A FAST VOXEL-BASED INDICATOR FOR CHANGE DETECTION USING LOW RESOLUTION OCTREES

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

[1] Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB,
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: Change Detection, Local Deformation Analysis, Volumetric Environment Representation

ABSTRACT:

This paper proposes a change detection approach that uses a low-resolution octree enhanced with Gaussian kernels to describe free and occupied space. This so-called Gaussian Occupancy Octree is derived from range measurements and used to represent spatial information for a single epoch. Changes between epochs are encoded using a Delta Octree. A qualitative and quantitative evaluation of the proposed approach shows that its advantages are a fast runtime and the ability to make a statement about the re-exploration of space. An evaluation of the classification accuracy shows that our approach tents towards correct classifications with an overall accuracy of 51.5 %, but is also systematically biased towards the appearance of occupied space.

1. INTRODUCTION

Mobile mapping is a fast and efficient way to capture high resolution data of extensive areas. It is an important source of street-level information for the ever-growing field of geoinformation services. Already today, companies use data collected by mobile mapping systems to enrich their services with additional information, be it speed limits extracted from street signs for navigation systems or 3D building models to provide additional context to mapping services. Modern LiDAR sensors, which are often applied in mobile laser scanning (MLS), usually sample the environment with a very high frequency and resolution, therefore accumulating large amounts of dense sensor data even over small periods of time. Completely recording an urban environment such as a small or medium-sized city often takes multiple hours and - due to the naturally grown structure of a city - requires to follow the same streets multiple times in order to gain a complete scan of the area. The collected point clouds therefore often contain redundant information, have a resolution too high for the task at hand or simply need to be downsized before processing is possible at all. A good starting point to deal with these large amounts of point clouds are occupancy grids.

Occupancy grids represent the spatial occupancy information derived from range measurements, thereby compressing the underlying data due to discretization. The loss in accuracy can be quantified and adjusted to the task at hand. Given a maximum resolution, occupancy grids have an upper memory boundary which allows the estimation of memory requirements. These memory-related properties make them very valuable for long-term data storage. State-of-the-art approaches even incorporate uncertainty, thereby allowing better handling of real-world sensor data. Since the representation of free and unexplored space is possible, occupancy grids support tasks such as change detection. The goal of automatic change detection is to autonomously analyze two or more epochs of spatial data, thereby identifying and quantifying changes, summarizing these in a way that benefits the task at hand. This may be a heatmap that highlights areas with a high amount of changes or more high-level information such as the approximate living space gained or lost due to the erection or tear-down of a building.

In the context of mobile mapping, change detection can be used to evaluate the recordings of a measurement vehicle during a measurement run. The process of recording an environment usually includes a driver following the directions given by a navigator. It is the task of the latter to plan a course that leads to a comprehensive coverage of the area. For the navigator, it is not easy to determine whether or not the area has been sufficiently recorded. Since change detection based on occupancy grids is also able to identify areas which have been visited before, but not again, this information can be used to guide the vehicle back into areas where more information gathering is required. Information about changes in an area as well as missing areas can be presented vividly in form of heatmaps. These allow a navigator to quickly assess the situation and determine the best course of action.

The main contribution of this paper is a change detection approach based on a hierarchical occupancy grid with dynamic resolution. The approach uses local deformation analysis based on Gaussian kernels in order to enhance the resolution the octree and to address drawbacks of classical occupancy grids.

2. RELATED WORK

2.1 Occupancy Grids

Occupancy grids are a potent way of representing spatial information, since they not only represent free and occupied space, but also allow to deduce the location of unobserved areas. Moravec and Elfes (1985) proposed an approach for indoor mapping using ultrasonic sensors. A 2D occupancy grid is used to describe the state of the environment along a plane on the sensor level. A probability describes the occupancy of each grid cell, which can either be free, occupied or unseen. Although the aperture angle of

*Corresponding author

each ultrasonic sensor is quite wide, the resulting representation is considerably accurate. An approach to represent arbitrary geometries using an octree has been presented by Meagher (1982). Initially it has been designed for binary occupancy only, but later has been extended to use probabilistic information (Payeur et al., 1997). Hornung et al. (2013) introduced propability clamping to ensure fast adaption of the representation to a changing environment. This also allows for a nearly lossless compression strategy. The framework implemented by the authors has gained huge popularity within the robotics community and is known by the name *OctoMap*. Based on the theoretical foundation of this work, we proposed a concept for occupancy representation on a global scale (Gehrung et al., 2016). Later on, an algorithm for iterative refinement has been proposed that also prevents artifacts caused by discretization inherent to occupancy grids (Gehrung et al., 2018). It was also shown that the probabilistic occupancy grids can be used to extract moving objects (Gehrung et al., 2017).

2.2 Change Detection

Change detection is a generic description for a topic that can be found within many different research fields. Considered in the context of urban surveying, change detection can be seen as being part of *Geospatial Sciences* and *Remote Sensing* (Du et al., 2016). However, variations of it can also be found within some areas of 3D computer vision, such as the *Detection and Tracking of moving Objects (DATMO)* (Litomisky and Bhanu, 2013) or the distinction of dynamic objects and static background (Azim and Aycard, 2012). Since the focus of this work is on 3D data collected via MLS, this overview will only consider respective approaches which can essentially be divided into the following categories.

Surface points. Approaches based on surface point coordinates can be considered as one of the most straightforward ways to do change detection, since point clouds are one of the most common representations for range measurements. Girardeau-Montaut et al. (2005) proposed several simple cloud-to-cloud comparison algorithms that utilize an octree for spatial organisation of the point cloud. Zeibak and Filin (2008) compare multiple epochs of data of a stationary terrestrial laser scanner using depth images in order to identify changes in its field of vision.

Rays. Since point clouds result from range measurements, each point can be interpreted as a ray between the sensor's position and the measured surface point. This has the advantage that, in addition to the measured point, also the traversed free space is considered. The approach proposed by Underwood et al. (2013) utilizes a ray comparison strategy based on spherical coordinates for rays originating from two point clouds from different epochs or view points. Hebel et al. (2013) apply the Dempster–Shafer theory of evidence to determine changes based on rays from an airborne laser scan, thereby using a voxel-based data structure for efficient ray management. Xiao et al. (2015) proposed a similar approach and applied it to MLS data. Such approaches require the storage and evaluation of every ray within the region of interest.

Semantic units. Approaches of this category require a segmentation of surface points into clusters. Since these usually have some degree of semantic meaning, e.g. because of prior knowledge included in the process, they may be considered to be semantic units such as objects. The actual change detection is then executed based on these units. Schachtschneider et al. (2017) assess the temporal behavior of clusters extracted from point clouds of urban environments using an occupancy grid. Aijazi et al. (2013) classify clusters into known permanent and temporary classes. A similarity map derived from an evidence grid is used, inter alia, for change detection. All approaches of this kind require a reliable segmentation approach and - in case of classification - a well working classifier with predefined object classes.

Occupancy grids. Due to the nature of grid based occupancy representation, this category of approaches has the most information available, since in addition to occupied and free space, also unobserved space can be considered. Pagac et al. (1996) combine a 2D occupancy grid with the Dempster–Shafer theory of evidence to create an environment representation for autonomous driving. Wolf and Sukhatme (2004) use a similar approach for a mobile robot, utilizing two grids to determine static and dynamic parts of a scene. Azim and Aycard (2012) also subdivide the environment into static and dynamic elements, utilizing conflict search on an occupancy grid based on the Octomap framework.

3. CHANGE DETECTION USING OCCUPANCY GRIDS WITH DYNAMIC RESOLUTION

3.1 Octrees with fixed and dynamic resolution

Occupancy grids like the one used by Octomap are usually generated with a fixed octree depth (Hornung et al., 2013). The major disadvantage here is that this process is computationally expensive. Most of the time the resolution is finer then required, especially with regard to free space. In an octree with dynamic resolution such as the one used in our previous work (Gehrung et al., 2018), a voxel is only refined if a conflict between free and occupied space is detected. The decision regarding refinement is based on the ratio between the rays traversing and ending in a voxel. This process is computationally cheaper, since ray casting is only applied whenever a voxel is divided.

Since ray casting is still a computationally expensive task, shallow octrees are to be preferred. One way to achieve these is to stop refining at a given depth and then enhance each voxel with an additional representation of free and occupied space. This is not only faster, it also helps preventing artifacts such as the one described in Figure 1. These appear if an area is only little observed. An additional local occupancy representation provides the means necessary to prevent these artifacts.

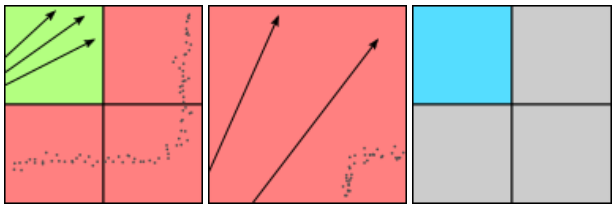


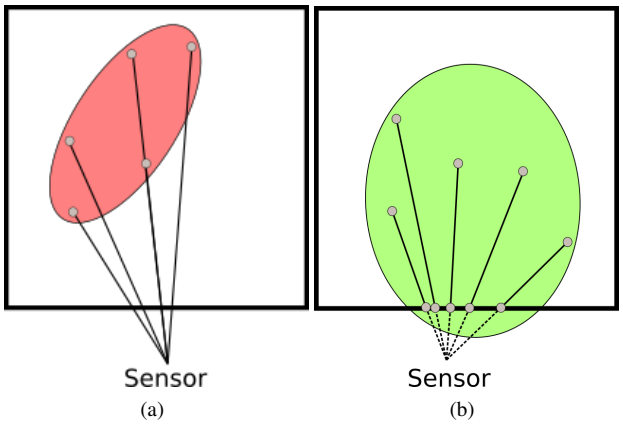
Figure 1. (a) A quadtree with free (green) and occupied (red) cells. (b) The same area, after parts of the surface have been removed. Only a few rays traverse the cell, therefore it is falsely classified as occupied. (c) A quadtree representing changed (blue) and unchanged (grey) areas, showing that all cells are classified incorrectly.

3.2 Local Spatial Analysis using 3D Gaussians

By adding a complementary local occupancy representation, additional conclusions about the encompassed space can be drawn.

Therefore we call this enhancement to classical octree comparison *Local Spatial Analysis*. A suitable data representation has to fulfill a few requirements. It should be cheap to compute and must be able to describe both free and occupied space in an adequate manner. It doesn't have to precisely describe the form of the underlying data, a rough representation is considered to be enough. Furthermore, the representation should be able to provide information about when to refine a voxel. Refinement is required whenever the encompassed space is either too large or to crudely approximated. Last but not least, an instance of the representation must provide the capability to quantify the degree of intersection with other instances in order to detect conflicts between free and occupied space.

Data representation. A representation that fits all these requirements is the Gaussian distribution. It is computationally cheap, since only the mean vector and covariance matrix need to be calculated. Determining a Gaussian kernel that covers a given set of surface points is straightforward. The approximation of free space is achieved by truncating each ray at the voxel boundary and generating both mean and covariance matrix utilizing both the start- and endpoints of the truncated rays (cf. Figure 2).

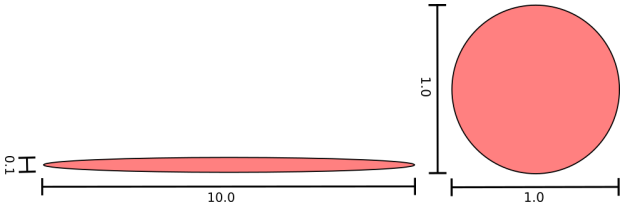


(a) (b)

Figure 2. Calculating mean and covariance for (a) occupied space over surface points and (b) traversed space over the start- and endpoints of the rays truncated at the voxel boundaries.

A mixture of Gaussians would achieve a better approximation of both free and occupied space, but the computational costs required to estimate the appropriate number of kernels via trial-and-error outweigh the possible benefits. A single Gaussian kernel is considered to be exact enough since free space usually encompasses an area without distinct borders. In case of occupied space, which is usually a surface, a very long but flat kernel is sufficient to approximate it. A Gaussian kernel is allowed to leak probability mass into space occupied by neighboring voxels, since only the part of a kernel that is within the voxel under consideration is considered for Local Spatial Analysis.

Refinement. A Gaussian kernel that describes free space is expected to have a large size, whereas a kernel approximating occupied space is usually small and flat. Refinement is only required if the state of the Gaussian deviates from these cases. To determine the expanse of a Gaussian distribution, *principal component analysis (PCA)* can be applied. However, this is computationally expensive to calculate. Another way is the so called *generalized variance (GV)*. The generalized variance of a three-dimensional vector of a random variable X is defined as the determinant of its covariance matrix $|\mathbf{\Sigma}|$ (Sengupta, 2004). Therefore, it is also the product of the eigenvalues. It can be used instead of the PCA, since it provides the product of the eigenvalues of the covariance matrix without explicitly calculating the latter. The higher the GV, the further the measurements are scattered. The multiplication of eigenvalues leads to ambiguities, as illustrated in Figure 3. However, this is a benefit for the task at hand, since both examples match the expected forms of free and occupied space mentioned above.



(a) (b)

Figure 3. Both kernels have the same generalized variance, since the product of their eigenvalues equals 1.

Intersection. In order to calculate the degree of intersection between two Gaussian kernels, the *Hellinger distance* has been chosen (Le Cam and Lo Yang, 2000). This metric is the probability analog of the Euclidean distance. It is considered to be more reliable than the Mahalanobis distance, since this tends to have a value of zero whenever the mean vectors of both Gaussians are similar, even if the standard deviations are different. The Hellinger distance is defined as

$$H^2(P,Q) = 1 - \frac{|\mathbf{\Sigma}_1|^{\frac{1}{4}}|\mathbf{\Sigma}_2|^{\frac{1}{4}}}{|\frac{1}{2}\mathbf{\Sigma}_1+\frac{1}{2}\mathbf{\Sigma}_2|^{\frac{1}{2}}} \exp\left\{-\frac{1}{8}\mathbf{u}^T(\frac{1}{2}\mathbf{\Sigma}_1+\frac{1}{2}\mathbf{\Sigma}_2)^{-1}\mathbf{u}\right\}, \quad (1)$$

where $\mathbf{u} = \mu_1 - \mu_2$. The range of the metric is within the interval $[0, 1]$. The higher the value, the larger is the difference between both Gaussians. In order to measure the **similarity between Gaussians**, we use the logarithm of the Hellinger Distance:

$$S = -\log\left(H^2(P,Q)\right) \quad (2)$$

This allows the easier determination of threshold values as the relevant part of the interval is scaled up.

3.3 Creating a Gaussian Occupancy Octree

The dynamic resolution octree enhanced with the data representation described in the previous section is referred to as a *Gaussian Occupancy Octree*. In order to be able to represent even large areas without the need to rebase the octree, the world is structured into 3D tiles of equal size. Ray casting is used to distribute the range measurements (which are also referred to as rays) to their respective tiles. If a tile contains at least a single ray, iterative refinement of the tile's octree is executed (cf. Algorithm 1).

For each octree voxel, the Gaussian kernels representing free and occupied space are calculated. The covariance matrix requires at least 3 independent data points, but more are recommended in order to increase the likelihood for their independence. The threshold t_r denotes the minimum number of rays required. For each kernel, the generalized variance is calculated. If it is larger than a predefined threshold, then the voxel is subdivided. There are two different thresholds t_f and t_o for free and occupied space.

Algorithm 1: Generation of the Gaussian Occupancy Octree.

Data: List of rays $R = \{r_1, \ldots, r_n\}$
Current depth d.
Result: Current voxel with subtree V.

Function `generate(`R, d`)`**:**
- $V \longleftarrow \emptyset$
- **if** $d > d_{max}$ **then**
  - **return** V
- **else if** $|R| >= t_r$ **then**
  - $g_f \longleftarrow gaussianFree(R)$
  - $g_o \longleftarrow gaussianOccupied(R)$
  - **if** $genVar(g_f) >= t_f \vee genVar(g_o) >= t_o$ **then**
    - **for** $i \in \{1, \ldots, 8\}$ **do**
      - $S \longleftarrow rayS ubsetForVoxel(R)$
      - $V_i \longleftarrow generate(S, d + 1)$
- **return** V

This is to ensure that free space is represented by large Gaussians whereas surfaces are approximated by small Gaussians.

To divide a voxel, all rays are distributed to its child voxels. This is done by executing a ray-box intersection test between each ray and the bounding boxes of each possible child. If there is an intersection, then the ray is added to the corresponding child voxel. Non-existing child voxels are generated and the process is repeated recursively. The procedure is terminated once both generalized variances are smaller than their thresholds or a maximum octree depth d_{max} is exceeded.

3.4 Encoding Changes using Delta Octrees

Change detection is executed by comparing the Gaussian Occupancy Octrees of two epochs m and n. The changes are encoded using another data structure we call a *Delta Octree*. This is also an octree, similar in structure to a concatenation of both Gaussian Occupancy Octrees, but with a label for each voxel that describes the state change between both epochs (cf. Figure 4). Valid state changes are *appeared*, *disappeared*, *unchanged*, *known location not measured* and *new location measured*.

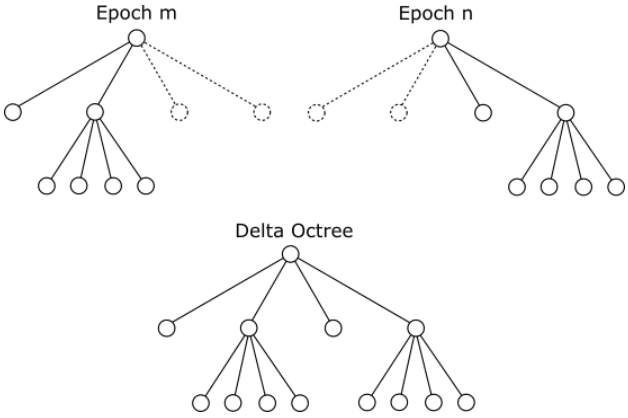


Figure 4. An example for a Delta Octree (bottom), derived from two Gaussian Occupancy Octrees (top left and right).

The generation of a Delta Octree from Gaussian Occupancy Octrees with different spatial resolutions can be seen in Algorithm 2. Both octrees need to occupy the same space and therefore have the same bounding box. Traversal is done recursively and in a depth-first manner in order to derive the Delta Octree's structure that reflects both octrees. In case both octrees have an identical structure, then the Delta Octree's structure is also identical. As soon as one octree represents an area with a higher resolution than the other one, the structures of both octrees are different. In this case, the node with the highest possible resolution is chosen. This leads to the following rule for selecting child nodes. If the depth of the current node in epoch m is less than the depth of the current node in epoch n, then use the current node of epoch m. Otherwise continue with the respective child node, if it exists. If is don't exist, continue with the current node of epoch m. The same is done to select the respective child node from the octree representing the other epoch.

Changes between two voxels (of equal or different size) are determined using the following case differentiation:

- A voxel has occupied space in the first epoch m, but not in the second epoch n. If there is no free space in the second epoch (e.g. due to occlusion), then the state *known location not measured* is assigned to the Delta Octree's corresponding voxel. If there is free space and the intersection between occupied space in m and free space in n is less than a similarity threshold t_s, then the state *disappeared* is assigned.
- A voxel has occupied space in the second epoch n, but not in the first epoch m. If there is no free space in epoch m, the state *new location measured* is assigned. Otherwise, the intersection between occupied space in n and free space in m is determined and checked against the similarity threshold t_s. If the check is successful, than the state *appeared* is assigned, otherwise it is considered to be another case of *new location measured*.
- If none of the above cases occurs, no change is assumed and the state *unchanged* is assigned.

The intersection between Gaussians is determined as shown in Equation 2. In addition, there are two minor special cases. If a 3D tile exists in epoch m, but not in epoch n, then it is considered to be previously known space which has not been measured again. In the opposite case, if a tile does not exist in epoch m but in epoch n, it is considered to be a newly observed location. In both cases, the Delta Octree consists of a single voxel the size of a tile, with either the state *known location not measured* or the state *new location measured*.

3.5 Visualization using Heatmaps

In order to allow a human observer to easily interpret the results, we have chosen heatmaps. These can considered to be a kind of two-dimensional histogram, since they accumulate the intensity of information along the height-axis. To derive a two-dimensional heatmap from a Delta Octree, it is of advantage to interpret the latter as a function $f\colon \mathbb{R}^3 \rightarrow \{l_1, \ldots, l_n\}$ that maps a point in space onto a label l_i. For each label l_i, another function g_i can be derived that returns 1 for each point in space that has label l_i and 0 otherwise. For each of these functions, a heatmap may be generated. The heatmap of a function is derived by sampling it in a $k \times k$ grid. At a sample's position, all voxels in the Delta Octree along the z-axis are determined and a sum of their values is assigned to the corresponding cell in the heatmap. Afterwards, the heatmap is normalized to be in the interval $[0, 1]$.

Algorithm 2: Recursive generation of the Delta Octree $node_{delta}$ from two occupancy octrees $node_m$ and $node_n$.

Data: Gaussian Occupancy Octrees $node_m$ and $node_n$.
Result: Delta octree $node_{delta}$.

```
Function construct(node_m, node_n):
    for i ← 0 to 7 do
        /* Select child of node_m.                    */
        if getDepth(node_m) < getDepth(node_n) then
            next_m ← node_m
        else
            if hasChild(node_m, i) then
                next_m ← getChild(node_m, i)
            else
                next_m ← node_m
        /* Select child of node_n.                    */
        ...
        /* Recurse.                                   */
        addChild(node_delta, i) ← construct(next_m, next_n)
    /* Determine label of node_delta.                 */
    ...
```

4. EVALUATION

4.1 Experimental Setup

The evaluation is based on a dataset showing the demolition of multiple buildings in Ettlingen, Germany. It consists of 15 epochs recorded from mid-april to mid-june 2018 that show the buildings in different stages of demolition (cf. Figure 5). The dataset has been recorded by the measurement vehicle MODISSA of the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) (Borgmann et al., 2018). The point clouds were recorded using two Velodyne HDL-64E LiDAR sensors mounted at an angle of 25 ° on the vehicles front roof. Navigational data was recorded using an Applanix POS LV navigation system, an inertial measuring unit and a distance rotary encoder. The navigation data has been postprocessed in order to increase accuracy. A LiDAR-SLAM based fine-registration approach has been used to further improve the registration within epochs and between all epochs and the first one.

The epochs 1 and 5 have been labeled in order to have ground truth for change detection. Labeling was done at point level using a hand-labeling tool implemented by the authors. The labels include surface points removed from epoch 1 and surface points added to epoch 5. Figure 5(a) and 5(b) show that between both epochs, the front of the northern building, some of its balconies as well as a shed in its south have been deconstructed. Part of the roof is missing. Also, a large part of the western building, a crane on its east side and some debris have been removed. Another crane appeared, an excavator and two heaps of debris.

4.2 Qualitative Evaluation

In order to evaluate the properties of the proposed approach, a Delta Octree between both epochs is examined by a human observer. This case study consists of two parts. The first one is considered with the detection of changes and therefore investigates voxels labeled as *appeared* or *disappeared*. The second part draws conclusions regarding the exploration of space by examining voxels labeled as *known location not measured* and *new location measured*. In addition to the case study, heatmaps representing changes as well as the state of spatial exploration are discussed.

4.3 Quantitative Evaluation

To supplement the case study, a quantitative evaluation based on the ground truth mentioned above has been carried out. Several methods for comparison were examined before finding an appropriate one. The reason for this is that in order to evaluate the results of this work, it is required to compare a volumetric representation with surface-based labels. This is aggravated by the fact that the volumes are not uniform and therefore may summarize multiple change-related events. In addition, the surface representation does not contain any information regarding the exploratory state of the area.

After careful considerations, the following approach has been chosen. The actual state of every voxel is determined based on the subset of labeled points that are located within the voxel. Per voxel, two counters represent appeared and disappeared surfaces. Each labeled point located within a voxel increments the corresponding counter. The actual state of a voxel is then determined based on these counters. If both are zero, the voxel is assumed to be *unchanged*. If the first counter is higher than the second one, an actual state of *appeared* is assumed, otherwise an actual state of *disappeared*. Both counters may be of equal value, but this is considered unlikely due to the large amount of labeled points per square meter.

The predicted state is provided by the Delta Octree. Both the actual and predicted state are then used to generate a confusion matrix. Two confusion matrices created from Delta Octrees with a maximum resolution of 1 m and 2 m are discussed.

4.4 Runtime

The evaluation is concluded by a short discussion of the runtime. Therefore, the approaches for generating the Gaussian Occupancy Octree and the Delta Octree are investigated.

5. RESULTS AND DISCUSSION

5.1 Results of the Qualitative Evaluation

Figure 6(a) shows a heatmap encoding all changes that occurred between both epochs. It does not distinguish between appearances and disappearances. High degrees of change are marked in red, medium ones in green and a low ones in blue. The circles are annotations made by a human observer marking areas of major changes. Red circles imply disappearances and blue circles appearances. A binarized version of the heatmap in which only the areas with the most changes remain can be seen in Figure 6(b).

Despite the noise present in the heatmap, all major changes were detected. The deconstructed part of the northern building has been recognized correctly. Also parts of its roof, the balcony and the shed on its southern side. The area around both the appeared excavator and crane have a higher intensity than the little background noise present. Most parts of the removed western building part have been recognized as changed, except the southeastern section. As can be clearly seen in Figure 6(d) at location marker *N1*, this part of the building was previously obscured and therefore has only be observed after the demolition.

(a) (b)

Figure 5. Overview over the point clouds of (a) epoch 1 and (b) epoch 5. Red circles imply disappearances, blue circles appearances.

The heatmap in 6(c) shows the locations observed in epoch 1, but not again in epoch 5. The area at location marker *M1* has remained unobserved because the path of the measurement vehicle in epoch 5 deviates strongly from its previous path in epoch 1. Therefore, the removed crane has not been detected as a change. The same applies to the area at marker *M3*, which is obscured in epoch 5 by a mound. Location marker *M2* refers to free space above the building. All this can be deduced from a 3D visualization of the Gaussian Occupancy Octrees of both epochs.

Figure 6(d) shows the heatmap of space not observed in epoch 1, but in epoch 5. Our algorithm detected the appearance of occupied space at the location markers *N2*, *N4* and *N5*, but the intersection between the free space of epoch 1 and the occupied space of epoch 5 exceeded the similarity threshold t_s. The algorithm had deduced that the area has not been observed completely before and marked it as newly observed, which can be considered a valid assessment of the situation. The areas marked as *N2* and *N6* have previously been unobserved and are now visible as both the western building part and the shed have been removed.

5.2 Results of the Quantitative Evaluation

Table 1 shows the confusion matrix for the Delta Octree with a 2 m resolution. A pronounced diagonal shows that the classifier tends to correctly classify changes. A closer look at the precision and recall of the individual classes imply that our approach is biased towards the class *appeared*, since this class has a high recall, but a low precision. This effect is also the main reason why the overall accuracy of all classes is only 51.5 %.

We assume that the reason for this is the way we generate Gaussian kernels. By definition, at least 10 surface points must be available per voxel in order to generate a kernel. Especially in case of smaller voxels it can happen that not enough measurements are available. Epoch 5 contains less surfaces and therefore larger voxels, since these are mostly representing free space. Because the voxels are bigger, they are also more likely to contain the required amount of surface points. For this reason, a Gaussian kernel may exist in epoch 5 but not in epoch 1, causing the algorithm to conclude the appearance of occupied space.

Comparing Table 1 and 2 suggests that the overall accuracy decreases once the maximum resolution increases. A closer analysis

		Actual			
		App.	Dis.	Un.	Total
Predicted	App.	68	5	632	9.7 %
	Dis.	2	294	147	66.4 %
	Un.	22	40	540	89.7 %
	Total	73.9 %	86.7 %	40.9 %	51.5 %

Table 1. The confusion matrix determined by evaluating the Delta Octree with a 2 m resolution.

		Actual			
		App.	Dis.	Un.	Total
Predicted	App.	206	11	2292	8.2 %
	Dis.	6	886	739	54.3 %
	Un.	98	148	1023	80.6 %
	Total	66.5 %	84.8 %	25.2 %	39.1 %

Table 2. The confusion matrix determined by evaluating the Delta Octree with a 1 m resolution.

leads to the conclusion that the Hellinger distance applied to determine the overlap of Gaussians rapidly degrades towards zero whenever two Gaussian kernels with very different variances are compared. An example for this can be seen in Figure 7. This situation occurs whenever a low-resolution voxel is compared to a high-resolution one. The effect occurs more often when the resolution is increased, since this also means more high-resolution voxels. This effect limits the maximum resolution of our approach.

5.3 Discussion of the Runtime

All reported results were generated on a machine with 32 Gigabyte of RAM and an Intel Core i7 processor with 3.5 GHz and 12 cores. The Gaussian Occupancy Octrees for 2 m and 1 m resolution are created within 12 s respectively 35 s per tile and epoch. The generation of the Delta Octree requires less than a second.

6. CONCLUSION AND FUTURE WORK

In this paper, a fast approach to determine both changes and exploratory information using low-resolution octrees has been proposed. Gaussian kernels are used to represent free and occupied

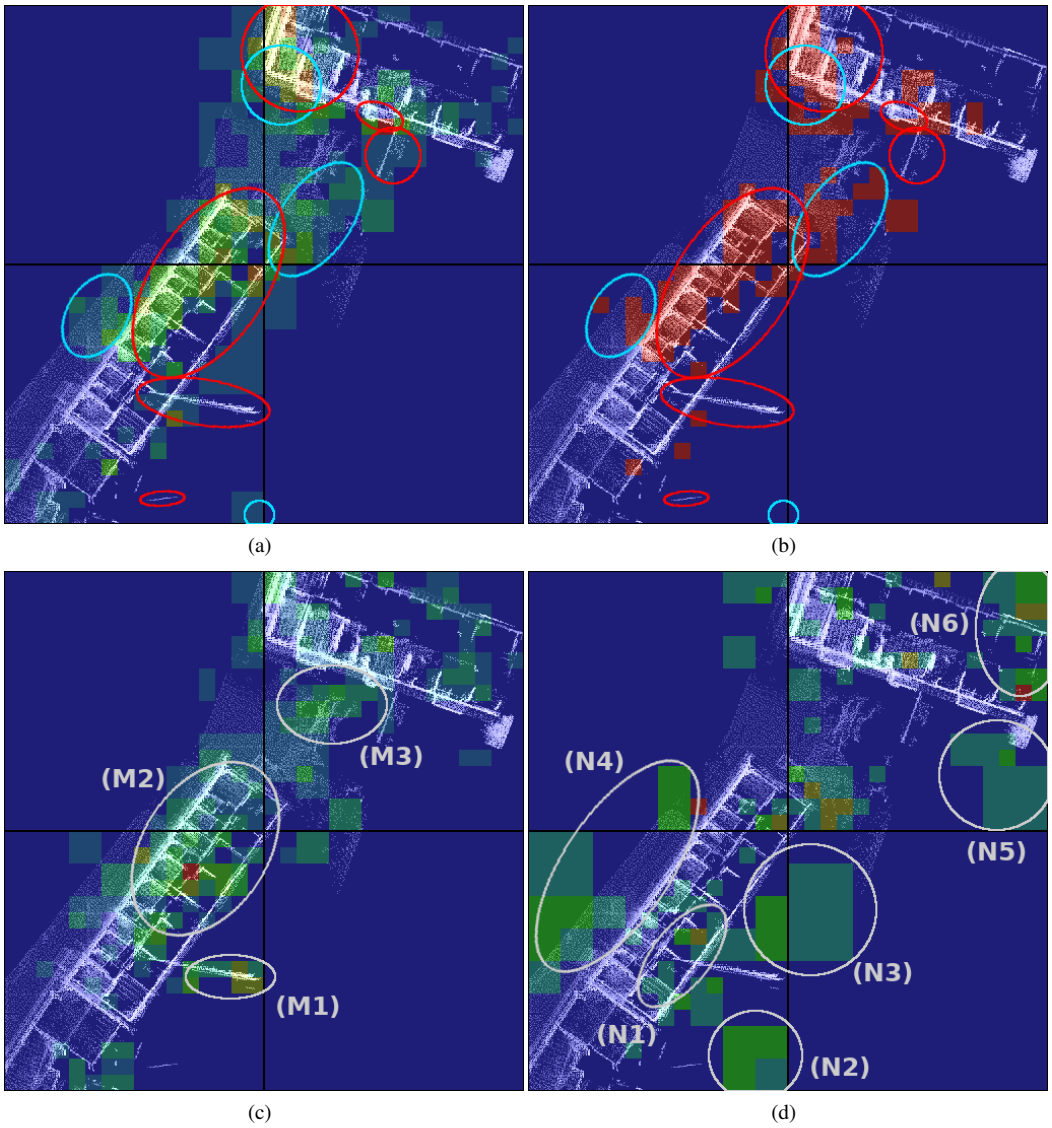


Figure 6. The heatmap in (a) encodes changes regarding both appearance and disappearence, whereas the first ones are annotated with blue circles, the latter one with red ones. A version binarized with a threshold of 0.2 can be seen in (b). The heatmaps in (c) and (d) show missed locations and newly measured locations. All heatmaps are created with a 2 m resolution.

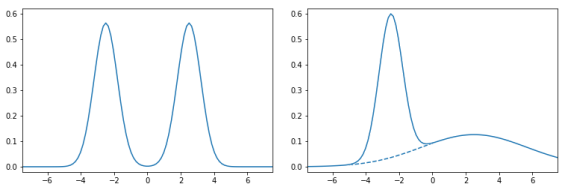


Figure 7. Example for the degradation of the Hellinger distance. The Hellinger distance is 0.99 for both shown Gaussian kernels (a). It degrades to 0.64 once the variance of the second kernels is greatly increased (b).

space in order to further enhance the octree resolution. The advantages of our approach are the short runtime and the ability to make a statement about the re-exploration of space. Both a qualitative and quantitative evaluation show that the approach tends towards correct classification. However, since there is a bias towards the appearance of occupied space, the overall accuracy for an octree with 2 m resolution is only 51.5 %. Also, the maximum spatial resolution is limited by the distance metric used to detect changes. One possible solution that may be explored in future work is it to use measurement rays instead of Gaussians, since these do not require any model assumptions.

References

Aijazi, A. K., Checchin, P. and 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, pp. 7–12.

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

Borgmann, B., Schatz, V., Kieritz, H., Scherer-Klöckling, C., Hebel, M. and Arens, M., 2018. Data processing and recording using a versatile multi-sensor vehicle. Vol. IV-1, pp. 21–28.

Du, S., Zhang, Y., Qin, R., Yang, Z., Zou, Z., Tang, Y. and Fan, C., 2016. Building Change Detection Using Old Aerial Images and New LiDAR Data. *Remote Sensing*.

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

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

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

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

Hebel, M., Arens, M. and 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, pp. 52–64.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C. and Burgard, W., 2013. OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees. *Autonomous Robots* 34(3), pp. 189–206.

Le Cam, L. and Lo Yang, G., 2000. *Contiguity - Hellinger Transform*. Springer.

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

Meagher, D., 1982. Geometric modeling using octree encoding. *Computer Graphics and Image Processing* 19(2), pp. 129–147.

Moravec, H. and Elfes, A., 1985. High resolution maps from wide angle sonar. In: *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, pp. 116 – 121.

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

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

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

Sengupta, A., 2004. *Generalized Variance*. American Cancer Society.

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

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

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

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