-of-the-art change detection approaches.

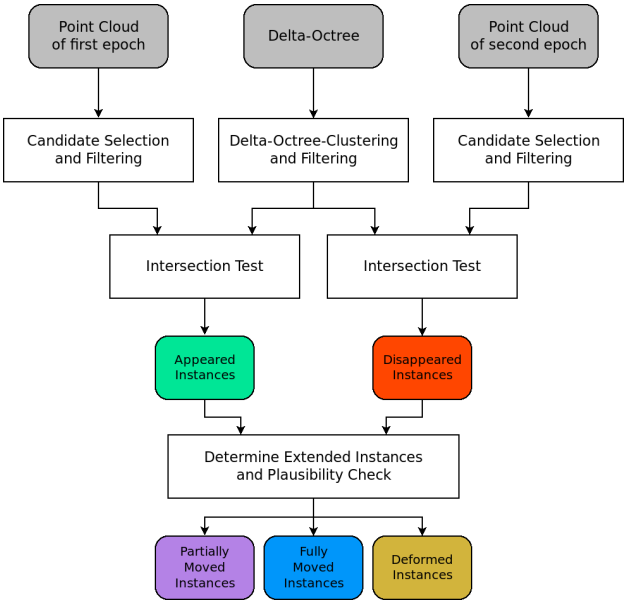

Figure 5. The individual sections of the change detection method presented in this work.

Based on the basic instances, the *extended instances* are determined. These are *partially and fully moved object instances*, as well as *deformed object instances*. In a final step, the basic and extended instance lists are checked for plausibility. Each individual step is described in the subsequent sections.

4.2 Candidate selection from point clouds

The candidate selection describes the segmentation of the point cloud into clusters. These clusters are considered to have some kind of semantic meaning, they are therefore instances of an object set. They can, for example, correspond to people, cars, vegetation or buildings. For the sake of simplicity, we assume that removing the ground plane and applying a simple Euclidean clustering algorithm will provide said clusters.

Filtering is an important step following the clustering, be it in order to remove clutter or to provide additional boundary conditions. These could be a given cluster dimension when searching for persons or if a focus on large objects is required. The following criteria are used, specified as intervals.

- Number of points
- Cluster height, weight and length
- Height above ground

Other criteria are possible, therefore the above list is not necessarily complete.

4.3 Extracting voxel-clusters from the delta octree

The delta octree is a volumetric data structure that encodes the changes within the occupancy and observation of the environment. Note that all ground points had been removed before generating the occupancy grid in order to increase performance and to avoid clutter. The delta octree is segmented into clusters representing appeared and disappeard objects. Starting from a seed list, flood filling is applied in order to get said clusters. As with point clouds, filtering is used. The following criteria are used, specified as intervals:

- Volume
- Height above ground

Other criteria are possible, therefore the above list is not necessarily complete.

4.4 Determining basic instances

The lists of *appeared* and *disappeared* object instances, the *basic instances* are determined by calculating the overlap between the candidates extracted from the point clouds and the voxel clusters. All voxel clusters are intersected with all point cloud clusters within a given range. For each correspondence, the proportion of points inside the voxels of the cluster is calculated. Filtering based on the activity threshold t_a is applied in order to determine whether or not a candidate is added to the list. This process is executed for appeared and disappeared clusters.

4.5 Determining extended instances

The *extended instances* are derived from the basic instances. They represent *partially moved*, *fully moved* and *deformed* object instances. A search within the local neighborhood between both appeared and disappeared object instances is executed. Therefore, all appeared and disappeared instances are compared with each other. If the volumes of the bounding boxes have about the same order of magnitude, further comparisons as shown in Figure 6 take place.

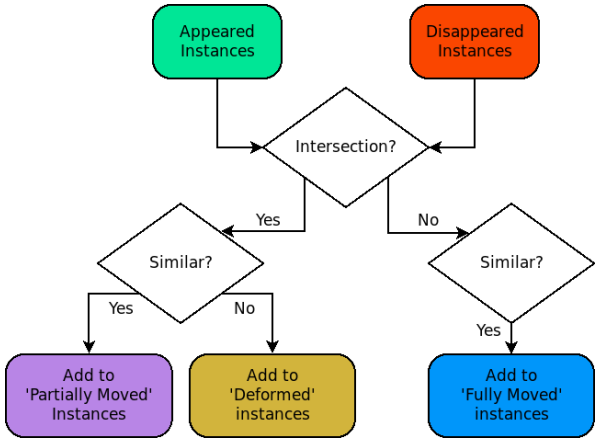

Figure 6. The case discrimination that is applied to determine the extended instances.

An intersection test based on axis-aligned bounding boxes is applied. If it is successful, an advanced intersection test using a k-d tree is used in order to determine whether or not object instances actually intersect. If they do, their similarity is compared using the technique described in Section 4.6. If there is a similarity, the object instance pair is considered to be *partially moved*. In case they are not similar, the pair is considered to be *deformed*. If there is a similarity but no intersection, the object is considered to be *fully moved*.

4.6 Comparing candidate similarities

Basic instances are checked for similarity by comparing the point clouds representing them. Two point clouds A and B of different epochs are compared in a point-wise manner. Since it cannot be assumed that there are no occlusions and the environment is completely observed, observability information extracted from the delta octree is incorporated. The difference between two clouds is determined using the error function e. Both clouds are considered to be similar to each other if the function value is less or equal to a similarity threshold t_s.

Before both clouds can be compared, cloud A needs to be aligned to cloud B. Therefore, an initial transformation is determined using a centroid-based technique. Based on this, RANSAC-enhanced ICP is applied. Cloud A is then compared to cloud B in a point-wise manner using a variation of the mean square error. It is defined as

$$e_A^B = \sum_{i=1}^{N_A} d_i. \tag{1}$$

N_A is the number of points in cloud A. The deviation d_i between two points in cloud A and B is defined as

$$d_i^{AB} = \begin{cases} \left(p_i^A - p_i^B\right)^2 & \text{if neighbor in B} \\ \frac{4t_s}{N_A} & \text{if no neighbor in B} \\ 0 & \text{if area unobserved in B.} \end{cases} \tag{2}$$

For each point p_i^A in cloud A, the neighbor point p_i^B in cloud B is determined. A maximum distance is used for the neighborhood search. If a neighbor is found, d_i is set to the squared distance between both points, otherwise to a penalty constant. The constant is designed in a way that a comparison to the similarity threshold t_s fails as soon as a predefined percentage of all points in cloud A does not have a neighbor in cloud B. The percentage of points is defined by the penalty threshold t_p. In case no neighbor is found, the delta octree is checked whether or not the area has been observed in cloud B. If this is the case, d_i is set to zero, since it is impossible to say whether or not that part of the object instance has changed or not been observed. The error function e combines both comparisons e_A^B and e_B^A. It is defined as

$$e = \log\left(e_A^B \frac{N_A}{N_A + N_B} + e_B^A \frac{N_B}{N_A + N_B}\right), \tag{3}$$

where the score for a comparison in each direction is weighted by the number of involved points. The logarithm is used to stretch the result space in a way that benefits the selection of the similarity threshold t_s.

4.7 Plausibility check

The purpose of the plausibility check is it to keep the lists containing the basic instances consistent with the lists of the extended instances. Since the latter ones are derived from the former ones, there are some object instances that are part of both the basic and extended instances. A simple search is applied to remove all of these object instances from the basic instance lists.

5. EXPERIMENTS

5.1 Synthetic dataset

In order to verify that our approach is capable to identify the change classes defined in Section 3, we created a synthetic LiDAR dataset using the **Blensor plugin** for the free software **Blender**. For each class we created a simple three-dimensional scene, each with two epochs. Each scene consists of a flat surface with one or more geometric primitives and is available in two versions. In a fully observed version, the scene has been scanned from all cardinal directions, therefore four LiDAR scans were generated. In the partially observed version, only a single LiDAR scan showing the scene from one cardinal direction has been created. In addition, a demo scene containing examples for all classes has been created. It serves for illustration purposes only.

5.2 TUM MLS datasets

The evaluation with real-world data is based on the **TUM City Campus**[1] datasets (Gehrung et al., 2017). They consist of two measurement rides that took place in spring 2016 and winter 2018. It is well suited for change detection, since there are a variety of minor and major changes between both epochs. Both datasets are publicly available under a Creative Commons License.

The datasets have been recorded using the measurement vehicle MODISSA (Borgmann et al., 2018). It is equipped with two Velodyne HDL-64E LiDAR sensors mounted at an angle of 25 ° on the vehicle front roof. This allows recording both the area in the vicinity of the vehicle as well as building facades. The datasets consist of a sequence of individual scans, with each scan covering approximately 0.1 seconds of data aquisition. All single points in all scans have individually been georeferenced using the on-board Applanix POS LV navigation system that combines navigation data from two GNSS antennas, an inertial measuring unit and a distance measurement indicator (Diehm et al., 2020).

In order to further increase the quality of intra- and inter-epoch registration of the scans in the dataset, we applied methods known from graph-based SLAM. In an additional fine-registration step, we combined multiple consecutive scans to chunks which in turn were registered against an accumulated point cloud of the first epoch using ICP. From this data we extracted a subset containing the TU Munich inner city campus.

Labels for all moving objects were manually generated and used to remove them from both epochs. In addition, we created labels for all objects that disappeared from the first epoch and appeared in the second one. The labels are used during the evaluation as ground truth for changed objects. Evaluating the plausibility of an object instance class is done by a human observer, due to reasons mentioned in Section 6.2.

[1] http://s.fhg.de/mls2

6. RESULTS AND DISCUSSION

6.1 Synthetic dataset

The synthetic dataset has been used for developement purposes and to determine initial parameters for the real-world data. For cluster activity threshold and penalty threshold $t_a = 0.1$ and $t_p = 0.25$ were found to be appropriate. The similarity threshold has been defined as $t_s = 1.5$. The maximum search distance for neighbors is set to 0.3 m. The synthetic dataset shows that object instances in every scene are recognized and classified correctly, even if the object instance is only partly observed. It also shows that there is a clear difference in the error functions value for objects that are similar ($e < 1.0$) and those that are not ($e > 4.0$). Furthermore, it demonstrates that considering the effects of incomplete observation is important in order to suppress apparent deviations of object instances due to occlusion. Figure 7(a) and 7(b) show the demo scene.

6.2 TUM MLS datasets

For reasons explained in (Gehrung et al., 2016), we broke the dataset down into 16 3D tiles, each of them with an edge length of 32x32x32 m. Our change detection method was applied to each of them separately. The parameters are the same as in Section 6.1. The edge length of a voxel is 0.5 m. Figure 7(c) and 7(d) shows pedestrians correctly identified as *fully moved*. It also demonstrates the effect of segmentation errors. In the first epoch the point cluster of the dumpster is merged with the point cluster of the building. Therefore the dumpster is recognized as *appeared*, not as *deformed*. Figure 7(c) and 7(d) demonstrate that an object instance can be part of multiple extended instances. The golden object could either have been deformed (switched with another, unknown object) or been replaced by one of the objects to both of its sides.

		Predicted					
		App.	Dis.	P.M.	F.M.	Def.	Un.
Actual	App.	45	1	6	1	5/2	29/0
	Dis.	–	103	–	–	9/2	28/13
	P.M.	–	–	10	–	–	3/2
	F.M.	–	–	–	26	–	12/1
	Def.	4/3	1/0	–	–	14	13/3
	Un.	4/2	5	–	–	3/0	–

Table 1. The confusion matrix achieved by applying our method to the TUM City Campus dataset. Values after the slash result from excluding segmentation errors.

During the evaluation, it turned out that the main cause of incorrect classifications can be attributed to the above-mentioned segmentation errors. Those distort the results and make the evaluation challenging, since the assessment of such cases requires interpretation by a human observer. This is the reason why the qualitative evaluation was done based on labels, but manually. Table 1 shows the quantitative results of the evaluation. The overall accuracy of our change detection method is 88 %. The exclusion of all misclassifications caused by segmentation errors would lead to an overall accuracy of 93 %.

It should be noted that both the *partially moved* and *deformed* cases are too underrepresented to draw meaningful conclusions. This is mostly caused by said segmentation errors, but also due to the nature of the datasets. Table 2 shows precision and recall with and without considering segmentation errors. Both precision and recall are high for classes with many examples. The precision for *appeared* and *disappeared* object instances is

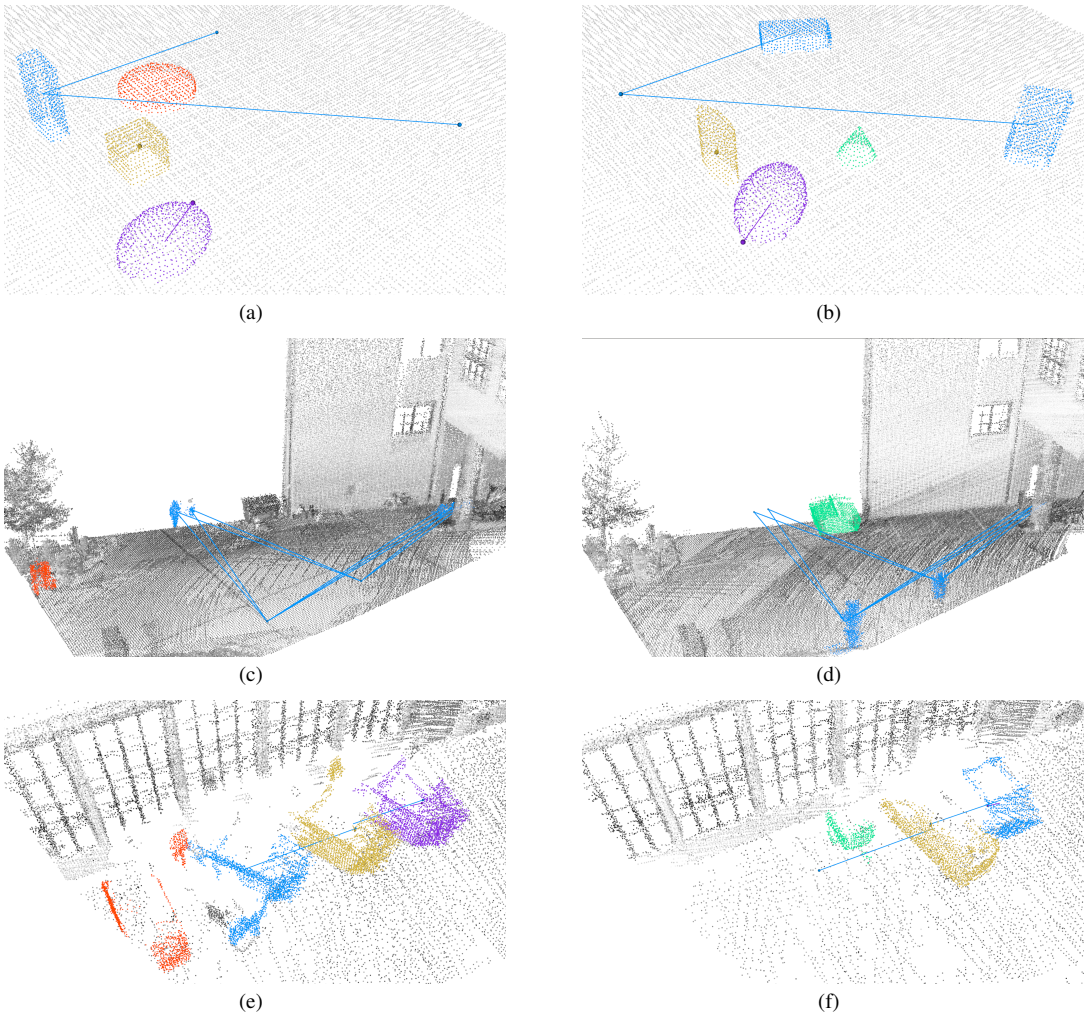

Figure 7. The demo scene from the synthetic dataset with two epochs (a) and (b). Two scenes (c,d) and (e,f) with two epochs from the TUM City campus datasets. The classes *appeared* (green), *disappeared* (red), *partially moved* (purple), *fully moved* (blue) and *deformed* (gold) are shown. Lines symbolize associations of object instances between epochs.

	Appeared	Disappeared	Partially moved	Fully moved	Deformed
Precision	84.9%/90.0%	93.6%/94.5%	62.5%/62.5%	96.3%/96.3%	45.2%/77.8%
Recall	51.7%/81.8%	73.6%/87.3%	76.9%/83.3%	68.4%/96.3%	43.8%/70.0%

Table 2. Precision and recall for all change classes with and without considering segmentation errors.

high, since those are the simplest cases to detect. The precision for *partially moved* and *deformed* object instances is in the medium range, most likely because of the underrepresentation of these cases as mentioned above. Both precision and recall increase drastically once the segmentation errors are excluded.

6.3 Runtimes

All reported results were generated on a machine with 64 Gigabyte of RAM and an Intel Core i7 processor with 3.5 GHz and 36 cores. The accumulation of all LiDAR-measurements and and distribution on 3D tiles needs about 10 minutes. Generating the occupancy grid requires 17.9 minutes per 3D tile and epoch on average, that is 4.8 hours per epoch. Creating the delta octrees for all 3D tiles and epochs takes less than 2 seconds. The runtime for our change detection approach varies with the amount of cloud clusters and the number of points within. It is between 30 seconds and 10 minutes. While the overall runtime of our method for the TUM City Campus datasets is in the range of hours, point-based approaches usually require minutes, whereas ray-based methods such as the one presented by (Underwood et al., 2013) require days.

7. CONCLUSION AND FUTURE WORK

In this paper we presented a novel change detection approach that is able to identify *partially moved*, *fully moved* and *deformed* object instances. Our method is able to handle incomplete observations, but is far from being as computationally expensive as comparable approaches. The overall accuracy is 88 %, but can be improved to at least 93 % by eliminating segmentation errors in the candidate selection. For future work, we propose to solve these segmentation errors using advanced techniques auch as semantic segmentation. Furthermore, we intend to further decrease the runtime by only generating the occupancy grid at locations that contain possible candidates.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-2-2020-703-2020 | © Authors 2020. CC BY 4.0 License.

REFERENCES

Aijazi, A. K., Checchin, P., Trassoudaine, L., 2013. Detecting and Updating Changes in Lidar Point Clouds for Automatic 3D Urban Cartography. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5/W2, 7-12.

Azim, A., Aycard, O., 2012. Detection, classification and tracking of moving objects in a 3D environment. *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, 802–807.

Borgmann, B., Schatz, V., Kieritz, H., Scherer-Klöckling, C., Hebel, M., Arens, M., 2018. Data processing and recording using a versatile multi-sensor vehicle. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1, 21–28.

Diehm, A. L., Gehrung, J., Hebel, M., Arens, M., 2020. Extrinsic self-calibration of an operational mobile LiDAR system. M. D. Turner, G. W. Kamerman (eds), *Laser Radar Technology and Applications XXV*, 11410, International Society for Optics and Photonics, SPIE, 46 – 61.

Du, S., Zhang, Y., Qin, R., Yang, Z., Zou, Z., Tang, Y., Fan, C., 2016. Building Change Detection Using Old Aerial Images and New LiDAR Data. *Remote Sensing*, 8(12). http://www.mdpi.com/2072-4292/8/12/1030.

Gehrung, J., Hebel, M., Arens, M., Stilla, U., 2016. A Framework for Voxel-based Global Scale Modeling of Urban Environments. *ISPRS International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W1, 45–51.

Gehrung, J., Hebel, M., Arens, M., Stilla, U., 2017. An approach to extract moving objects from MLS data using a volumetric background representation. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-1/W1, 107–114.

Gehrung, J., Hebel, M., Arens, M., Stilla, U., 2018. A voxel-based metadata structure for change detection in point clouds of large-scale urban areas. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2, 97–104.

Gehrung, J., Hebel, M., Arens, M., Stilla, U., 2019. A fast voxel-based indicator for change detection using low resolution octrees. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5, 357–364.

Girardeau-Montaut, D., Roux, M., Marc, R., Thibault, G., 2005. Change detection on point cloud data acquired with a ground laser scanner. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36-3/W19, 30–35.

Hebel, M., Arens, M., Stilla, U., 2013. Change detection in urban areas by object-based analysis and on-the-fly comparison of multi-view ALS data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 86, 52–64.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., Burgard, W., 2013. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots*, 34(3), 189-206. http://octomap.github.com.

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, 10-26.

Lindenbergh, R., 2010. *Airborne and Terrestrial Laser Scanning*. Whittles Publishing, Dunbeath, chapter 7, 237–369.

Litomisky, K., Bhanu, B., 2013. *Removing Moving Objects from Point Cloud Scenes*. 7854, Springer Berlin Heidelberg, Berlin, Heidelberg, 50–58.

Meagher, D., 1982. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2), 129–147. http://dx.doi.org/10.1016/0146-664x(82)90104-6.

Moravec, H., Elfes, A., 1985. High Resolution Maps from Wide Angle Sonar. *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 116 – 121.

Pagac, D., Nebot, E. M., Durrant-Whyte, H., 1996. An evidential approach to probabilistic map-building. *Proceedings of IEEE International Conference on Robotics and Automation*, 1, 745–750 vol.1.

Payeur, P., Hebert, P., Laurendeau, D., Gosselin, C. M., 1997. Probabilistic octree modeling of a 3D dynamic environment. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2, 1289–1296.

Pesci, A., Teza, G., Bonali, E., Casula, G., Boschi, E., 2013. A laser scanning-based method for fast estimation of seismic-induced building deformations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 79, 185-198.

Rutzinger, M., Rüf, B., Vetter, M., Höfle, B., 2010. Change detection of building footprints from airborne laser scanning acquired in short time intervals. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38, 475–480.

Schachtschneider, J., Schlichting, A., Brenner, C., 2017. Assessing temporal behavior in LIDAR point clouds of urban environments. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1/W1, 543–550.

Underwood, J. P., Gillsjö, D., Bailey, T., Vlaskine, V., 2013. Explicit 3D change detection using ray-tracing in spherical coordinates. *2013 IEEE International Conference on Robotics and Automation*, 4735–4741.

Wolf, D., Sukhatme, G. S., 2004. Online simultaneous localization and mapping in dynamic environments. *2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04*, 2, 1301–1307.

Xiao, W., Vallet, B., Brédif, M., Paparoditis, N., 2015. Street environment change detection from mobile laser scanning point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 38, 38-49.

Xu, S., Vosselman, G., Elberink, S. O., 2013. Detection and Classification of Changes in Buildings from Airborne Laser Scanning Data. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, II-5/W2, 343–348.

Zeibak, R., Filin, S., 2008. Change detection via terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36, 430–435.