Analytical Change Detection on the KITTI dataset

Johannes Schauer and Andreas Nüchter[1]

***Abstract*— We present an algorithm for explicit change detection on 3D point cloud data from a mobile mapping scenario, namely the KITTI dataset. Our method is able to partition a 3D point cloud into static and dynamic points using ray traversal of a 3D voxel grid. We are thus not using a machine learning approach or RGB camera data but instead compute the intersections of the scene volume with the lines-of-sight between the sensor and the measured points. Our approach does thus not require any object detection or tracking and has comparatively low requirements on the hardware. While our earlier work focused on dense point clouds from terrestrial 3D laser scans, here we investigate its application on the sparse 3D point clouds produced by a Velodyne laser range finder in a mobile mapping scenario and compare our results to two competing implementations using the ground truth annotation from FuseMODNet for a quantitative analysis. We also introduce spherical quadtree point cloud reduction as a way to only work on less than 1% of the original data, making processing multiple times faster while at the same time producing results with equivalent F_1 scores.**

I. INTRODUCTION

3D point clouds collected in urban environments are usually cluttered with noise from moving objects. Figure 1 shows a busy crossroad with the static environment in yellow and moving cars, cyclists and pedestrians in magenta. Since the points in the scene come from hundreds of individual scans, moving objects are seen multiple times and show up as long stripes or smears throughout the scene. For most applications it is not desirable to have the artifacts created by moving objects be part of the final point cloud. In an earlier publication [1] we presented a solution for removing dynamic objects from 3D point clouds from terrestrial scans. In this publication we apply our solution to scans gather from a continuously scanning mobile platform. Using such point clouds as input is challenging because in contrast to dense point clouds from terrestrial mapping, the point clouds are sparse and do not always provide clear surface normals. Registering thousands of point clouds also poses greater demands on the utilized SLAM algorithm and scanners used for mobile mapping are usually less precise than stationary terrestrial scanners.

In this publication we evaluate our algorithm on the KITTI [2] dataset. We show how we achieve better F_1 scores than a competing solution while also being faster. On a normal desktop computer we are able to process nearly 700 scans per second, purely on a Desktop CPU, which makes our solution fit for real-time applications. All our code is provided as part of *3DTK – The 3D Toolkit*[1] under the terms of the GPL. To make it possible for other researchers to benchmark their solution against ours, we also publish a shell script which downloads, compiles and runs all required code to output table I from the results section.[2] The website also contains the video published together with this paper as well as high-resolution images.

The input to our algorithm is registered 3D range data, typically acquired by a 3D laser range finder. While we only test our approach with LIDAR scans, it is in principle also compatible with scans obtained from RADAR or RGB-D systems or point clouds from stereo vision. Given such a set of registered point clouds, our algorithm is able to partition the points into those belonging to static as well as dynamic objects. Essentially, if a volume is seen as free where another scan measured points, then these points must be non-static and get marked for removal. We approximate occupied volume by a voxel grid and determine free voxels by traversing the lines of sight from the sensor to the measured points through the voxel grid. Our approach is able to detect change where two or more scans overlap in the volumes they explicitly observe as free or occupied. Apparent change created by occlusion is suppressed.

Our algorithm makes very few requirements on the underlying geometry of the scanned data, vantage points and the temporal separation between individual scans. The vantage points together with the geometry of the scene must be chosen such that the volumes of interest are not occluded from the sensor. Instead, the volumes that one wants to remove moving objects from must have been observed at

[1]Informatics VII: Robotics and Telematics, Julius-Maximilians-University, Am Hubland, Würzburg 97074, Germany johannes.schauer@uni-wuerzburg.de, andreas.nuechter@uni-wuerzburg.de

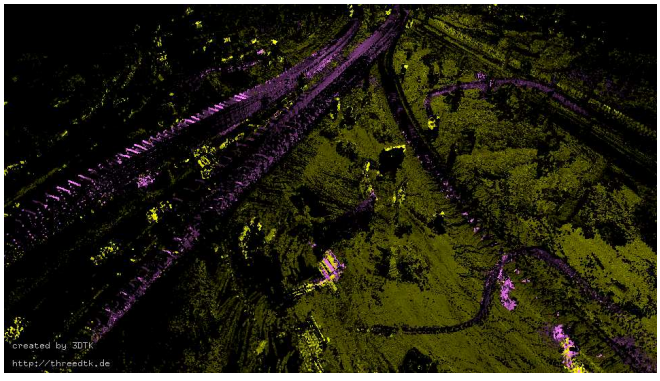


Fig. 1: crossroad from KITTI scene 51 with moving objects in magenta and static objects in yellow

[1]http://threedtk.de

[2]https://robotik.informatik.uni-wuerzburg.de/telematics/download/icarcv2020/

least by two different scans. Furthermore, the temporal difference between these two scans must be large enough such that any object that one considers "dynamic" in the observed volume was moved to a different location. But if a given voxel volume was observed more than twice, then it is sufficient that the voxel was seen as "free" by only a single scan.

This paper is structured as follows. In the next section we handle work related to ours. In the following section III we explain our measurement method. Section IV contains an explanation of how we use the spherical quadtree that we introduced in our earlier research to speed up processing. Section V then contains timing benchmarks as well as quantitative and qualitative results of our approach as applied on the KITTI dataset and a discussion thereof. We describe our future work and draw conclusions in the last two sections.

II. RELATED WORK

Our solution falls into the realm of change detection [3] but only few publications deal with classifying points as either dynamic or static. Even fewer approaches compute the free volume between a measured point and the sensor itself. Most solutions for change detection compare incoming geometries or point clouds in a way that results in "change" merely due to occlusion or incomplete sensor coverage. One example for such an approach is the method by Vieira et al which uses spatial density patterns [4]. Or the solution shown in by Liu et al. [5] which just computes the difference in voxel occupation between two input scans. But for our purpose of "cleaning" scans, it is undesirable to remove these parts from the dataset. Doing so would mean to remove potentially useful data from the input. Instead, we designed our algorithm to be conservative. It only removes volumes which it is able to confidently determine to be dynamic. Volumes which it cannot make a decision upon, for example because they were only measured by a single scan, are left untouched. Meeting this requirement is only possible by computing unoccupied volumes and detecting change explicitly. The changes we are interested in can only be detected if a given point falls into the volume that another measurement observed as free.

The work most similar to ours is the seminal work by Underwood et al. [6] It is able to detect changes between two scans by ray tracing points in a spherical coordinate system. But since their algorithm is limited to comparing no more than two scans at a time it is not directly applicable to our use case without either additional heuristics or quadratic runtime with respect to the number of scans. Given N input scans and without additional processing to find scan pairs with a "meaningful" overlap in their observed volume, the only way to find all changed points is to compare all possible pairs of scans. With N scans this results in a worst case scenario of $\frac{N(N-1)}{2}$ comparisons and thus quadratic runtime. Our approach is of linear complexity relative to the input size because all comparisons are made against a global occupancy grid and not directly against point data from other scans. The authors publicly provide their code and their datasets which we thus use to benchmark our own method against theirs.

The KITTI dataset [2] is a popular dataset for computer vision benchmarks from the Karlsruhe Institute of Technology and their raw datasets are free to download.[3] They equip a standard station wagon with two high-resolution color and grayscale video cameras, a Velodyne laser scanner and a GPS localization system with an IMU. The focus of the dataset lies on object recognition, object tracking and visual odometry or laser-based SLAM. For these purposes, the dataset comes with extensive ground truth annotations in form of 2D and 3D bounding boxes for the 2D camera images and the 3D point cloud from the laser scanner, respectively. But while objects like cars, vans, trucks, pedestrians, trams and cyclists are correctly annotated, this does not yet supply a fitting ground truth for the evaluation of our approach to change detection because many of these objects can also be stationary, like parked cars or sitting people.

To still use the KITTI dataset for evaluation, we make use of the third party annotations provided by the authors of MODNet [7] in form of the MoSeg-KITTI Motion Segmentation or KITTI MoSeg dataset. That dataset provides binary masks for 1300 images from different scenes from the KITTI dataset. This data was further expanded by the authors of FuseMODNet [8] to binary masks for 12919 images.[4]

Due to space constraints, we do not go into detail about our algorithm in this paper. To understand concepts like "point shadows" or the spherical quadtree data structure, we kindly refer the reader to our earlier detailed publications.[9] The novel contributions by this paper are the application of our existing algorithm to a different problem space and the speed-ups we gain without reducing the output quality by reducing the input point clouds to below 1% of their original size by using the existing spherical quadtree data structure for point cloud reduction.

III. METHOD

Since change detection approaches are highly sensitive to registration errors, the KITTI scans were registered using slam6d, a SLAM implementation from 3DTK. For this purpose, the scan locations were transformed from WGS-84 coordinates into ECEF coordinates, centered at the position of the first scan as coordinate origin. This transformation is not only useful because most software requires cartesian coordinates as input but also because the 32 bit floating point datatype is not precise enough to store large values as they are common in WGS-84 coordinates to a precision necessary for change detection. Attempting to handle WGS-84 positions as 32 bit floats will commonly result in a loss of precision in the range of several decimeters. At the latitude where the KITTI dataset was recorded, the distance between two representable poses in 32 bit floating point is $42\ cm$ and thus unsuitable to accurately represent the point cloud data.

[3] http://www.cvlibs.net/datasets/kitti/raw_data.php

[4] https://sites.google.com/view/fusemodnet

The image masks from FuseMODNet apply to the images captured by the left camera on the KITTI measurement setup. White pixels signify moving objects and black pixels mark static objects in the captured image. Since our algorithm works on the point cloud data only, we use the transformation matrices between the laser scanner and the left camera as provided in the KITTI datasets to project the point clouds onto the image space. Of the total point cloud, only 15.87% of all points on average are also seen by the left camera. We classify assign the ground truth values of being either dynamic or static points to them depending on whether they are projected onto a white or black pixel, respectively.

Our own method as well as the method by Underwood et al. that we will be benchmarking against use several parameters. For both methods, we selected optimal input parameters by computing the F_1 score for several discrete values. Since the approach by Underwood et al. only compares scan pairs and would thus require $\frac{N(N-1)}{2}$ comparison for a scene with N scans, we only compare the current scan to 10 randomly selected different scans within a radius of $10\ m$ of the current scan That strategy produced better F_1 scores than other approaches like picking 5 scans immediately before and after the current scan. A possible explanation for this effect is, that the latter approach does not account for situations in which the vehicle is stationary, because picking scans from different vantage points improves the result. This reduces the necessary number of comparisons from 6125127 down to 128547 or 19 hours of computation time for the whole dataset. The best parameters for the approach by Underwood et al. are $T_a = 1.3$ and $T_r(m) = 0.74$. The only parameter for our method is the voxel size. The voxel size with the best results was $0.74\ m$.

IV. SPHERICAL QUADTREE REDUCTION

The runtime of our approach linearly depends on the number of input points to compute the intersection between measurement rays and the voxel grid. The closer to the sensor, the more measurement rays intersect with the same voxel but to mark a voxel as "free" it only needs to be traversed by a single ray. The effect is more pronounced for dense point clouds but is even applicable for sparse point clouds from the KITTI dataset because the ground truth data only labels objects that are relatively close. The problem also occurs for scanners with more than one rotational axis where the closer to one of the axis one gets, the more dense the point cloud becomes.

From knowing the geometry of the Velodyne scanner used in the KITTI dataset, it would be straight forward to generate a sampling pattern that would produce an overall equal sampling of the measurement rays. Instead, we choose a solution that works equally well without knowing the scanner geometry upfront. For this task we make use of the spherical quadtree which our algorithm already created for computing "point shadows". The data structure is created by recursively subdividing the projection of a octahedron onto a unit sphere surface. This produces a data structure where each triangle on the unit sphere surface is subdivided into four smaller triangles of similar area until a stopping criterion is reached. We already used this data structure to efficiently look up angular neighbors in our earlier publications. Now we use the same existing data structure to sample the point cloud up to a certain number of points per angular radius. This process works analogous to how points are reduced in an Octree.[10]

V. RESULTS

A. Performance

We carried out our benchmarks on an Intel Xeon e5-2630 v3 octacore desktop system with 2.4 GHz and 32 GB RAM. If the time to load data from disk and writing the results back to disk are ignored, then our algorithm is able to process the whole KITTI dataset (1553 Million points) in 183 seconds. This translates to 70 scans per second or 8.5 Million points per second. If one is only interested in the points that are seen by the left camera (246 Million overall), then our method is able to process 202 scans per second. By using spherical quadtree reduction, the number of points the algorithm needs to process can be further reduced. Using the parameters that produced the results in the sevenths column of Table I, only 1661637 (0.67%) points remain. Even including the time it takes to perform the point reduction, we can measure a performance of 693 scans per second. This means, that a 3.4 time speedup can be realized on the KITTI dataset without sacrificing output quality in terms of F_1 score. Since the KITTI dataset contains 10 scans per second, we would be able to process it nearly at 70 times the normal speed. With dense 3D point cloud data from terrestrial mapping, the speedup gained by this method can be up to two orders of magnitude with equal solution quality.

B. Quantitative Results

The results of running our approach as well as the method by Underwood et al. on the KITTI dataset can be seen in Table I. The first column shows the KITTI scene identifier. The second column shows the number of scans in that scene. Each full scan contains 120000 points on average. Since the results in this table are limited to the points seen by the left camera, each scan contains around 19000 points. The third column shows the percentage of non-static points according to the ground truth information from the FuseMODNet image masks. The fourth column "F_1 UW" shows the F_1 score from applying the method by Underwood et al. The fifth column shows the F_1 score for our method. The sixth column "F_1 210" shows the F_1 score with a minimum cluster size of 210. The seventh column "F_1 Red" shows the F_1 sore achieved after reducing the points using the spherical quadtree such that at most 10 points remain in an angular radius of 0.2 radians as well as a minimum cluster size of 210. The last column shows the result of the same experiment as for column F_1 but using the Intersection over Union metric for the moving points. We omitted the IoU for the static points because it was consistently close to or above 0.98 with a total non-moving IoU of 0.9783. This effect can also be seen in Figure 2b. The last line shows grand totals. The total

TABLE I: Quantitative results

ID	scans	dyn	F_1 UW	F_1	F_1 210	F_1 Red	IoU
1	108	0	0	0	0.0	0.0	0.0
2	77	0.03	0.0019	**0.0047**	0.0	0.0	0.0
5	154	2.37	0.2633	**0.4279**	0.5162	0.5015	0.3479
9	443	1.47	0.3005	**0.3658**	0.5817	0.2916	0.4101
11	233	3.24	0.4385	**0.5544**	0.6523	0.4185	0.4840
13	144	3.31	0.2610	**0.5684**	0.6643	0.3323	0.4973
14	314	2.49	0.1311	**0.2582**	0.2277	0.0835	0.1285
15	297	2.31	0.4906	**0.6331**	0.6818	0.4406	0.5172
17	114	3.00	**0.7122**	0.6285	0.6155	0.5311	0.4446
18	270	3.62	0.4409	**0.4975**	0.7065	0.6561	0.5462
19	481	1.27	0.2287	**0.4534**	0.7434	0.5370	0.5916
20	86	0.10	0.0055	**0.0281**	0.0384	0.0	0.0196
22	800	0.58	0.1035	**0.2295**	0.6454	0.6044	0.4764
23	474	0.02	0.0004	**0.0005**	0.0	0.0	0.0
27	188	0.71	0.3008	**0.4856**	0.6988	0.4305	0.5371
28	430	0.49	0.0373	**0.4312**	0.6795	0.0	0.5146
29	430	1.67	0.1426	**0.3943**	0.4678	0.6095	0.3053
32	390	1.80	0.0270	**0.4960**	0.6001	0.3069	0.4287
35	131	0.02	0	**0.0021**	0.0	0.0	0.0
36	803	1.73	0.0879	**0.4845**	0.6702	0.6255	0.5040
39	395	0.61	**0.2994**	0.2739	0.7382	0.4790	0.5850
46	125	2.13	0.2373	**0.6562**	0.8290	0.7178	0.7080
48	22	6.39	0.2051	**0.5656**	0.5972	0.0	0.4257
51	438	5.63	0.2161	**0.6529**	0.7724	0.6834	0.6292
52	78	0	0	0	0.0	0.0	1.0
56	294	1.61	0.1141	**0.4341**	0.5047	0.4971	0.3375
57	361	3.38	0.2125	**0.2708**	0.3096	0.2527	0.1831
59	373	2.98	0.4090	**0.4449**	0.5758	0.4893	0.4043
60	78	0.26	0.1634	**0.3380**	0.6480	0.0	0.4792
61	703	0	0	0	0.0	0.0	1.0
64	570	0.07	0.0229	**0.0520**	0.4670	0.4784	0.3047
70	420	0.42	0.0462	**0.3831**	0.6564	0.6885	0.4886
79	100	0	0	0	0.0	0.0	1.0
84	383	0.79	0.1249	**0.2260**	0.3877	0.3764	0.2405
86	706	0	0	0	0.0	0.0	0.0
87	729	0	0	0	0.0	0.0	0.0
91	340	0	0	**0.0005**	0.0007	0.0010	0.0004
93	433	0	0	0	0.0	0.0	0.0
all	12915	1.19	0.1462	0.2936	0.4173	0.4232	0.2637

number of scans is lower than the 12919 masks provided by the FuseMODNet dataset, because some camera images in the KITTI dataset did not come with an associated 3D point cloud. The remaining numbers in the last row are not an average of the numbers above, but are computed as if all KITTI scenes together were one single dataset.

Both the F_1 score as well as IoU yield low results when the number of moving points in a scene is low compared to the number of static points or even zero. Rows with close to zero or zero dynamic points (column three) in Table I also yield F_1 scores or IoU values close to zero. In figure 2a we plot the relationship between the percentage of dynamic points of the total number of points versus their respective F_1 scores. No obvious relationship between the two values exists.

The minimum cluster size of 210 in column six of Table I was chosen after evaluating the F_1 score for multiple minimum cluster sizes as can be seen in Figure 2b. Larger cluster sizes produce better F_1 scores, because the ground truth data from FuseMODNet does not label pedestrians or cyclists as moving. Thus, our algorithm achieves better scores if it ignores smaller moving objects in favor of cars and trucks.

Figure 2c shows how there is a relatively large range of voxel sizes that produce close to optimal F_1 scores. Values between 25 and 50 all produce F_1 scores over 0.5. More research is needed to evaluate how scene or scanner specific the voxel size really is.

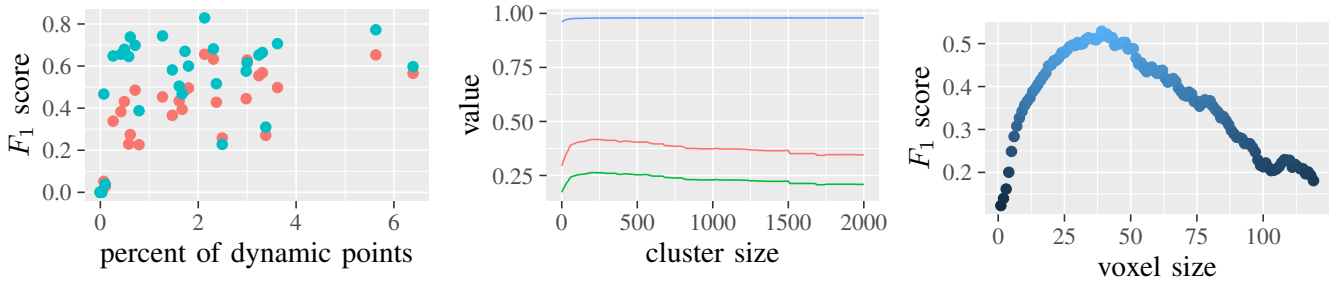
Even though we provide results using the IoU metric, a direct comparison to the results from the FuseMODNet paper is not possible because our approach works on the point cloud while theirs evaluates IoU in the two-dimensional image space of the camera. Since the point clouds from the velodyne scanner are not dense enough, even projecting the point clouds into the image plane would not yield comparable results.

While our solution does outperform the algorithm by Underwood et al, our F_1 scores are much below those of over 0.95 that we achieved for terrestrial scans in earlier publications. There are multiple reasons for this effect. Importantly, the FuseMODNet labeling only labels vehicles. Any other moving objects that are correctly identified will be classified as false positives according to our ground truth. Furthermore, the sparse point clouds from the Velodyne laser scanner provide less reliable surface normals and thus our approach to avoid artifacts by computing "point shadows" fails.

Figure 3 shows "impossible" points situated under the street surface which are a result of various reflections in the scene. Common sources for these reflections are parked cars and window fronts besides the street. Such points also exist in other locations but those below the street are easiest to visualize to demonstrate this problem. Since the algorithm assumes that the line-of-sight between the laser range finder and all points is "empty", reflected points will introduce false positives.

Another problem source is, that the masks often misclassify points as can be seen in Figure 4. The figure shows the left camera frame number 385 from the KITTI scene 9 and is overlayed with the corresponding mask from the FuseMODNet project. Pixels belonging to dynamic objects are marked white. On the left-hand-side of the camera image, one can see that large parts of the traffic sign were marked as dynamic even though the traffic sign is static. In the center of the image, it can be seen, that the front of the car was not marked as dynamic even though it belongs to a dynamic object. Thus, imperfections of the underlying ground truth introduce false positives as well as false negatives into our results.

Speaking in general, our algorithm produces false positives in situations where scans are either not correctly registered or due to sensor noise. An example is a flat surface where not all points lie on the surface. The points "in front" of the surface in scanner direction will then be marked as "see through" even though they belong to a static object. Another source of false positives arises when surface normals are wrongly computed and thus point shadows are not determined correctly. Due to the noisy nature of the KITTI dataset there were many sources of both of these issues, leading to a high number of false positives. Another source



(a) F_1 score with (red) and without (green) minimum cluster size by percentage of dynamic points in a KITTI scene.

(b) F_1 score (green), moving IoU (red) and non-moving IoU (blue) for different minimum cluster sizes over the whole KITTI dataset

(c) F_1 score with different voxel sizes for the scans marked grey in Table I

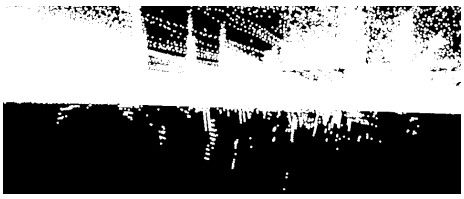


Fig. 3: Points from reflections under the street surface

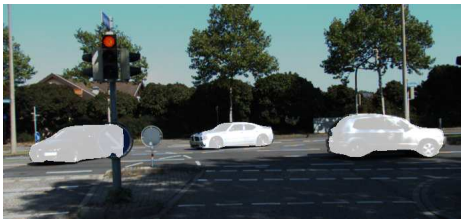


Fig. 4: Examples of wrong classifications of binary masks from FuseMODNet from KITTI scene 9, frame 385.

of false positives are mirrors and transparent objects. Lastly – if enabled – some false positives are introduced by our approach to subvoxel accuracy which we only enabled for the qualitative results.

False negatives are created either in situations where a volume was only seen by a single laser scan or in volumes that were ”shadowed” by closer points. We observed the latter problem in a dataset where we placed the scanner directly on the ground instead of on a tripod to take a scan. This resulted in points from the ground directly adjacent to the scanner to shadow most of the lower part of the scan and thus make it impossible for our algorithm to classify any points close to the ground as dynamic. Additionally, false negatives are introduced if the chosen voxel size is so small, that rays are able to penetrate objects without intersecting a voxel with points in it. Since the point density typically decreases with their distance from the sensor, this effect also occurs at very far distances. Applying a clustering filter can also introduce false negatives if the dynamic object is smaller than the chosen minimum cluster size.

C. Qualitative Results

Qualitative results can be seen in Figure 5. Additionally we publish a video alongside with this publication which can also be viewed on the project website as well as on YouTube[5] using the scenes marked in grey in table I. In contrast to the quantitative results from Table I, we use the full point clouds from the Velodyne scanner and not just the 15.87% of points that can be projected to the left camera image. To produce the best possible results, we use the optimal voxel size and minimum cluster size as parameters and do not disable subvoxel accuracy.

Most figures show moving cars and trucks. A pedestrian is most prominent in Figure 5b but also show in Figures 1 and 5l. Cyclists can be seen in Figures 1, 5g and 5h. Figure 5j shows a registration error that resulted in a wall seen in multiple scans not being properly aligned in the registered final point cloud. The result is, that the “outer” walls are marked as dynamic because they appear as “see through”.

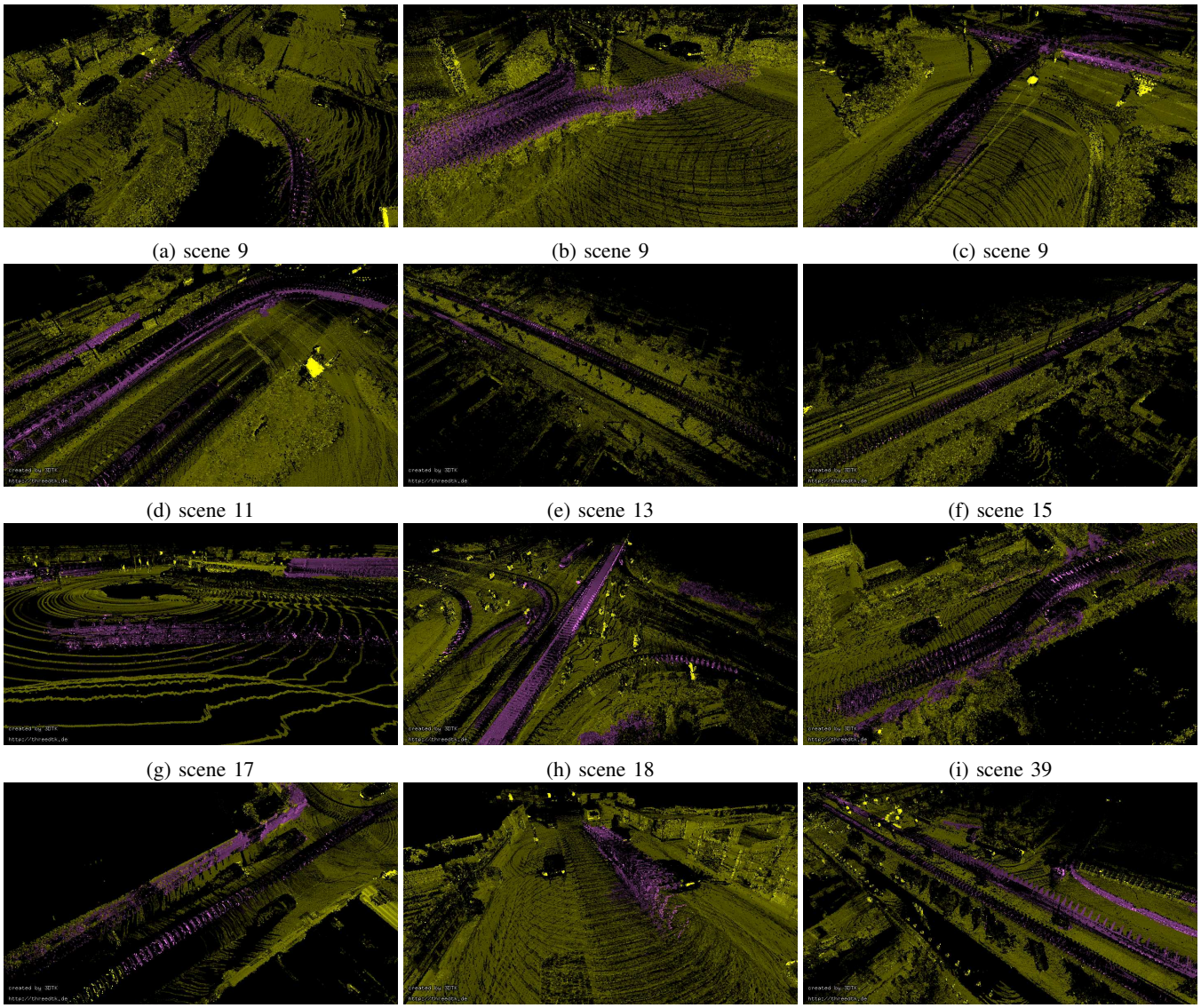
VI. FUTURE WORK

Needless to say a lot of work remains to be done. While we used our approach for the purpose of partitioning terrestrial laser scan data into static and dynamic points for the purpose of obtaining a point cloud free of dynamic points, applying our approach to a mobile mapping scenario creates many new possibilities. Currently our solution is a post-processing method but there is no conceptual reason not to execute the algorithm on live data. Another interesting application could be to use the built-in point cloud segmentation functionality to preprocess data for machine learning approaches.

VII. CONCLUSION

In this work we applied our algorithm for explicit change detection on the well known KITTI dataset and compared it with another competing approach. We have shown how our algorithm achieves higher F_1 scores and compares favourably in runtime as well as in qualitative results. Using the spherical quadtree as an existing pre-computed data structure of our solution we are able to significantly speed up our algorithm. Since it is very light on computing resources,

[5] https://youtu.be/khH7Bs7Cp3o



(a) scene 9 (b) scene 9 (c) scene 9

(d) scene 11 (e) scene 13 (f) scene 15

(g) scene 17 (h) scene 18 (i) scene 39

(j) scene 46 (k) scene 48 (l) scene 59

Fig. 5: Qualitative results from our method using the datasets marked gray in table I

it is an ideal companion algorithm as part of a bigger pipeline.

REFERENCES

[1] J. Schauer and A. Nüchter, "The Peopleremover — Removing Dynamic Objects From 3-D Point Cloud Data by Traversing a Voxel Occupancy Grid," *IEEE Robotics and Automation Letters (RAL)*, vol. 3, no. 3, pp. 1679–1686, July 2018.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[3] R. Qin, J. Tian, and P. Reinartz, "3d change detection–approaches and applications," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 122, pp. 41–56, 2016.

[4] A. W. Vieira, P. L. Drews, and M. F. Campos, "Spatial density patterns for efficient change detection in 3d environment for autonomous surveillance robots," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 3, pp. 766–774, 2014.

[5] K. Liu, J. Boehm, and C. Alis, "Change detection of mobile lidar data using cloud computing," in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives*, vol. 41. International Society of Photogrammetry and Remote Sensing (ISPRS), 2016, pp. 309–313.

[6] 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.

[7] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, "Modnet: Moving object detection network with motion and appearance for autonomous driving," *arXiv preprint arXiv:1709.04821*, 2017.

[8] H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, "Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[9] J. Schauer and A. Nüchter, "Removing non-static objects from 3d laser scan data," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, vol. 143, pp. 15–38, 2018.

[10] J. Elseberg, D. Borrmann, and A. Nüchter, "Efficient Processing of Large 3D Point Clouds," in *Proceedings of the XXIII International Symposium on Information, Communication and Automation Technologies (ICAT '11)*. Sarajevo, Bosnia: IEEE Xplore, October 2011.