IEEE International Conference on Robotics and Automation (ICRA), Montreal, Canada, May 2019.

Detection and Tracking of Small Objects in Sparse 3D Laser Range Data

Jan Razlaw, Jan Quenzel, and Sven Behnke

***Abstract*— Detection and tracking of dynamic objects is a key feature for autonomous behavior in a continuously changing environment. With the increasing popularity and capability of micro aerial vehicles (MAVs) efficient algorithms have to be utilized to enable multi object tracking on limited hardware and data provided by lightweight sensors. We present a novel segmentation approach based on a combination of median filters and an efficient pipeline for detection and tracking of small objects within sparse point clouds generated by a Velodyne VLP-16 sensor. We achieve real-time performance on a single core of our MAV hardware by exploiting the inherent structure of the data. Our approach is evaluated on simulated and real scans of in- and outdoor environments, obtaining results comparable to the state of the art. Additionally, we provide an application for filtering the dynamic and mapping the static part of the data, generating further insights into the performance of the pipeline on unlabeled data.**

I. Introduction

As robotics is getting more and more popular, autonomous robots are utilized in a growing variety of environments and situations. One basis for the safe deployment of autonomous machines is a robust perception and anticipation of continuous changes in the world. This problem is addressed by detection and tracking algorithms. Detection consists of identifying or perceiving objects of interest, while tracking is the task of monitoring the objects' states over time. Knowledge about their temporal history allows to anticipate future behavior.

Recent developments in the field of lightweight light detection and ranging (LiDAR) sensors facilitate their use on micro aerial vehicles (MAVs). These MAVs are utilized in an increasing number of applications, like mapping [1], inventory [2], or even health care [3]. For those, collision avoidance and dynamic path planning ensure safety and enable the efficient usage of restricted resources with regards to energy consumption and flight time. Detection and tracking of dynamic objects is a key feature for these tasks and interaction with the environment in general.

Another beneficiary of improvements in this field is autonomous driving. The cars are usually equipped with powerful computers and a variety of different sensors. MAVs, on the contrary, are constrained by their lifting capacity—hence, providing limited computational power and allowing lightweight sensors only.

Our goal is detecting and tracking multiple objects of a specific size—e.g. humans—in sparse point clouds as depicted in Fig. 1. These point clouds are generated by a Velodyne VLP-16 sensor mounted underneath a DJI Matrice 600 MAV. Due to the limited compute power of the MAV, efficient algorithms have to be utilized for detection and tracking to achieve real-time performance.

All authors are with the Autonomous Intelligent Systems Group, University of Bonn, Germany {razlaw, quenzel, behnke}@ais.uni-bonn.de

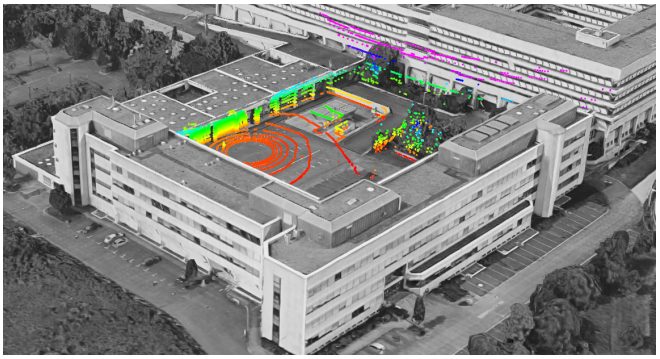

Fig. 1: An exemplary point cloud generated from one scan of the courtyard of the Landesbehördenhaus in Bonn, projected into an aerial photography [4]. Points are colored by height.

II. Related Work

Object tracking algorithms in general can be subdivided into two categories. In the first category are model-based tracking algorithms that utilize a detector to discriminate target objects from others based on a model description. The second category covers model-free tracking comparing consecutive scans of the environment to distinguish dynamic objects from the static background.

The challenge for the former lies in the creation of a precise object model able to discriminate targets from non-targets while accounting for different settings and special cases. Numerous works investigated the usage of trained classifiers [5], [6], [7], [8] on a variety of features and descriptors. Others utilized convolutional or recurrent neural networks [9], [10], [11] with promising results, but mostly on camera images.

Model-free tracking on the contrary is independent of a predefined model. Objects are detected either by searching for similar regions in consecutive scans implicitly building and updating a model [12] or, as usually applied for multi object tracking (MOT), by extracting the background and tracking the remaining measurement groups [13], [14]. Such methods rely on the dynamics of objects as static or temporarily static objects are not tracked.

MOT algorithms additionally need to provide an assignment method capable of matching detections to corre-

sponding tracked objects. Such assignment methods range from simple approaches minimizing pairwise distances of matches [7] to sophisticated but computationally more demanding joint probabilistic data association filters [15] or likelihood-based sampling based methods [16]. Recent works [11], [17], [8] investigated the capability of recurrent neural networks, such as Long Short-Term Memory networks [18], for assignment and tracking.

In this work, we attempt to combine the best of both worlds by relying on a simple, thus general, object model to not only preserve the ability of tracking static or temporary static targets but also to reduce parametrization effort and generalize to different settings. Additionally, we utilize the tracker's temporal information in the detector to reduce the rate of missed detections and concentrate on the usage of efficient algorithms to process data on limited hardware in real-time.

In summary, the key features of our method are:

- A novel approach to segment point groups of a specified width range,
- a detector utilizing segments, temporal information and the inherent structure of the data,
- an efficient multi object tracker able to maintain tracks through short occlusions characteristic to the data,
- real-time capability on a single CPU core of our MAV hardware,
- and a practical application for filtering the dynamic and mapping the static part of the world.

III. Method

In the following, we provide a step-by-step description of the implemented MOT pipeline depicted in Fig. 2. Starting with the point cloud generated by the sensor, we preprocess the data by segmenting foreground point groups of a specified width range. This segmented cloud is used to create object detections that are fed to the MOT algorithm. The estimated tracks are then returned to the detector aiding the detection in following scans. We estimate the object states in the world frame. For this purpose, we utilize Multi Resolution Surfel Mapping [19] to estimate the sensors position in the world. This mapping algorithm was explicitly developed to work in real-time on sparse laser range data.

A. Point Cloud Generation

The chosen sensor to detect small objects in a long range is the Velodyne VLP-16. The sensor has 16 laser-detector pairs placed on a vertical axis and oriented with an angle of 2° to each other resulting in a 30° vertical field of view (FoV). This setup allows getting 16 measurements at a time. Spinning the laser-detector pairs around the sensor's vertical axis generates a 360° horizontal scan of the environment consisting of 16 *scan rings*. Fig. 1 shows an example scan.

We exploit the configuration of $16 \times n$ measurements to generate one organized point cloud per full rotation, preserving the grid-like structure of the scan.

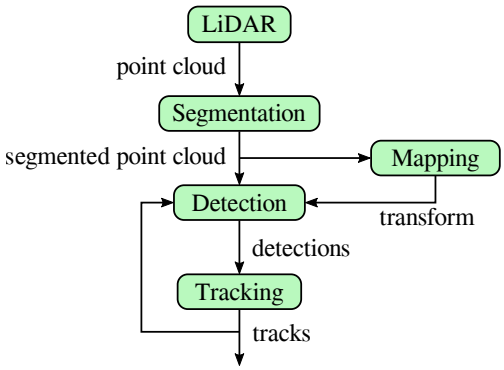

Fig. 2: Overall concept of the MOT pipeline.

B. Segmentation

The scan rings generated by the sensor are deformed by objects in the environment, resulting in grouped measurements closer to the sensor than their neighboring measurements from the background. Due to the sensors low vertical resolution, especially small or distant objects raise the risk of laying in between scan rings or corresponding to only very few measurements. Training of sophisticated object models under these circumstances is hard if not impossible. Hence, we segment objects according to their width, as this is the most distinct feature we can compute even for distant targets. Our goal is to find all points belonging to foreground groups of a specified width range.

We utilize the segmentation method we presented in [20] consisting of two median filters—one for noise, one for background—with different kernel sizes applied to the distance readings of single scan rings. This method segments foreground point groups of a specified width, by filtering all narrower objects utilizing the *noise filter* with a smaller kernel size and additionally filtering the target objects themselves using a slightly bigger kernel size in the *background filter*. Points for which the filters return different results are classified as target points. This approach naturally extends to segmenting groups of a specified width range by defining the minimal and maximal width explicitly through the noise and background filter kernel sizes.

By exploiting the organized structure of the data, we introduce points with invalid distance readings—e.g. measurements in the direction of the sky or on absorbing surfaces. We automatically classify these as background points and replace the invalid distance by a fixed value exceeding the maximal measuring range. This way, invalid points can still be utilized by neighboring valid measurements during the median computation and allow segmenting objects in the sky, e.g. other MAVs.

C. Detection

Detection generation is split up into clustering of segmented points and subsequent filtering. For clustering, we utilize region growing on the organized structure of the point cloud. The region growing algorithm connects a seed point to its neighboring segment points and those successively to their

Algorithm 1: Region Growing Clustering (RGC)

```
1:  function RGC(Organized grid of segmented points P)
2:      C : empty list of clusters
3:      Q : empty queue of points to process
4:      c_min : minimal number of points within valid cluster
5:      for each not visited segment point p_i ∈ P do
6:          c : empty cluster
7:          add p_i to Q and c
8:          mark p_i as visited
9:          while Q not empty do
10:             dequeue p_j from Q
11:             for each neighbor p_n : ||p_n − p_j||_1 ≤ r do
12:                 if p_n not visited and a segment point and
                        ||p_n − p_j||_2 < θ then
13:                     add p_n to Q and c
14:                     mark p_n as visited
15:         if |c| ≥ c_min then
16:             add c to C
17:     return C
```

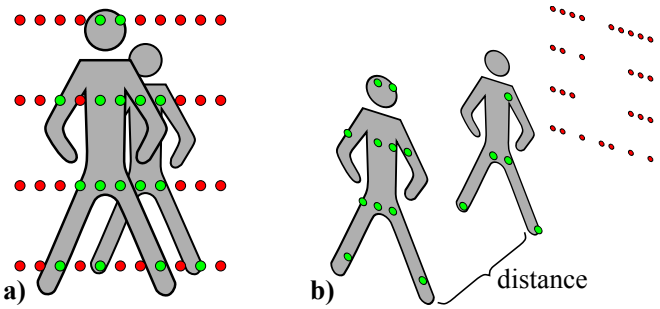

Fig. 3: Two persons scanned behind each other with green segment points and red background points. **a)** Sensor perspective: Simply connecting the direct neighbors would result in wrong clusters. **b)** Shifted perspective: Accounting for the distance between neighboring points helps to distinguish segment points corresponding to different objects.

neighboring segment points. The neighborhood is defined on the grid structure of the cloud. We apply Alg. 1 to each unclustered segment point.

We adapt the clustering at two points to work more robustly on the special structure of our data. Due to the sparsity of the data, the neighborhood search radius in line 11 is extended to check more than just the direct neighbors on the grid (Fig. 3a). Additionally, in line 12 the Euclidean distance of the current point to its neighbors has to be taken into account to prevent under-clustering of several distinct but partially occluding objects (Fig. 3b).

One drawback of the organized grid for this clustering method is a potential overlap of the start and end of a circular scan. We need to handle this case explicitly by computing the approximate overlap, finding the clusters lying within and fusing those corresponding to the same objects in the world.

After clustering the segment points, we need to filter those clusters not matching our simple object model consisting of a height range and a maximal diagonal width. Due to the sensor's limited vertical FoV, objects might be scanned partially. Consequently, we refrain from testing for a minimal height for clusters containing at least one point from the top and bottom scan rings. Additionally, we exploit the tracker's temporal information by relaxing the object model—omitting the minimal height and increasing the maximal width—for clusters in the vicinity of already tracked objects. Clusters fitting this description are considered valid detections.

D. Multi Object Tracking

Multi Object Tracking in general is the task of monitoring the states of several objects simultaneously. For this purpose, the algorithm maintains a set of object hypotheses. These are represented by an axis aligned bounding box and a state consisting of a 3D position and velocity. The hypotheses are updated using points corresponding to valid detections in the most current scan. One Kalman filter with a constant velocity model is deployed for each hypothesis.

For tracking multiple objects, it is essential to know which detection corresponds to which object hypothesis. We solve this classical assignment problem in polynomial time utilizing the Hungarian method [21]. The algorithm finds a one-to-one assignment for a given cost matrix minimizing the total assignment costs. Hence, we model our problem as an adjacency matrix between detections and hypotheses utilizing the Bhattacharyya distance as a proximity measure. We forbid individual assignments that exceed a distance threshold.

Each assigned detection is used to correct the matched hypothesis state. We prevent velocities induced by sensor noise and varying amounts of partial occlusions by truncating velocity estimates of up to 1 km/h to zero. Additionally, we truncate estimated velocities to a maximum of 10 km/h to reduce the effect of volatile detections, due to sparse measurements on altering object parts.

A detection without a matching hypothesis might be correspondent to an object entering the FoV and thus creates a new hypothesis. Detections generated using the relaxed object model implicitly correspond to already tracked objects—hence do not create new hypotheses. Unassigned hypotheses might correspond to objects that left the FoV and thus need to be tested for validity. As unassigned hypotheses' covariances grow with each prediction step, we declare those with a high covariance as non-valid and initiate deletion. For this, we compute the eigenvalues of the Kalman filter's covariance matrix and test if at least one value exceeds a threshold. To account for oversegmentation or misassignments, we delete the younger hypotheses being in the vicinity of older hypotheses.

Additionally, we classify tracked objects into the classes *static* and *dynamic*. Each hypothesis is generated static and becomes dynamic once its current bounding box does not intersect with its initial bounding box.

E. Dynamic Objects Filter

As a practical application, we filter out tracked dynamic objects from all point clouds and generate a map of the static

part of the scene. For this, we discard every detected point corresponding to a dynamic hypothesis from each currently processed point cloud. These filtered clouds are stored in a log. Once a hypothesis becomes dynamic, all its previous bounding boxes—from the time it was static—are utilized to remove corresponding points from these logged clouds. If a dynamic hypothesis loses track of an object but recovers, we filter points corresponding to the predicted bounding boxes from intermediate steps similarly. The result is a log of point clouds corresponding to the static part of the world.

IV. Evaluation

We evaluate efficacy and efficiency of our proposed MOT approach as follows. We start by giving an overview of two metrics commonly used to evaluate MOT algorithms against ground truth annotations. These annotations are provided in two data sets we inspect afterwards. We utilize both for evaluation and parameter optimization and discuss our achieved results.

A. Evaluation Metrics

The first metric is the CLEAR MOT metric [22]. It utilizes the Hungarian method to assign hypotheses to ground truth labels in each time step. Valid matches have a Euclidean distance below a threshold—$0.5\,\mathrm{m}$ are suggested by the authors for visual people tracking. Matched hypothesis-label pairs from the previous time step are not reassigned to others if both are still present and close to each other. Using the sum of distances d_t between each hypothesis and matched label for time t and the number of matches c_t it computes the Multi Object Tracking Precision (MOTP) as

$$MOTP = \frac{\sum_t d_t}{\sum_t c_t}. \tag{1}$$

Additionally, it counts the number of missed labels m_t, false positives fp_t, mismatch errors mme_t and labels g_t for each time t to compute the Multi Object Tracking Accuracy (MOTA) as

$$MOTA = 1 - \frac{\sum_t (m_t + fp_t + mme_t)}{\sum_t g_t}. \tag{2}$$

The MOTA is a measure for the consistency of the generated tracks.

Another way to evaluate the performance of an MOT algorithm is to inspect how much of the object tracks were covered by the hypotheses [23]. This metric is split up into three ratios: Mostly Tracked (MT), Partially Tracked (PT), and Mostly Lost (ML). An object trajectory is mostly tracked if at least 80% of it is covered by hypotheses. It is mostly lost if less than 20% is covered and partially tracked for the remaining cases. We apply the same constraints as for the CLEAR MOT for an object to be classified as tracked.

This metric does not account for identity switches, false positives or precision. It can be seen as an addition to the CLEAR MOT metric, providing a more detailed insight to the ratio of misses.

TABLE I: Properties of simulated sequences.

ID	Area	Targets	Duration	Scans	Environment
1	100m × 100m	6	98s	986	Empty field
2	100m × 100m	50	99s	995	Empty field
3	200m × 200m	6	97s	978	Empty field
4	100m × 100m	6	54s	547	Industrial
5	200m × 200m	6	92s	925	Industrial, Shops, Houses
6	100m × 150m	6	97s	974	Park

B. Data Sets

Three data sets were utilized for evaluation and parameter optimization. The first two are used for a quantitative evaluation against ground truth data. The third data set consists of scans recorded from our MAV setup during two flights in a large courtyard. Due to missing ground truth for the latter, we utilize our filtering application for a qualitative evaluation.

1) InLiDa: The Indoor LiDAR Dataset *InLiDa* [7] consists of six hand labeled sequences captured using a Velodyne VLP-16 in an indoor environment. The sequences have a total duration of 501 seconds and contain 4823 scans. The sensor is placed in a fixed location in a corridor or a hall. Up to eight dynamic objects—seven humans and one robot—are simultaneously visible and labeled at point-level. The data set provides challenging situations with occlusions, groups of close objects moving in the same direction, rapid velocity changes and other dynamic objects, like doors.

2) Simulated: Additionally, we simulated the Velodyne VLP-16 in a set of diverse outdoor settings to generate another data set (Fig. 4). For simulation, we used Gazebo [24] and adapted a Velodyne VLP-16 simulator [25] to generate organized point clouds. The data set provides sequences with a varying number of dynamic persons within static environments with different amounts of clutter and distractions (Tab. I). Our simulated persons avoid obstacles, change their velocities from $3.5\,\mathrm{km/h}$ to $12.5\,\mathrm{km/h}$, and pause from time to time. The environments are limited to a distance of up to $140\,\mathrm{m}$ to the sensor in the center at a height of $2.5\,\mathrm{m}$. This implies that measurements of target objects do get very sparse or disappear completely due to occlusions or objects leaving the measurement range of the sensor. We recorded a data set using a static sensor and another set of sequences with a dynamic sensor moving on a trajectory with the shape of an 8 within a radius of about $4\,\mathrm{m}$ around the center of the environment. Ground truth labels are provided at point-level.

3) Real World Data: Our own data set was recorded during flights of a piloted MAV in the courtyard of the *Landesbehördenhaus* (LBH) in Bonn (Fig. 4). In the first sequence, only the pilot is visible to the sensor as a dynamic object. He moves in a slow pace within an area of about $7\,\mathrm{m} \times 7\,\mathrm{m}$.

The second sequence was recorded with four visible humans. They are walking and running, crossing each other's paths, changing speeds and occluding each other. The MAV and sensor are flying with velocities of up to $20\,\mathrm{km/h}$.

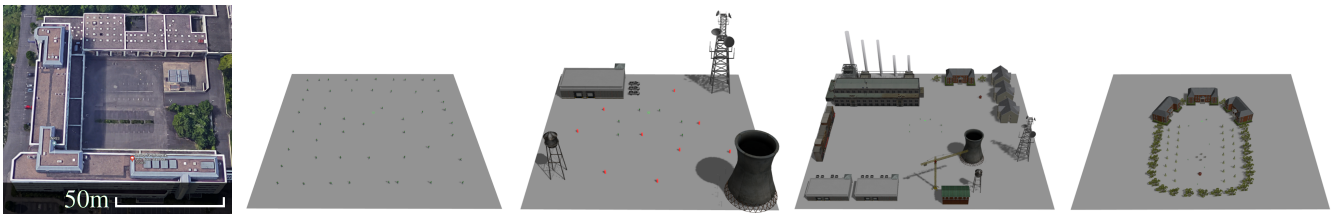

Fig. 4: Exemplary views on environments the data sets were recorded in. Left to right: Real-World: Aerial photo [4] of the LBH in Bonn, Germany. Simulated: *Empty field* sequences serve as a baseline providing easier circumstances. *Industrial setting* with six dynamic persons and nine static distractors in red. *Apartments, industry and shops* containing more natural distractors and occluders on a wider field. *Park* with buildings, fountains and a variety of trees providing numerous possibilities for short-term occlusions.

C. Parameter Optimization

For parameter optimization, we used *hyperopt* [26], a distributed asynchronous hyperparameter optimization library. It utilizes the Tree of Parzen Estimators [27] to optimize parameters in a specified search space by minimizing a cost function depending on the given parameters—e.g. object size and covariance threshold.

The cost function we utilized is defined by

$$costs = 1 - MOTA. \tag{3}$$

The costs are equal to zero for a perfect MOTA (Eq. 2) of 1.0 and rising as the MOTA decreases.

D. Quantitative Results

For quantitative evaluations, we start by comparing the results of our approach to the results reported in [7]. For all evaluations of our method, we only use those hypotheses classified as dynamic at least once.

The *InLiDa Tracking* approach concentrates on multi person tracking. It utilizes global Ensemble of Shape Functions (ESF) descriptors [28] on extracted point clusters and classifies them using random forests into the classes *Person* and *Not person* to generate person detections. For tracking, existing hypotheses are matched to their closest detections within a search radius of 0.5m and propagated utilizing a circular velocity buffer.

The task for evaluation was to track humans only, distinguishing them from the dynamic robot present in four of six sequences. They evaluated their approach by training parameters on one InLiDa sequence and testing on the remaining. Each sequence was used for training once.

The *InLiDa Tracking* achieves a total MOTA of -0.213, our approach evaluated in the same way achieves a total MOTA of 0.071. The resulting MOTAs are rather low, considering that using no tracker at all results in a MOTA of zero. One possible reason is the utilized evaluation procedure. Training on one sequence only increases the risk of overfitting the parameters. Additionally, the methods have difficulties to distinguish between the robot and humans, if there is no robot present in the training sequence. Applying the methods to other unseen sequences, with an attendant robot, yields bad results.

TABLE II: Results of quantitative evaluation.

Data Set	MOTA	MOTP	MT	PT	ML
InLiDa	0.562	0.108m	0.49	0.43	0.08
Simulated static sensor	0.677	0.044m	0.75	0.25	0.0
Simulated dynamic sensor	0.533	0.033m	0.57	0.40	0.03

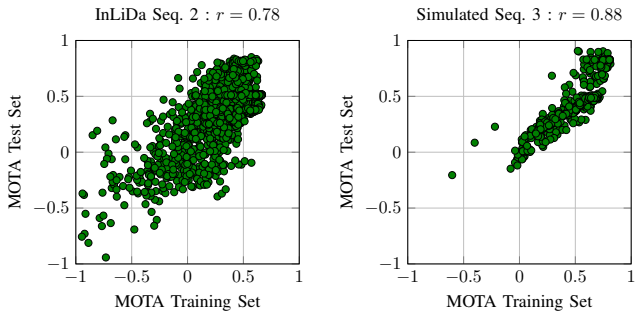

Fig. 5: Plots visualizing the MOTA on the training set against the MOTA on the test sequence during parameter optimization. The titles report the sequence utilized as the test set and the Pearson correlation coefficient r.

We evaluated our method a second time on the InLiDa. Due to the limited size of the data set, we perform a Leave-one-out cross-validation (LOOCV). For this, we split the data set containing n sequences into a training set of size $n-1$ for parameter optimization and a test set consisting of the left out sequence. The test set serves the purpose of evaluating the method's performance and ability to generalize on unseen data. This process is successively repeated n-times, each time leaving another sequence out. Table II presents the results of this evaluation on the InLiDa and the simulated data sets.

To get a better insight into the method's ability to generalize to unseen data, we plot the MOTAs $\{x_1, \ldots, x_n\}$ on the training set against the MOTAs $\{y_1, \ldots, y_n\}$ on the test set computed during n runs of the optimization process. For each sequence, we additionally compute the Pearson correlation coefficient r defined by

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4}$$

with $\bar{x}$ and $\bar{y}$ as the means of the MOTAs on the training

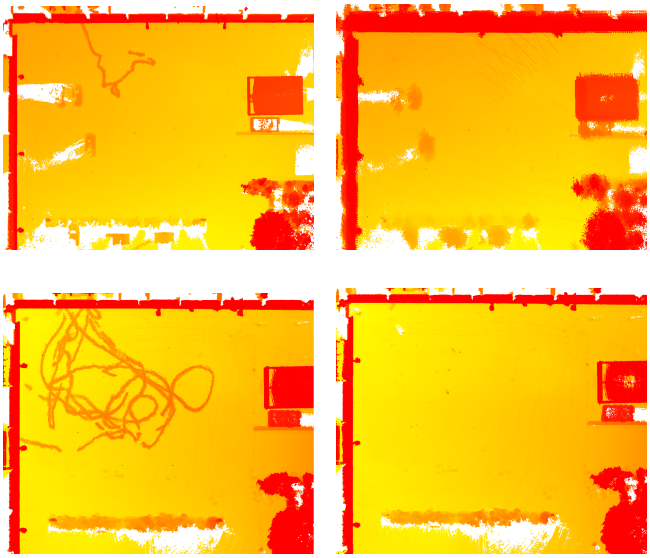

Fig. 6: Top down views of two mapped sequences. Color encodes height (yellow low, red high). Left: Raw measurements mapped into one coordinate frame. Measurements on dynamic objects result in artifacts visible as orange lines on the yellow ground. Right: Same scans with dynamic objects filtered out. Top: Pilot only. Bottom: Four persons.

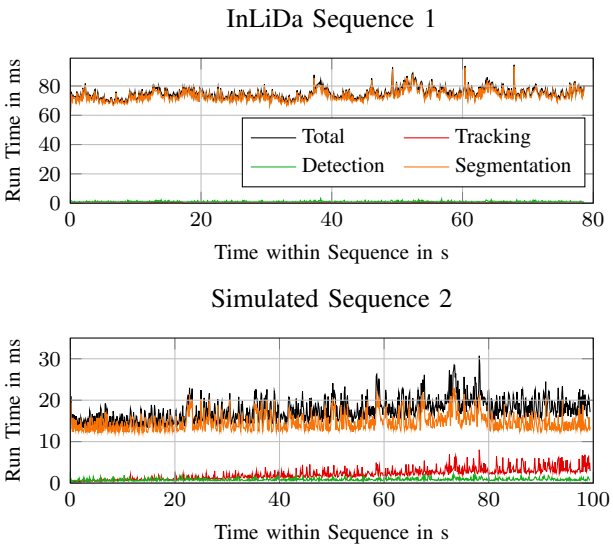

Fig. 7: Run time in milliseconds of method and modules per scan on exemplary sequences.

and test set, respectively. The correlation coefficients during the optimizations on the InLiDa sequences range from 0.68 to 0.83, on the simulated sequences from 0.47 to 0.90. We present the plots corresponding to the test sequences with the median correlation coefficient per data set in Fig. 5.

E. Qualitative Results

For a qualitative evaluation, we filter the sequences recorded in the courtyard of the LBH as described in Sec. III-E. Measurements on dynamic objects create artifacts. Our method is able to filter out most points corresponding to those objects, even for noisy mapping results present in the first sequence (Fig. 6).

F. Run Time

Finally, we inspect the real-time capability of our method by plotting the run time per scan for two example sequences (Fig. 7). We measured the time for each module—Segmentation, Detection and Tracking—to process incoming data. We assume a sequential procession of the data on one CPU core. In practice, all modules are able to process the data of the next time step directly after processing the current data. The method was executed on the hardware of our MAV consisting of an *Intel Core i7-6770HQ* CPU and 32 GB of RAM.

We chose the most demanding sequences from the InLiDa and the simulated data set: a sequence in the hall with a wall close to the sensor resulting in large kernel sizes during segmentation and the simulated sequence with 50 persons present. Our method processes the data before the next scan is available after 100 ms.

G. Discussion

For better results, the application of a more sophisticated but still efficient tracking approach should be investigated. Complex movement patterns of humans can only be tracked to a limited extent by the utilized Kalman filter.

Furthermore, the information about occlusions could help to adjust the time an occluded hypothesis is retained. The same information can be utilized to more robustly distinguish between static and dynamic objects. During partial occlusions, the bounding box of the occluded object changes its shape and size, as the object is only partially visible to the sensor. In some instances this is interpreted as a movement of the occluded object. Incorporating this information would counteract false classifications and, hence, enable the approach to filter dynamic objects more precisely.

Lastly, the run time of the proposed segmentation method depends on the distance to the environment. Close objects combined with a large specified target width increase the run time. Replacing the median filters by an approach that compares the measured distance of a point to the distances of two neighboring background points could generate a similar segmentation while being computationally more efficient.

V. Conclusion

We implemented a method for real-time multi object tracking of small objects in the sparse point clouds generated by a Velodyne VLP-16 on the limited hardware of a MAV. For this, we proposed a novel segmentation approach to segment point groups of a specified width range in single scan rings and implemented efficient algorithms for detection and tracking utilizing the structure of the data. We evaluated our approach on simulated and real in- and outdoor data sets achieving results comparable to the state of the art. As a practical application, we filter data corresponding to dynamic objects and map the static part of the scene.

REFERENCES

[1] D. Droeschel and S. Behnke, "Efficient continuous-time SLAM for 3D lidar-based online mapping," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2018.

[2] M. Beul, D. Droeschel, M. Nieuwenhuisen, J. Quenzel, S. Houben, and S. Behnke, "Fast autonomous flight in warehouses for inventory applications," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3121–3128, 2018.

[3] S. J. Kim, G. J. Lim, J. Cho, and M. J. Côté, "Drone-aided healthcare services for patients with chronic diseases in rural areas," *Journal of Intelligent & Robotic Systems (JINT)*, vol. 88, no. 1, pp. 163–180, 2017.

[4] Google. Landesbehördenhaus Bonn. [Online]. Available: https://goo.gl/maps/3tNpQEdmqTy

[5] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3D range data." in *AAAI*, vol. 10, 2010.

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

[7] C. Romero-González, A. Villena, D. González-Medina, J. Martínez-Gómez, L. Rodríguez-Ruiz, and I. García-Varea, "InLiDa: A 3D lidar dataset for people detection and tracking in indoor environments," in *Proc. of the Int. Conference on Computer Vision Theory and Application (VISSAPP)*, 2017.

[8] H. Farazi and S. Behnke, "Online visual robot tracking and identification using deep LSTM networks," in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2017.

[9] D. Maturana and S. Scherer, "3D convolutional neural networks for landing zone detection from lidar," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2015.

[10] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," *arXiv preprint arXiv:1602.00991*, 2016.

[11] A. Milan, S. H. Rezatofighi, A. R. Dick, I. D. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks." in *AAAI*, 2017.

[12] H. Possegger, T. Mauthner, and H. Bischof, "In defense of color-based model-free tracking," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[13] F. Moosmann and C. Stiller, "Joint self-localization and tracking of generic objects in 3D range data," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2013.

[14] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, "Motion-based detection and tracking in 3D lidar scans," in *Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2016.

[15] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with mobile robots using sample-based joint probabilistic data association filters," *The International Journal of Robotics Research*, vol. 22, no. 2, pp. 99–116, 2003.

[16] K. Granström, L. Svensson, S. Reuter, Y. Xia, and M. Fatemi, "Likelihood-based data association for extended object tracking using sampling methods," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 1, pp. 30–45, 2018.

[17] J. Dequaire, P. Ondrúška, D. Rao, D. Wang, and I. Posner, "Deep tracking in the wild: End-to-end tracking using recurrent neural networks," *The International Journal of Robotics Research*, 2017.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] D. Droeschel, J. Stückler, and S. Behnke, "Local multi-resolution surfel grids for MAV motion estimation and 3D mapping," in *Proc. of the Int. Conference on Intelligent Autonomous Systems (IAS)*, 2014.

[20] M. Schwarz, D. Droeschel, C. Lenz, A. S. Periyasamy, E. Y. Puang, J. Razlaw, D. Rodriguez, S. Schüller, M. Schreiber, and S. Behnke, "Team NimbRo at MBZIRC 2017: Autonomous valve stem turning using a wrench," *Journal of Field Robotics*, vol. 36, no. 1, pp. 170–182, 2019.

[21] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[22] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *Journal on Image and Video Processing*, vol. 2008, p. 1, 2008.

[23] Y. Li, C. Huang, and R. Nevatia, "Learning to associate: Hybrid-boosted multi-target tracker for crowded scene," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[24] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator." in *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS)*, 2004.

[25] Dataspeed, "Velodyne simulator," 2018. [Online]. Available: https://bit.ly/2x9ybfm

[26] J. Bergstra, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms," in *Proceedings of the 12th Python in Science Conference*, 2013.

[27] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in Neural Information Processing Systems*, 2011, pp. 2546–2554.

[28] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2011.