Reliable Real-time Change Detection and Mapping for 3D LiDARs

Lorenz Wellhausen Renaud Dubé Abel Gawel Roland Siegwart Cesar Cadena*

***Abstract*— A common scenario in Search and Rescue robotics is to map and patrol a disaster site to assess the situation and plan potential missions of rescue teams. Particular importance has to be given to changes in the environment as these may correspond to critical events like building collapses, movement of objects, etc. This paper presents a change detection pipeline for LiDAR-equipped robots to assist humans in detecting those changes. The local 3D point cloud data is compared to an octree-based occupancy map representation of the environment by computing the Mahalanobis distance to the closest voxel in the map. The thresholded distance is processed by a clustering algorithm to obtain a set of change candidates. Finally, outliers in these sets are filtered using a random forest classifier. Changes are continuously mapped during a sortie based on their classification score and number of occurrences. Changes are reported in real time during robot operation.**

I. Introduction

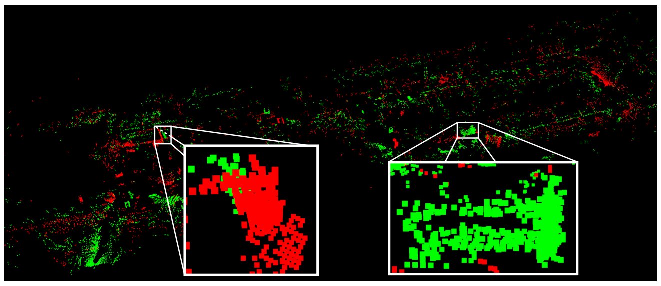
In Search and Rescue (SaR) scenarios, robots can be used to create highly detailed metric maps of the environment which represent potentially large and cluttered disaster areas. Robots can patrol disaster areas and revisit them over several days for surveillance, where detecting changes in the environment is of special interest. These changes can represent moved objects, collapsed structures or environment deformations.

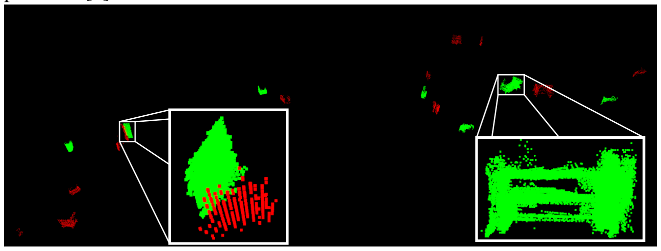
An online change detection algorithm is desirable to assist robot operators or enable autonomous inspection of dynamic parts of the environment. As a research problem this generalizes to detecting changes that occurred in the environment since a reference map was built. Differentiating between actual dynamics and apparent changes caused by sensors noise and data misalignment is the main challenge. Additional challenges are posed by changes in view point and incomplete or erroneous map data. Current approaches either require a high rate of point cloud acquisition or cannot cope with the errors from previously mentioned sources.

The input to our algorithm is 3D point-cloud data stemming from LiDAR sensor readings of a Unmanned Ground Vehicle (UGV). In our approach, we define a measure for change likelihood by computing the Mahalanobis distance between points in the reading point cloud and those in a reference map. A set of change candidates is then obtained by clustering the points based on a distance threshold. These candidates are classified as static or dynamic using a random forest classifier trained on separate data. The output of the classifier is continuously mapped during a sortie. Based on our evaluation on a real-world dataset of an abandoned power plant, this approach shows to be reliable when applied on distorted point clouds and imperfect map data.

To the best of our knowledge, this is the first paper to present LiDAR-based novelty detection using a learning approach to facilitate change detection. Specifically, this paper presents the following contributions:

1) A probabilistic distance metric as a measure for point change likelihood.
2) A real-time change detection system with learning-based novelty classification reporting changes online.
3) An evaluation of the algorithm in urban and SaR


(a) Change detection results using thresholded Hausdorff distance as proposed in [1].


(b) Change detection results using our approach.

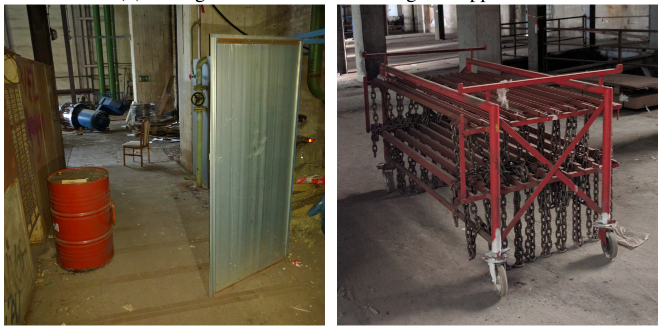

(c) Door and steel cart which where moved to create dynamics highlighted in the images above.

Fig. 1: Change detection results of different approaches for a dataset recorded in the decommissioned Gustav-Knepper power plant. Red points indicate points which disappeared since the last visit. Green points are new points in the scene.

*Authors are with the Autonomous Systems Lab, ETH, Zurich authors@mavt.ethz.ch.

This work was supported by the European Union's Seventh Framework Programme for Research and Technological Development under the TRADR project No. FP7-ICT-609763, and the armasuisse S+T UGV research program.

scenarios.

In the following section we present an overview of the related work in 3D change detection. Our system is presented in detail in Section III, and its performance is validated in real-word scenarios in Section IV. Finally, we give our conclusions in Section V.

II. Related Work

Previous works on novelty detection in geometric data can be separated in two general types of approaches, point or voxel based and cluster based.

Approaches falling in the first category aim at detecting dynamics using individual points or voxels [1–5]. These approaches generally achieve a high level of detail for the detected dynamics. However, outliers are required to be handled explicitly in order to achieve robustness.

Girardeau-Montaut et al. [1] compute the distance between every point in one point cloud to its closest equivalent in another point cloud and extract changes by thresholding the distance. Expressing points in spherical coordinate space is used in [2] and [5] to facilitate quick-raycasting. The algorithms proposed in [3] and [4] use multiple consecutive sensor measurements to deal with outliers and refine the detection, which requires a high rate of point cloud acquisition so that multiple measurements from a similar viewpoint are available. This makes these techniques impractical to deploy on systems where a high measurement rate cannot be achieved, e.g. when assembling 3D point clouds from a slowly rotating 2D LiDAR.

The second type of approaches are those that cluster points and approximate the point distribution with a function. Novelties are detected using these functions and points are later re-associated to obtain point-accurate dynamics estimates [6–8]. Drews et al. [6] represent both reference and reading point cloud as a collection of Gaussian Mixture Model (GMM)s and classify distributions that do not have matching GMMs in both clouds as novelties. In [7] changes are detected based on implicit volume by computing the local density function of the point clouds. Andreasson et al. [8] represent the prior map with a 3D normal distribution transform and compute a probability that reading cloud points are different from this distribution. Because these approaches fit a function to a group of points they have a certain level of implicit robustness to outliers. The downside of this approximation is that information about the expected visibility of map points is lost. As a result, none of these approaches can detect disappearing objects.

An additional mention has to be made of the field of object discovery and segmentation which has been studied extensively [9–14]. However, many of these approaches make assumptions about object shape [9–11] or size [12] which are not guaranteed to hold in unstructured SaR scenarios. An approach proposed by Herbst et al. [13, 14] uses dense RGB-D data of tabletop scenes where the pipeline takes several minutes per frame. This approach is not suitable for real-time robotic applications in unstructured scenarios with no control on the illumination and with sparser LiDAR data to work with.

Our approach is most closely related to [1] with extensions to make it suitable for online operation. We extract change candidates using the distance between individual points in the reading point cloud and reference map, which provides a similar level of detail to approaches like [1–5]. By classifying point clusters, we benefit from the robustness to outliers as in approaches [6–8].

III. Change Detection System

In this section we present our change detection algorithm facilitating online novelty detection for 3D LiDAR-equipped mobile robots.

An overview of the pipeline is shown in Figure 2 and modules are described in the same order as illustrated in the block diagram. The inputs are the current reading point cloud captured by the LiDAR and a reference map. The base representation of this map is an occupancy octree, specifically an OctoMap [15].

A. Reference Map Alignment

In order to initialize our change detection system, the reading point cloud and the reference map need to be aligned. With our current implementation, a general alignment transformation T_{reloc} is found between the current and reference map using the $SegMatch$ [16] algorithm. Every new reading point cloud is further aligned to the reference map in an Iterative Closest Point (ICP) step, with T_{reloc} as an initial guess, to obtain the current alignment transformation T_{align}.

B. Ground Plane Removal

In order to reduce the risk of observing false dynamics caused by the sparse reference map ground plane, we detect and remove every reading point which lies on the ground plane.

As stated in the original OctoMap work [15], configurations where LiDAR sensors are mounted close to the ground level can cause voxels to be falsely marked as unoccupied. This is a result of laser beams hitting the ground at shallow angles, intersecting multiple voxels and marking them as free when that might actually not be the case. Therefore, the reference OctoMap will have many free voxels in the ground plane and if points of the reading point cloud fall in this free space they would appear as dynamics.

Our ground segmentation implementation uses an algorithm proposed by Himmelsbach et al. [17]. Slight changes to the ground plane condition were made for cases where points are spaced far apart in radial direction.

C. Distance Computation

The distance computation between points in the reading point cloud and reference map is the core module of the change detection system. It encompasses two parts, one for detecting disappearing objects and one for detecting appearing objects.

We define the distance $\mathcal{D}^2(\mathbf{p}, S')$ as the squared Mahalanobis Hausdorff distance. It is computed between a point

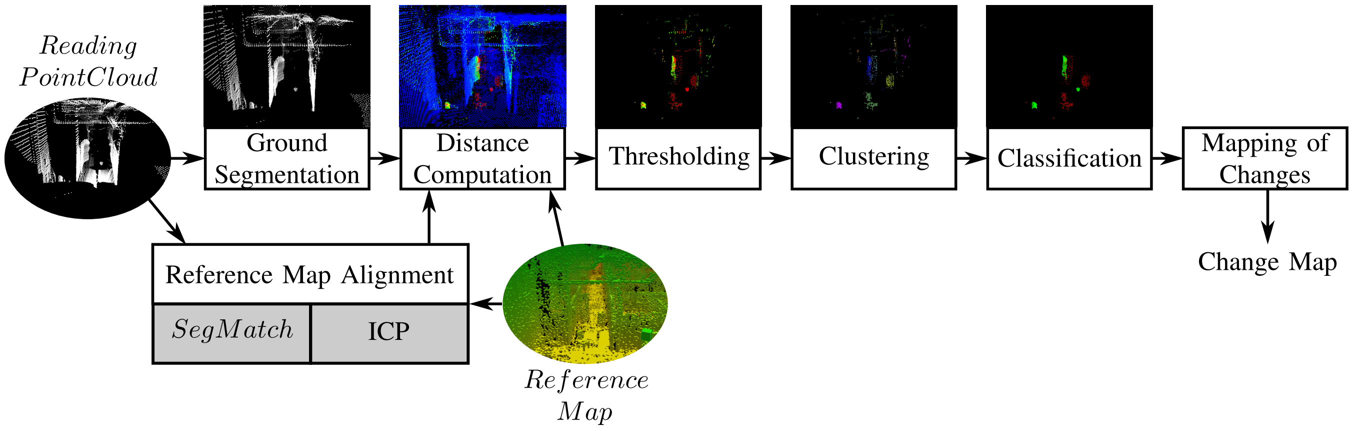


Fig. 2: Block diagram of the proposed change detection pipeline. Inputs to the system are the reading point cloud and a reference map. The output is a map of changes which is reported online. Alignment components with grey background are external to the system.

$\mathbf{p}$ from set S with covariance matrix $\boldsymbol{\Sigma}$ and the closest point $\mathbf{p}'$ in another point set S' as follows:

$$\mathcal{D}^2(\mathbf{p}, S') = \min_{\mathbf{p}' \in S'} \{(\mathbf{p} - \mathbf{p}')^T \boldsymbol{\Sigma}^{-1} (\mathbf{p} - \mathbf{p}')\} \qquad (1)$$

The computed distance is used in Section III-E as a measure for the likelihood that a point belongs to an object or structure which has changed since the environment was first mapped.

Assuming negligible robot pose uncertainty, the reading point cloud covariance $\boldsymbol{\Sigma}$ mainly characterizes the uncertainty in the range measurement as well as uncertainty in beam direction and spread. Hence, when expressed in spherical coordinates, the covariance matrix for all reading points is constant, assuming that radial noise is additive. For this reason and because spherical coordinates simplify ray-casting, all points are converted to spherical coordinates, denoted with the superscript *spher*.

The occupied voxels of the reference OctoMap are extracted as a point cloud and transformed to the sensor coordinate frame as P_{ref}.

1) Disappearing Objects: For detecting objects that disappeared, the distance is computed from every occupied map voxel which is intersected by a laser beam to the closest point in the reading point cloud P_{read}. The pseudocode for the distance point cloud computation is shown in Algorithm 1. In this case, the inputs to the algorithm are $P_A := P_{ref}$, $P_B := P_{read}$ and $P_B^{spher} := P_{read}^{spher}$. The variable $ObsSpace_B$ expresses information about the observed space obtained from ray-casting.

The output of Algorithm 1 (P_{dist}), in this case corresponds to $P_{disappear}$, which contains all observed map voxel centerpoints and their distance to the closest reading point.

2) Appearing Objects: To detect appearing objects, the distance from all reading cloud points is computed to the closest occupied map voxel. Distances are computed analogous to disappearing points such that Algorithm 1 is also valid for appearing points. In this case, the inputs to the algorithm are $P_A := P_{read}$, $P_B := P_{ref}$, $P_B^{spher} := P_{ref}^{spher}$. $ObsSpace_B$, the information on space observed in the reference map can be obtained from the reference OctoMap. The output of the algorithm P_{dist}, now corresponding to P_{appear}, contains all reading cloud points in observed space and their distance to the closest map voxel.

Algorithm 1 Distance computation.

```
Require: $P_A, P_B, P_B^{spher}, ObsSpace_B$
  $P_{dist} = \emptyset$
  for $\mathbf{p}_i \in \{P_A \cap P_B\}$ do
    if $ObsSpace_B.isObserved(\mathbf{p}_i^{spher})$ then
      $neighbors_i \leftarrow knn$ search of $\mathbf{p}_i^{spher}$ in $P_B^{spher}$
      $\mathcal{D}^2(\mathbf{p}_i, P_B) \leftarrow$ GETDIST($\mathbf{p}_i^{spher}, neighbors_i$)
    else
      $\mathcal{D}^2(\mathbf{p}_i, P_B) \leftarrow 0$ ▷ Unobserved means no change.
    end if
    $P_{dist} \leftarrow P_{dist} \cup \{\mathbf{p}_i, \mathcal{D}^2(\mathbf{p}_i, P_B)\}$
  end for
```

Algorithm 2 Compute Mahalanobis distance from point to voxel boundary.

```
Require: $resolution_{octomap}, \boldsymbol{\Sigma}$
  function GETDIST($\mathbf{p}_{spher}, neighbors$)
    $\mathbf{p}_{neigh} = \underset{\mathbf{n} \in neighbors}{\arg\max} \ (\mathbf{n} - \mathbf{p}_{spher})^T \boldsymbol{\Sigma}^{-1} (\mathbf{n} - \mathbf{p}_{spher})$
    $\mathbf{p}_{diff} = \mathbf{p}_{spher} - \mathbf{p}_{neigh}$
    $r_p = \mathbf{p}_{spher}[r]$                ▷ Range value of $\mathbf{p}_{spher}$
    $d_{octo} = {}^{resolution_{octomap}}/_{2}$
    $\mathbf{p}_{corr} = \begin{bmatrix} {}^{d_{octo}}/_{r_p} \\ {}^{d_{octo}}/_{r_p} \\ d_{octo} \end{bmatrix}$
    $\mathbf{p}_{dist} = \mathbf{p}_{diff}(1 - \frac{|\mathbf{p}_{diff}|}{||\mathbf{p}_{diff}||} \cdot \frac{\mathbf{p}_{corr}}{||\mathbf{p}_{diff}||})$
    return $\mathbf{p}_{dist}^T \boldsymbol{\Sigma}^{-1} \mathbf{p}_{dist}$
  end function
```

3) Voxelization Error Correction: The distance error introduced by the voxelization of the reference map is corrected as described in Algorithm 2. It expresses the voxel boundary in spherical coordinates and subtracts the distance from centerpoints to boundaries from the total distance.

D. Clustering

To remove points that have a low probability of having changed, a threshold is applied to P_{appear} and $P_{disappear}$, generating two sets of change candidate points. They are clustered to identify points which belong to the same object such that we can use the information on all cluster points to later classify them as static or dynamic.

We use the HDBSCAN algorithm [18] which uses the distance to the k-nearest neighbor as a local density measure. It extracts clusters of locally constant density based on the stability of clusters and is robust to outliers.

E. Classification

Despite the excellent performance of HDBSCAN for clustering change candidates, it does not perform any classification step which requires a dedicated classification algorithm for point clusters.

A simple way to determine whether a cluster is dynamic or static is to make a decision based on a threshold on the point distances. A representative distance $\mathcal{D}^2_{cluster}$ is computed from all cluster points. Now, a simple threshold $\mathcal{D}^2_{classify}$ can be applied such that all clusters for which $\mathcal{D}^2_{cluster} > \mathcal{D}^2_{classify}$ are considered dynamic, other clusters are assumed to be false candidates. The following ways to compute $\mathcal{D}^2_{cluster}$ have been evaluated.

T_{min}	Minimum distance of any cluster point.
T_{max}	Maximum distance of any cluster point.
T_{med}	Median distance of all cluster points.

We illustrate the ROC curves of these approaches in Figure 6 which shows that T_{med} is superior to T_{min} and T_{max} for all possible $\mathcal{D}^2_{classify}$. Because the threshold classification approach does not satisfy our desired error rate, further classification algorithms were explored.

We extract 32 features from each cluster based on point distance distribution and cluster appearance. The five most significant features, according to a variable importance analysis are: median and maximum point distance, vertical point variance, spherical coordinate range variance and the highest value bin of a histogram of point distances.

We evaluated AdaBoost [19], random forest [20], support vector machines [21] as well as linear discriminant analysis [22] classifiers of which we found the random forest classifier to be consistently best performing and most robust to parameter changes. The random forest algorithm classifies samples using the set of features discussed above. During training, this model grows a multitude of decision trees, each one trained with boostrap samples from a training set. In order to reduce the correlation between the trees, a different subset of features is used to find the best split at every node of the decision tree.

The output of the algorithm is a scored label for every cluster of points indicating whether it is static or dynamic.

F. Mapping of Changes

Until this point in the pipeline we discussed how to detect dynamics by comparing only the current reading point cloud to the reference map.

Temporal filtering over the changes is performed by continuously mapping them. We require a cluster of the same type, appearing or disappearing, to be observed $n_{overlap}$ times at the same position to be considered an inlier. Since the viewpoint changes between observations, we require a measure that defines when clusters coincide. We use the intersection of the aligned bounding boxes of clusters to track clusters over multiple reading point clouds.

Let $BBox_i$ be the aligned bounding box around cluster C_i. The classification score of cluster C_i is $0 \leq Score_i \leq 1$. The volume of a bounding box and the volume of the intersection of two bounding boxes are defined as

$$V_i = Volume(BBox_i) \tag{2}$$

$$V_{ij} = Volume(BBox_i \cap BBox_j) \tag{3}$$

We label cluster C_i as a support cluster if $n_{overlap}$ other clusters C_j exist for which the overlap ratio is larger than a threshold based on the classification score.

$$\frac{V_{ij}}{V_i} \overset{?}{>} 1 - Score_i \tag{4}$$

All points which are part of clusters labelled as support are considered changes. Additionally, all points of clusters that themselves overlap a support cluster more than the threshold of Equation (4) are considered changes. These overlapping segments are added to capture fringe observations and obtain a more complete point cloud of the changed object.

IV. Experiments

We evaluate our change detection system on two real-world datasets, gathered in different environments, i.e., an urban scene and an industrial facility.

A. Datasets

Both datasets were recorded using a highly agile UGV equipped with encoders, IMU and sweeping 2D LiDAR scanner, producing full 3D scans every 3 seconds.

1) Clausiusstrasse: The first dataset was recorded in Clausiusstrasse in Zürich, Switzerland, at different times of the day. Dynamic objects present in the scene and recorded in the reference map are parked cars, as well as pedestrians standing still for the duration of multiple scan acquisitions. A $100m$ long stretch of the road was mapped to create the reference OctoMap. It results in 370 000 occupied voxels with a resolution of $7.5cm$.

2) Power Plant: The second dataset was recorded within the "Long-Term Human-Robot Teaming for Robots Assisted Disaster Response" (TRADR) project in the abandoned Gustav Knepper power plant in Dortmund, Germany, which is illustrated in Figure 3. This environment is akin to a cluttered industrial disaster scenario. Dynamics are several objects of different sizes that were manually moved between sorties. A sample of different objects used as dynamics is pictured in Figure 1c. The power plant has a footprint of about $100m \times 25m$ with two different accessible levels. The reference map of the lower level consists of approximately 600 000 occupied voxels, the map of the upper level has about 400 000 occupied voxels, both at a resolution of $7.5cm$.

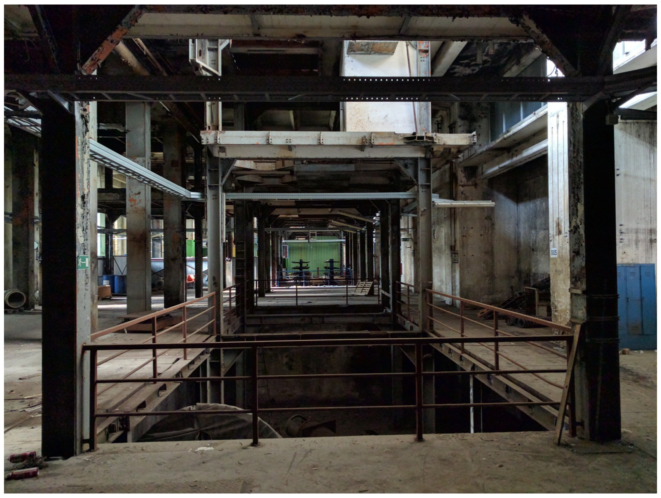


Fig. 3: Picture of lower floor inside Gustav-Knepper power plant where data was collected in order to evaluate our change detection system.

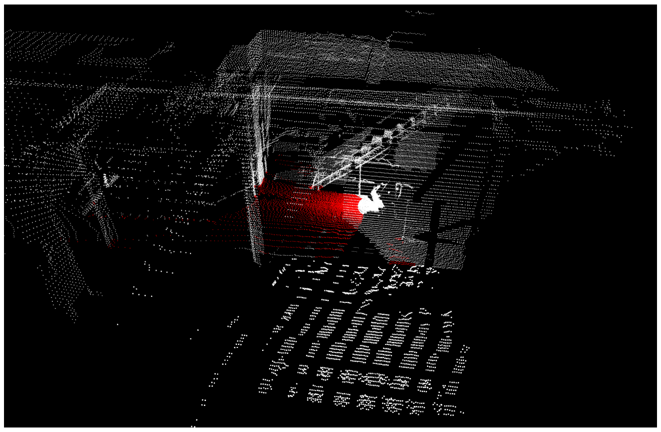
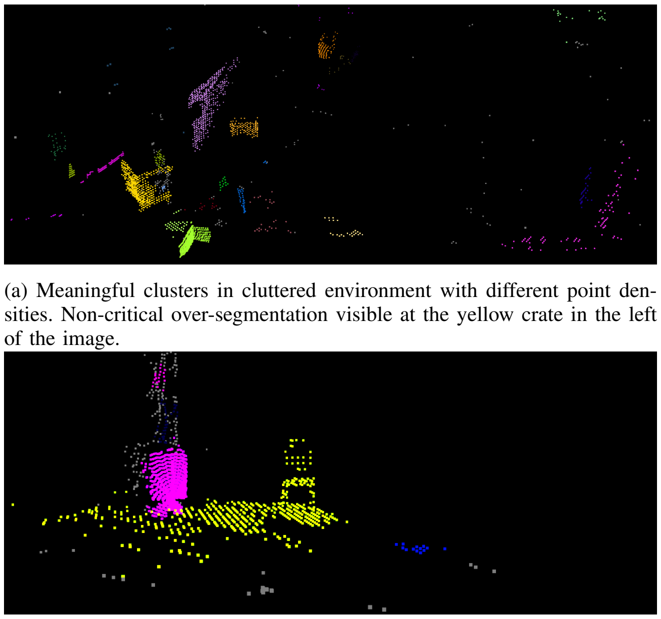


Fig. 4: Robot standing on steel grate with horizontal plane visible below ground plane. Ground points are colored in red.



(a) Meaningful clusters in cluttered environment with different point densities. Non-critical over-segmentation visible at the yellow crate in the left of the image.

(b) Clustering algorithm under-segments when disappearing chair sits on top of a false ground ray cast.

Fig. 5: Clustering results using HDBSCAN algorithm. Points with the same color belong to the same cluster.

B. Evaluation

This section evaluates the performance of every component of our pipeline on the mentioned datasets. Performance metrics used for each one are stated in the respective sections.

The performance of ground plane removal is very robust with minor outliers, even in difficult environments as illustrated in Figure 4. Outliers are caused by high point density regions close to the ground plane or under-sampled areas with too few points for reliable estimation.

1) Clustering: In the clustering step over-segmentation is tolerable as long as cluster sizes remain big enough such that they overlap during change mapping, as we aim to distinguish dynamic from static elements and not necessarily capture complete objects within a single cluster. Therefore, we evaluate clustering performance based on the rate of under-segmented points.

As can be seen in Figure 5a, points are generally meaningfully grouped, even in low density regions, with some over-segmentations occurring due to non-constant density of the occupied reference map voxels. Under-segmentations only occur in rare edge cases where disappearing objects sit on top of outlier change candidates caused by erroneous ray casts as illustrated in Figure 5b.

Erroneous ray casts occur irregularly and therefore the same under segmentation rarely occurs multiple times. This means that false clusters generated by this error are mostly removed during change mapping due to the temporal filtering.

2) Classification: Classification training and test data was gathered by manually labelling all change candidates after clustering. We apply a conservative distance threshold such that no change points are lost during thresholding and clustering. This was done for two sorties in the power plant dataset, one upstairs, one downstairs, and one sortie for the Clausiusstrasse dataset. This resulted in a total number of 5816 labelled data samples, 88% of them static, 12% dynamic, where each sample is a cluster. Samples from the downstairs power plant sortie were used for training, which contains 57% of all labelled samples. The samples from the upstairs power plant sortie and the Clausiusstrasse sortie were used as test data. It should be noted that, while the power plant sorties were recorded in the same general environment, the samples are from different floors of the power plant and do not contain any of the same objects. The Clausiusstrasse data is from an environment that was not used for training the classifier and is therefore an indicator for how well the classifier generalizes.

Classification results for the random forest classifier compared to the threshold based approaches are shown in Figure 6. The indicated point on the curve is the operating point, determined by the location of the maximal Youden's index, which lies at a classification score threshold of 0.17. Precision and recall for the operating point are 0.872 and

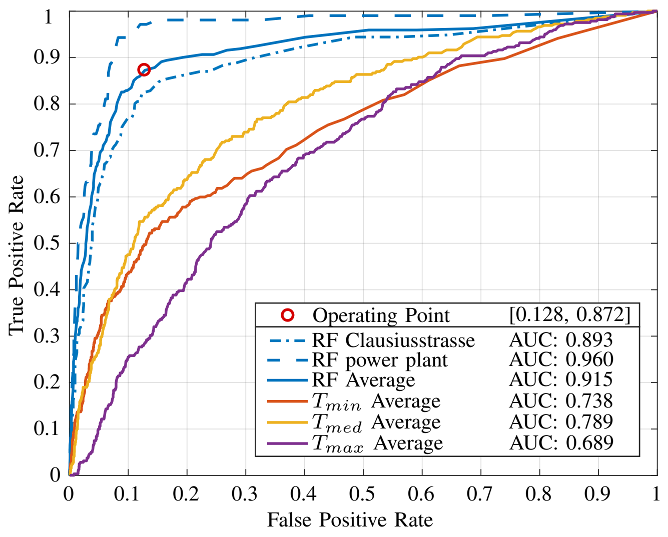


Fig. 6: ROC plot showing error rates for threshold based approaches compared to random forest classifier.

0.873 respectively. The random forest classifier significantly outperforms all threshold based approaches.

One examples of clusters that were most frequently classified as static are fine structures which are smaller in size than the voxel resolution in at least one dimension (e.g. steel grates, pipes). These clusters where characterized by low point density with one dominant direction of point variance. Another example are erroneous ray casts in the ground plane. These clusters typically have a large median distance value and low variance in vertical direction.

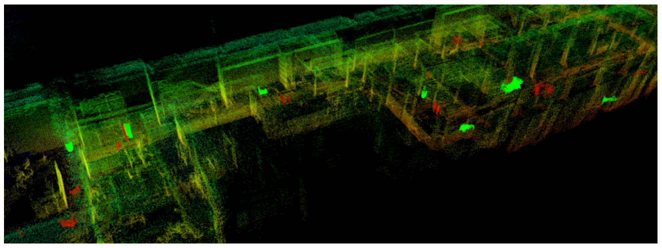
The most common error source present in the evaluated datasets were erroneous ray casts which produce change candidates that appear identical to true dynamics. This issue is exacerbated by our sensor platform, which assembles a 3D point cloud from a sweeping 2D LiDAR and should be less prevalent on other systems. Performing ray casting based on individual 2D laser scans instead of the 3D assembled cloud could alleviate this issue and should be implemented when this system is to be deployed on a robot with such a sensor system.

3) Mapping of Changes: Change mapping performance was evaluated manually, based on the number of correctly mapped clusters.

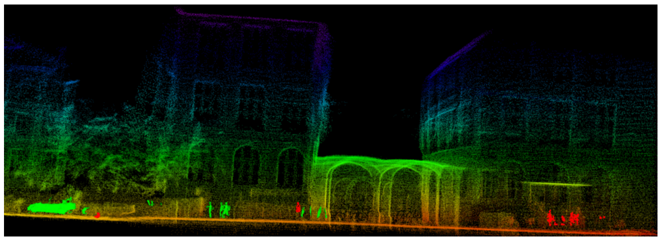
Precision and recall for a power plant sortie are listed in Table I and a map of changes for the same sortie with observancy requirement $n_{overlap} = 3$ is shown in Figure 7a. Here, 18 objects are mapped, all of which are true dynamics. Three actual environment changes, all of them small objects under $0.5m$ in size, are not mapped with this parameter configuration. Four environmental changes were detected by neither parameter setting. Two of them were disappearing objects that were observed multiple times but were under segmented during clustering, as discussed in Section IV-B.1. This caused the overlap of the clusters to be too small and prevented the objects from being added to the map. Two small appearing objects were observed only once during the sortie and could therefore not fulfill the requirement of being observed $n_{overlap}$ times.

$n_{overlap}$	Precision	Recall
2	0.91	0.84
3	1.0	0.72

TABLE I: Precision and recall with different $n_{overlap}$ for change mapping evaluated on a power plant sortie.



(a) Decommissioned power plant with several objects moved manually between sorties.



(b) Urban scene in Clausiusstrasse, Zürich. Changing clusters are cars and pedestrians.

Fig. 7: Map of changed points overlayed with the reference OctoMap colored by height. Bright green clusters indicate new objects in the environment. Red clusters indicate disappearing objects from the scene.

Results for a Clausiusstrasse sortie mapped with $n_{overlap} = 3$ are pictured in Figure 7b. All mapped clusters are true dynamics and none are missing from the map of changes.

4) Runtime: The change detection pipeline is designed to run online and in real-time on our target platform, which acquires a 3D point cloud every 3 seconds.

Table II shows mean computation times for different pipeline components running in a single thread, evaluated on a 8-core desktop CPU @3.3GHz with 16GB RAM, with LiDAR odometry and localization computed in parallel as would be done on a real system. Real-time performance is achieved on the target platform.

V. Conclusion

In this paper we proposed an environmental change detection pipeline that performs in real-time on distorted 3D point clouds with slow acquisition rate in cluttered environments.

We compute the Mahalanobis distance between points in the reading cloud and occupied voxels in the reference map as a measure for the change likelihood of a point. By classifying points with a learning-based algorithm we achieve robustness to noisy point clouds and under-sampled reference maps. Changes are mapped continuously and reported online.

We evaluated the change detection system on real-world datasets of urban and SaR environments. Overall, the change

Module	Time[ms]
Initialization	2.5±0.8
ICP	990.7±219.2
Disappearing Points	389.9±95.1
Appearing Points	52.1±15.0
Clustering	177.8±84.5
Classification	2.4±1.1
Change Mapping	26.4±37.2
Total	1642.2±453.3
Total - No ICP	651.4±234.1

TABLE II: Mean computation time for different pipeline parts.

detection system effectively detected 84% of the dynamics and real-time performance on the target platform was achieved. The output of distance computation is a meaningful metric for the likelihood of a point being dynamic. High classification performance is achieved despite the low amount of labelled training data available and can most likely be improved further with additional data, especially from environments not represented in the available datasets (e.g. unstructured outdoor environment). The change mapping results minimize the burden of false alarms of changes for a human operator and for higher robot autonomous tasks while still being computational affordable on the datasets used for evaluation. In the future, mapping performance can be improved further by employing a mapping approach which uses information on static points in addition to dynamic ones.

References

[1] D. Girardeau-Montaut, M. Roux, R. Marc, and G. Thibault, "Change detection on points cloud data acquired with a ground laser scanner," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. part 3, p. W19, 2005.

[2] F. Ferri, M. Gianni, M. Menna, and F. Pirri, "Dynamic obstacles detection and 3d map updating," in *IROS, 2015*. IEEE, 2015, pp. 5694–5699.

[3] A. Azim and O. Aycard, "Detection, classification and tracking of moving objects in a 3d environment," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 802–807.

[4] F. Pomerleau, P. Krüsi, F. Colas, P. Furgale, and R. Siegwart, "Long-term 3d map maintenance in dynamic environments," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 2014, pp. 3712–3719.

[5] J. P. Underwood, D. Gillsjö, T. Bailey, and V. Vlaskine, "Explicit 3d change detection using ray-tracing in spherical coordinates," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 4735–4741.

[6] P. Drews, P. Núñez, R. P. Rocha, M. Campos, and J. Dias, "Novelty detection and segmentation based on gaussian mixture models: A case study in 3d robotic laser mapping," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1696–1709, 2013.

[7] A. W. Vieira, P. L. Drews, and M. F. Campos, "Efficient change detection in 3d environment for autonomous surveillance robots based on implicit volume," in *IEEE Int. Conf. on Robotics and Automation*. IEEE, 2012, pp. 2999–3004.

[8] H. Andreasson, M. Magnusson, and A. Lilienthal, "Has something changed here? autonomous difference detection for security patrol robots," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2007, pp. 3429–3435.

[9] R. Ambruş, N. Bore, J. Folkesson, and P. Jensfelt, "Meta-rooms: Building and maintaining long term spatial models in a dynamic world," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 1854–1861.

[10] R. Ambrus, J. Ekekrantz, J. Folkesson, and P. Jensfelt, "Unsupervised learning of spatial-temporal models of objects in a long-term autonomy scenario," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015.

[11] A. Collet, B. Xiong, C. Gurau, M. Hebert, and S. S. Srinivasa, "Herbdisc: Towards lifelong robotic object discovery," *The International Journal of Robotics Research*, vol. 34, no. 1, pp. 3–25, 2015.

[12] R. Finman, T. Whelan, M. Kaess, and J. J. Leonard, "Toward lifelong object segmentation from change detection in dense rgb-d maps," in *Mobile Robots (ECMR), 2013 European Conference on*. IEEE, 2013, pp. 178–185.

[13] E. Herbst, P. Henry, X. Ren, and D. Fox, "Toward object discovery and modeling via 3-d scene comparison," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 2623–2629.

[14] E. Herbst, X. Ren, and D. Fox, "Rgb-d object discovery via multi-scene analysis," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4850–4856.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[16] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based loop-closure for 3d point clouds," *arXiv preprint arXiv:1609.07720*, 2016.

[17] M. Himmelsbach, F. v. Hundelshausen, and H.-J. Wuensche, "Fast segmentation of 3d point clouds for ground vehicles," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*. IEEE, 2010, pp. 560–565.

[18] R. J. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2013, pp. 160–172.

[19] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[20] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[21] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[22] G. McLachlan, *Discriminant analysis and statistical pattern recognition*. John Wiley & Sons, 2004, vol. 544.