DR 1.4: Sensing, mapping and low-level memory IV – Long term persistence

Tomáš Svoboda[*], Renaud Dubé[†], Abel Gawel[†], Hartmut Surmann[‡], Kalle Knipp[‡], Cesar Cadena[†], ..., and the TRADR consortium

[*]*CTU in Prague, Czech Republic*
[†]*ETH, Zürich, Switzerland*
[‡]*Fraunhofer IAIS, Sankt Augustin, Germany*
⟨svobodat@fel.cvut.cz⟩

Project, project Id:	EU FP7 TRADR / ICT-60963
Project start date:	Nov 1 2013 (50 months)
Due date of deliverable:	Month 50
Actual submission date:	
Lead partner:	CTU
Revision:	draft
Dissemination level:	PU

We report progress achieved in Year 4 of the TRADR project in WP1: *Persistent models for perception*. It describes the essential robot (UGV) perception functionalities and and new algorithms for realtime 3D mapping with loop closure and merging UGV and UAV maps.

Executive Summary

The key objective of WP1 is to provide sensory data from all involved robots registered in space and time, to keep creating and updating robot centric representations, and ground them into the world coordinate frame. The obtained representations are furnished to other WPs, which maintain higher level situation awareness.

Role of robot perception and metrical mapping in TRADR

In this report, robot perception is intended to be the robot ability to analyze its neighborhood and act accordingly. Terrain recognition is essential for robot locomotion regardless whether the robot is teleoperated or moves autonomously. It is desirable the robot overcomes obstacles reasonably – fast, safe, consuming less power and reducing cognitive load of a human operator.

The metrical mapping serves as the very basis for modeling the world. It is also the basis for sharing information between robots and also among several sorties and even missions. Real-time multi-robot 3D mapping essentially speeds up reconnaissance hence, situation awareness.

Persistence

Persistence in WP1 is addressed mainly by re-using the data in creating an enviroment model. The 3D metric map serves as the main basis for multi-modal (data), multi-source (robots), multi-level (abstraction, decisions) registration. In WP1, we are working on robust methods for merging partial 3D maps. The merging challenges include weak data overlap, dynamic changes in scenes, large displacement of local coordinate systems. The newly developed compact map representation which is learned from the data allows for database-like map handling - fast querying and matching. The terrain perception and robot control algorithms use machine learning techniques in a quest of gaining experience from operator-robot interactions.

Contribution to the TRADR scenarios and prototypes

The new 3D multi-robot mapping algoritm and map representaiton are important step towards multi-robot collaboration (WP4) and models for acting (WP2). New work on reducing drift of dead reckoning by a terrain classication helps in sensory deprived environments. The new robot model contributes to simulations needed to the models for acting (WP2), multi-robot

collaboration (WP4) and also support the human-robot teaming (WP5).

1 Tasks, objectives, results

1.1 Planned work

In Year4, WP1 planned to investigate "Sensing, mapping and low-level memory IV – Long term persitence" (Milestone MS1.4). The work was divided into two tasks:

- Scene part and object reconition (T1.7)
- Robot centric metrical maps and models storage – Long-term persistence, central yet distributed mission memory (T1.8)

1.2 Addressing reviewers' comments

The WP1 was evaluated as outstanding and we were encouraged to continue.

1.3 Actual work performed

1.3.1 Task T1.7 (Scene part and object recognition IV – Long-term persistence)

Terrain classification for reducing drift in dead reckoning

To improve localization of the UGV, we've focused on the basic odometric system of the robot. Its accuracy suffers while traversing vertical obstacles and that makes loop closures and map merging more difficult. We've proposed additional models for tracked odometry that improve localization in such conditions. The proposed method has been experimentally verified during the whole year indoors and outdoors, including data from T-EVAL. The experimental dataset is released[1] for the robotic community and the results are about to be submitted to a IEEE journal. Since the results were obtained in the end of the project, there is no time to fully integrate it without compromising the rest of the system. However, the results should be easily transferable to other tracked vehicles and thus contributing outside the TRADR project.

Adaptive traversal

The simulation model of the TRADR robot was improved and offered to the open-source community of the Gazebo simulator[2], so that everyone can profit from the work done. We also compared this model with other methods of simulation of tracked vehicles to be sure we are using the most suitable

[1] `http://ptak.felk.cvut.cz/tradr/visuals/bagfile_crawler/`

[2] `https://bitbucket.org/osrf/gazebo/pull-requests/2652/added-support-for-tracked-vehicles/diff`

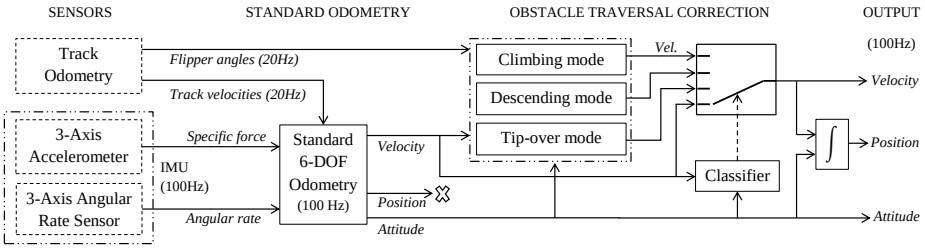


Figure 1: Block diagram of the proposed track odometry system. Standard 6-DOF odometry with input from proprioceptive sensors computes intermediate velocity, position and robot attitude. These values are corrected based on classifier decision to either keep the standard output or to switch to one of the proposed obstacle traversing modes.

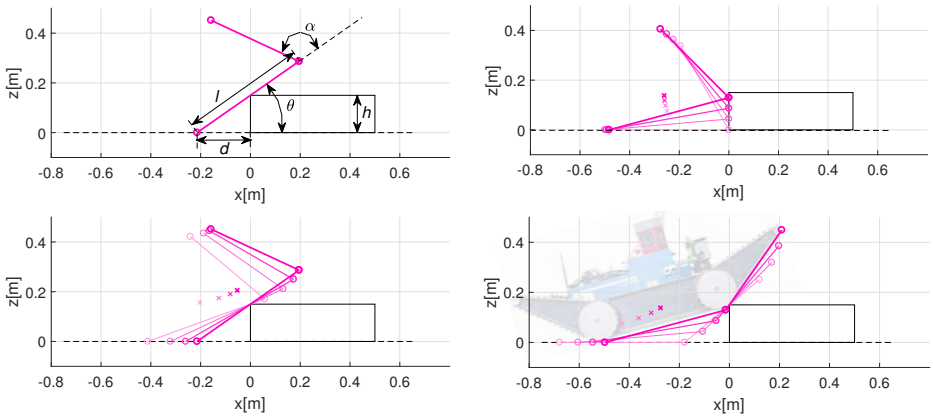


Figure 2: Kinematic model for obstacle climbing. From top left to bottom right: model parameters, climbing with flippers retracted (second and third figure), climbing using flippers. The violet x symbol denotes motion of the origin of the robot frame.

simulation model. The adaptive traversal ability of the robot was also improved by utilizing convolutional neural networks and a different training paradigm (and utilizing the improved simulation model). As an alternative path towards fully autonomous flipper and robot control, we investigated an end-to-end learing approach (see Annex 2.4). Results in simulations and early real robot experiments are promissing. A new RealSense RGB-D camera should bring faster flipper response (see Annex 2.6).

We also improved the user interface used for flipper control and teleoperation based on the comments of the end-users. Feedback from teleoperation is now collected and recorded to further improve the AT algorithm from mission data.

We created the user-facing documentation for adaptive traversal, flip-

per control and pan-tilt cameras control. Formerly, the adaptive traversal algorithm was mutually exclusive with mapping of the environment. We removed this limitation. Work was also done on integrating AT with the path planner, but some work reamins to be done in this domain.

We released an open-source component for easy integration of the Tensorflow framework with robots running the ROS middleware[3].

A dataset for visual scene classification and segmentation

A huge dataset were collected during the TRADR project. We selected 750 bag-files from the test missions and extracted 39.000 images from the robot omnicamera (image was captured every 2 meters of robot traverse) and we published this dataset on the web [4]. We manually annotated subset of 8000 images. Each image annotation contains image class information (20 classes) in the Pascal-VOC format. We also manually segmented 320 images in Kitti dataset format (20 classes) and developed tools for annotation and viewing of the dataset. All these files are accessible on the public website. The dataset was adopted for testing of the neural nets suitable for recognition of objects in robot visual space. We tested 3 state of the art networks for image classification, namely TransferNet [30], Yolo [62] and Mask-RCNN [28]. We trained Transfernet on 60.000 images from MS-Coco (pixel-wise annotation) and 7.000 images from our dataset (image-class annotation). As TransferNet allows transfer learning (transfer semantic segmentation from from MsCoco to novel classes in our dataset) we chose 20 classes that are desirable in rescue robotics (vehicles, wiring, fuse boxes, barrels, ruins etc.). During the test stage we faced very low accuracy (5%) so we switched to the Yolo. The algorithm was able to detect bounding boxes of the objects, but it was restricted to the pretrained set of classes that has small overlap with rescue robotics (vehicles, persons). The Mask-RCNN allows to segment each instance of the object class in the scene. Unfortunately, the Mask-RCNN is trained on MS-Coco dataset, that contains only few useful classes for the rescue robotics.

Active 3D mapping

This is a kind of work for future, while addressing upcoming a new type of depth sensors – Solid state lidars. We proposed an active 3D mapping method for depth sensors, which allow individual control of depth-measuring rays (see Annex 2.3). The method simultaneously (i) learns to reconstruct a dense 3D occupancy map from sparse depth measurements, and (ii) optimizes the reactive control of depth-measuring rays. The scheme is depicted on Figure 3. To make the first step towards the online control optimization,

[3] `https://github.com/tradr-project/tensorflow_ros`

[4] `http://ptak.felk.cvut.cz/tradr/visuals/tradr-seg-dataset/`

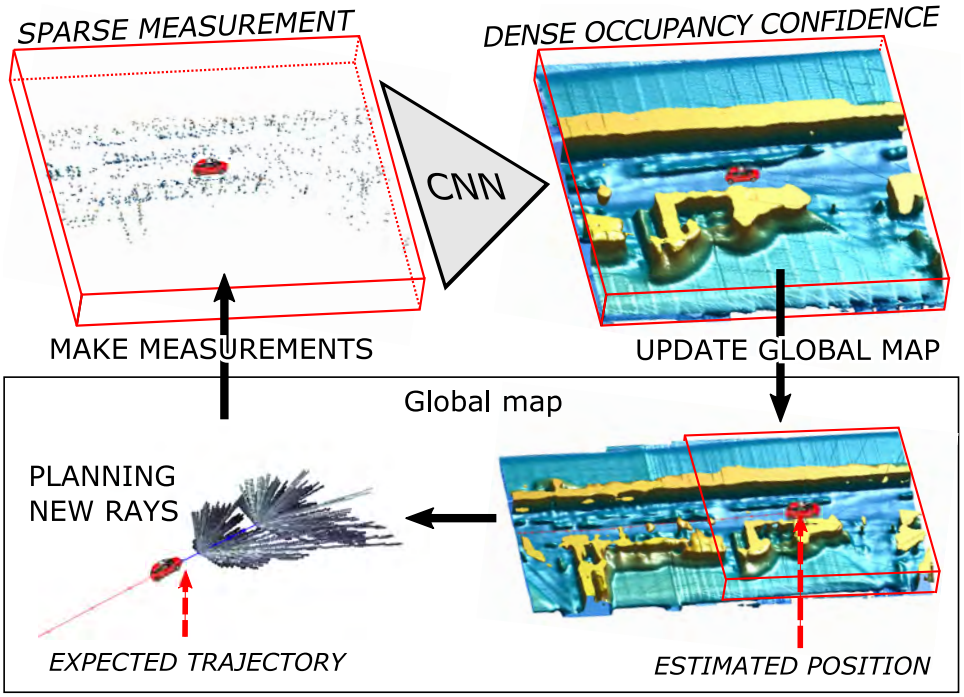


Figure 3: Iteratively learned deep convolutional network reconstructs local dense occupancy map from sparse depth measurements. The local map is registered to a global occupancy map, which in turn serves as an input for the optimization of depth-measuring rays along the expected vehicle trajectory. The dense occupancy maps are visualized as isosurfaces.

we propose a fast prioritized greedy algorithm, which needs to update its cost function in only a small fraction of possible rays. We also derived the approximation ratio of the greedy algorithm. An experimental evaluation on the subset of the KITTI dataset demonstrates significant improvement in the 3D map accuracy when learning-to-reconstruct from sparse measurements is coupled with the optimization of depth measuring rays. Evaluation code was released[5]. We also interfaced the simulating solid state lidar from Kitti data as a Lidar-gym environment[6] – compatible with the known AI-gym.

3D SLAM and Environment Reconstruction

Over the course of the project, the TRADR consortium progressively developed its SLAM framework for 3D LiDARs. To this date, each Unmanned Ground Vehicle (UGV) of the TRADR system uses this framework for registering LiDAR scans using Iterative Closest Point (ICP) and for creating a

[5] `https://github.com/salanvoj/active-3d-mapping`
[6] `https://gitlab.fel.cvut.cz/rozsyzde/lidar-gym`

3D point clouds of the environment. In the previous year, we introduced a novel place recognition technique for 3D point clouds [12]. State-of-the-art localization performances were achieved thanks to the principle of segment extraction and matching. As a reminder of this approach, an updated architecture is presented in Section 1.3.2.

This year, we presented the integration of the aforementioned pose-graph Simultaneous Localization and Mapping (SLAM) and loop-closure techniques in a complete multi-robot SLAM system (see Annex 2.7). Our system enables to estimate in real-time the trajectories of multiple robots and to build an unified 3D map representation. To the best of our knowledge, this was the first proposed solution to the online multi-robot SLAM problem for 3D LiDARs. We demonstrated the system's real-time capability by simultaneously processing data, from three UGVs, collected during the TRADR Evaluation 2016 at the Gustav Knepper powerplant[7]. We also illustrated how segment-based localization can effectively summarize the 3D structure to a handful of descriptors and segment centroids. This highly compressed data can easily be transmitted to a central computer for making global associations, even under limited communication bandwidth.

In this last year of the TRADR project, we further contributed to Task T1.7 by proposing a complete SLAM solution addressing important challenges related to multi-robot systems [68] and search & rescue operations:

- Large-scale and long-term missions
- Real-time performance
- Limited communication bandwidth
- Providing end-users 3D visual feedback

To this end, we built on our previous works by proposing *SegMap*: a novel 3D *map representation* solution to SLAM for multi-robot systems (see Annex 2.9). As illustrated in Figure 4, the core idea behind this approach is to use a unique data-driven descriptor to simultaneously achieve all three tasks of robot localization, 3D structure reconstruction, and semantics extraction. This reconstruction capability is particularly interesting for navigation tasks and for providing visual feedback to end-users such as robot operators in search and rescue scenarios. To the best of our knowledge, this is the first work on robot localization proposing to reuse the extracted features for reconstructing environments in three dimensions and for extracting semantic information.

In essence, a Convolutional Neural Network (CNN) is used to compress the 3D segments extracted in the vicinity of the robot into a descriptor of dimension 64x1. This descriptor can directly be used for localization through

[7] A video of this experiment is available at `https://youtu.be/JJhEkIA1xSE`.

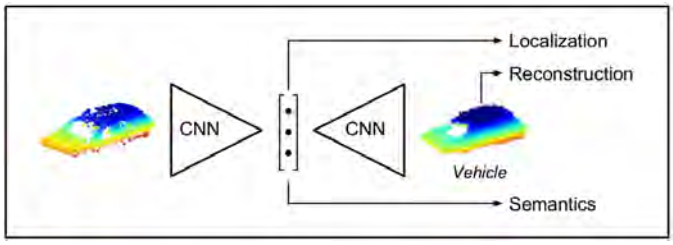


Figure 4: The data-driven *SegMap* descriptor can be used to localize, reconstruct the 3D structure, and extract semantic information.

efficient k-Nearest Neighbors (k-NN) descriptor retrieval in low dimensional space. The same descriptor can be fed to a second CNN for reconstructing the segment and to a fully connected network that can extract semantic information. When performing retrieval for localization, this descriptor offers increase of area under the ROC curve of 28.3% over current state of the art using eigenvalue-based features [12, 80]. The use of these compact segment-based features facilitates low computational, memory and bandwidth requirements, and therefore makes the approach appropriate for real-time use in both multi-robot and long-term applications.

We evaluated this approach on two multi-robot datasets collected at the Gustav Knepper powerplant and the Phoenix-West foundry (Figure 5). In both experiments, multiple inter-robot associations were made in real-time and on a single computer. As depicted in Figure 6, these global associations allows us to join the robot trajectories and to reconstruct the 3D structure of the disaster environments. Note that these point clouds are reconstructions which can be generated from the descriptors of the final compressed maps which weigh only 173kB and 123kB, respectively for the powerplant and the foundry. Moreover, a bandwidth of only 10kB/s would be sufficient to communicate the segment descriptors and centroids, a reduction of 20x over transmitting the segment point clouds. In future work, it would be interesting to compare this learning-based reconstruction capability to other 3D point cloud compression techniques [78, 31, 47].

The capacity of the network to extract semantic information is evaluated in a separate experiment with data collected in urban-driving environments. We demonstrated an accuracy of 85% at determining whether segments represent *vehicles*, *buildings*, or *others*. In the context of TRADR, it would be interesting to extend this work to distinguish between static and dynamic objects which are present in disaster environments. In this case, our localization module could rely on segments extracted from walls and stairs, adding robustness against dynamic changes.

Figure 5: Pictures of buildings where the multi-robot missions took place: the Gustav Knepper powerplant (left) and the Phoenix-West foundry (right).

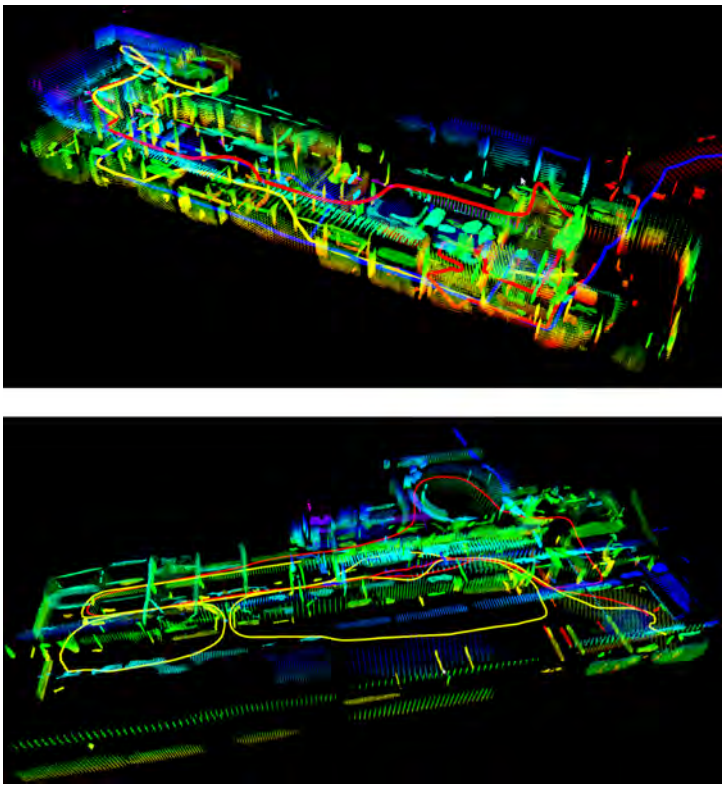


Figure 6: This figure illustrates a reconstruction of the buildings of the Gustav Knepper powerplant (top) and Phoenix-West foundry (bottom). The point clouds are colored by height and the estimated robot trajectories are depicted with colored lines.

1.3.2 Task T1.8 (Robot centric metrical maps and models storage IV – Long-term persistence, central yet distributed mission memory)

Control of an UAV in GPS-denied environments

This section describes the development of a state controller to move an UAV – in this case the Ascending Neo – automatically to a specific position. The controller is planned to be used indoors, so a high accuracy is needed. It is possible to specify the translation x, y, and z and also the rotation around the z-axis. A destination can be set if a map of the environment already exists. Then the UAV automatically navigates to this position. An overview on state-of-the-art techniques for modelling and controlling a UAV is given in section 1.4 of this document.

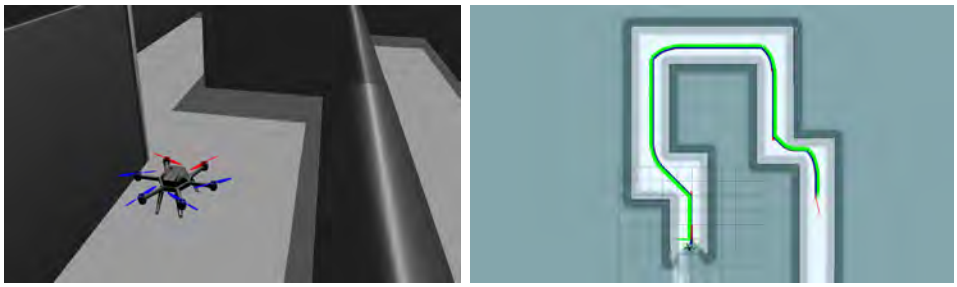


Figure 7: Test environment constructed in gazebo RotorS Simulator (Top). 2D map of the environment with path to the defined goal displayed in Rviz (Bottom)

The gazebo *rotorS Simulator* is used to execute an additional test of the state-controller. With this simulator it is possible to move an UAV in a three-dimensional space. A test environment is already set up in the simulator. This environment displays a corridor (see figure 7 - top). The visualization tool *rviz* of ROS is used to specify a destination point for the UAV. An appropriate path is computed if a map of the environment is available (see figure 7 - bottom). Now in the gazebo *rotorS Simulator* the UAV moves to the caclulated waypoints to reach the defined destination.

Consistent mapping of UAV und UGV pointclouds

We designed (see Annex 2.10)[8] a novel approach to build consistent 3D maps by merging UAV and UGV data. The UAV monocular camera data are used to generate 3D point clouds that are fused with the 3D point clouds generated by a rolling 2D laser scanner mounted on the UGV. The registration method is based on the matching of corresponding planar segments that are

[8] `https://www.youtube.com/watch?v=o969dUl2NOc`
`https://www.youtube.com/watch?v=xAVR5aFv8VY`

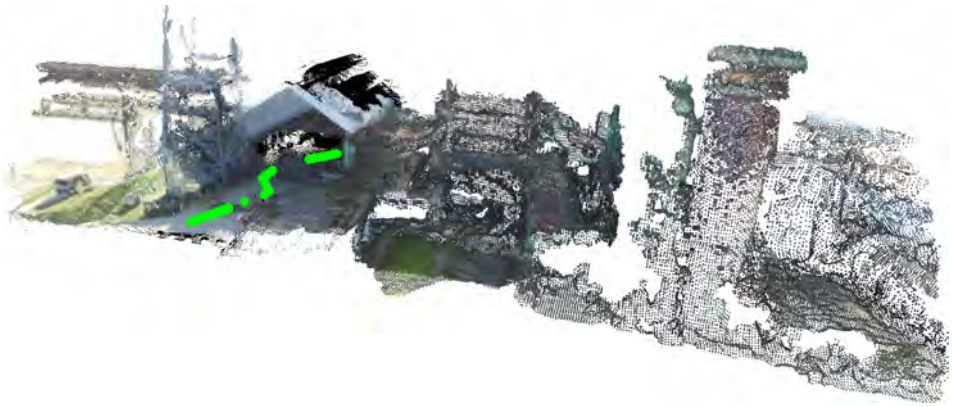


Figure 8: Registered point clouds of the globally optimized localization. Points of the laser point clouds were dyed with the color information of the global point cloud. Within the factory no color information could be extracted. The estimated trajectory is shown in green.

extracted from the point clouds. Based on the registration, we proposed a global optimization for localization. Apart from the structural information of the point clouds, it is important to mention that no further information is required for the localization. An example of merged maps is depicted on Figure 8.

Along these lines, we furthermore worked on an alternative approach of localizing the UGV LiDAR maps in 3D reconstructions generated by the UAV and report our results in WP 2. Here, we explored a different localization paradigm and go beyond the assumption of planar surfaces by using volumetric structural features of the environment.

Efficient Localization in 3D Point Clouds

The capabilities of the *SegMap* approach presented in Section 1.3.1 would not be possible without an efficient localization method. This year, we contributed to Task T1.8 by specifically addressing the real-time performance requirement of our localization module. In our previous work, we noted that normal estimation, segmentation, and geometric verification were often the most computationally expensive modules of our segment-based technique [12]. With this in mind, we proposed an efficient method for localization in 3D point clouds which is based on incremental solutions to the aforementioned modules (See Appendix 2.8). The efficiency of the method makes it suitable for applications where real–time localization is required and enables its usage on cheaper, low–energy systems.

The architecture of this incremental localization method is depicted in Figure 9. For the modules of normals estimation, segmentation and recog-

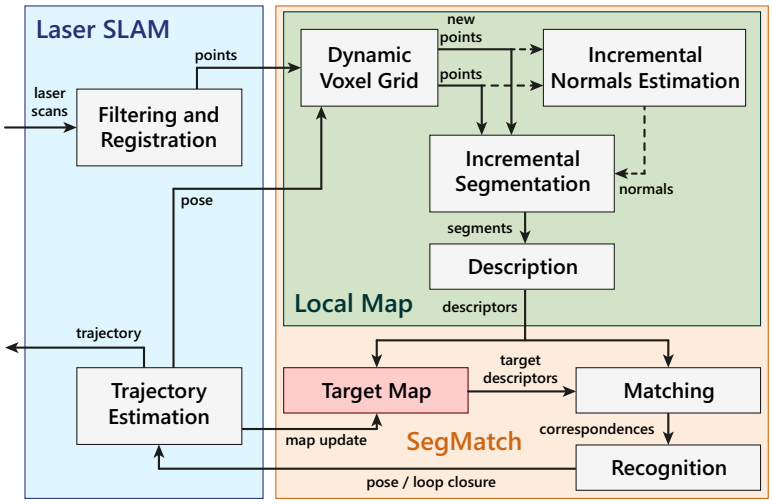


Figure 9: Architecture of our incremental segment matching pipeline and its interface with our pose-graph SLAM framework for 3D LiDARs.

nition, we reuse the processed information by retaining data structures that are likely to save computations in later localization attempts. We first developed a Dynamic Voxel Grid (DVG) which accumulates and filters a continuous stream of 3D point cloud and provides information about newly occupied voxels. This key information is leveraged for estimating normals by re-computing only those affected by newly occupied voxels and by caching information to incrementally compute covariance matrices. Incremental region growing segmentation is then performed by using only the newly occupied voxels as seeds and by merging with previously clustered points. An example of three segments being incrementally grown is depicted in Figure 10. This strategy enables us to robustly track segments between successive observations which offers multiple benefits to the overall framework. Amongst others, it allows us to perform efficient geometric verification by caching geometric consistencies.

We evaluated this method in a robot localization example in the Gustav Knepper powerplant. Overall, we note a speedup of x12.4 over the batch version of *SegMatch*. The expensive modules of normal estimation and segmentation benefited of speedups of x16.4 and x12.3 respectively. Due to the small dimension of the mapped building, both the batch and incremental geometric verification steps can be executed under 1ms. Note that, when considering larger environments, we observed a speedup of x14.2 for the incremental geometric verification algorithm. We also presented an experiment where data from five Velodyne HDL-64E sensors are processed in real-time on a single computer, in order to succesfully identify sufficient global associ-

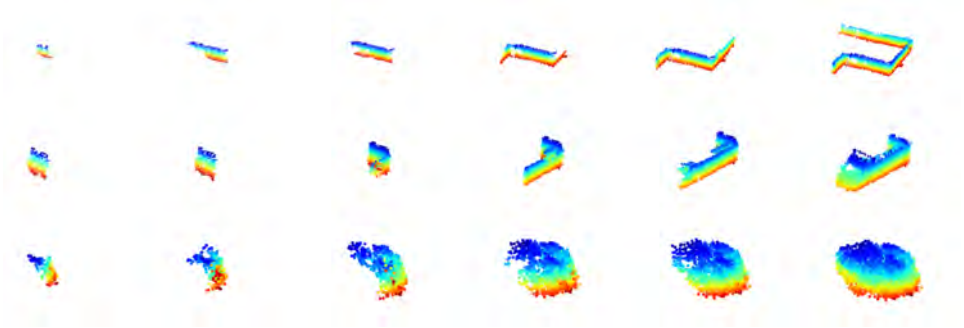


Figure 10: An illustration of three segments being incrementally grown over successive observations (from left to right).

ations for building an unified map[9].

Finally, our incremental segment-based localization algorithm is open-source available, along with our multi-robot SLAM back-end[10]. We provided multiple easy-to-run demonstrations, one of which uses data collected at the Phoenix-West Foundry during the TRADR Evaluation in 2015. Since we open-sourced our *SegMatch* library in November 2016, we received significant attention from the research community, with over 100 people following our github repository. In the last year, we supported multiple researchers, via email exchanges, github issues, and discussions at conferences, in applying our method to their own data. Some people even contributed back to the github repository by sharing impressive results which they achieved with the segment-based method. Needless to say, we are pleased to see that the science and systems developed during the TRADR project have a tangible and positive impact on the research community.

1.4 Relation to the state-of-the-art

Terrain classification for reducing drift in dead reckoning when climbing up and down

In previous years, we proposed several approaches to adapting robot morphology to terrain[89, 57, 24]; a computationally fast and plausible simulation of a tracked vehicle is presented in [57]. We also proposed method for localization in 6-DOF by fusing tracked odometry and complementary-filtering-based attitude estimation [63, 70, 37].

Our robotic platform uses flippers as well allowing it to climb stairs and overcome vertical obstacles. There are many more possible designs that allow obstacle traversal or step climbing; authors of [51] combine walking for hard soils and tracked motion for weak soils, in [27], wheel-track robot prototype

[9] A video demonstration is available at `https://youtu.be/cHfs3HLzc2Y`.

[10] The implementation is available at `https://github.com/ethz-asl/segmatch`.

is proposed and the work of [55] describes a novel robot body configuration allowing transformation between a "claw" mode intended for climbing stairs and a "wheel" mode for flat terrain.

In [41], authors analyze capability of a tracked robot to climb stairs. They compare different configurations of tracks and analyze climbing motion dynamics. Effect of changing payload mass held by tracked robot with flippers is investigated in [40].

Authors of [38] demonstrate that slippage of tracks can be estimated by an extended Kalman filter by observing trajectory data of a tracked vehicle. Alternative approach to track odometry via identification of instantaneous centers of rotation is proposed by [43]. Simplified slippage model which also benefits from angular rate sensor is presented in [14, 50]; the latter also extends the odometry to 6-DOF. An approach of utilizing external position reference to identify kinematic model parameters is presented in [58]. All these works deal with the problem of track slippage while turning and with the associated uncertainty in velocity measurements. The extension we propose can be combined with their results by identifying moments of vertical motion and temporarily switching to our model during those instances.

Learning for active 3D mapping

High performance of image-based models is demonstrated in [72], where a CNN pooling results from multiple rendered views outperforms commonly used 3D shape descriptors in object recognition task. Qi et al. [60] compare several volumetric and multi-view network architectures and propose an anisotropic probing kernel to close the performance gap between the two approaches. Our network architecture uses a similar design principle.

Choy et al. [8] proposed a unified approach for single and multi-view 3D object reconstruction which employs a recurrent neural architecture. Despite providing competitive results in the object reconstruction domain, the architecture is not suitable for dealing with high-dimensional outputs due to its high memory requirements and would need significant modifications to train with full-resolution maps which we use. We provide a comparison of this method to ours.

Model-fitting methods such as [69, 73, 65] rely on a manually-annotated dataset of models and assume that objects can be decomposed into a predefined set of parts. Besides that these methods are suited mostly for man-made objects of rigid structure, fitting of the models and their parts to the input points is computationally very expensive; e.g., minutes per input for [69, 73]. Decomposition of the scene into plane primitives as in [46] does not scale well with scene size (quadratically due to candidate pairs) and could not most likely deal with the level of sparsity we encounter.

Geometrical and physical reasoning comprising stability of objects in the scene is used by Zheng et al. [87] to improve object segmentation and

3D volumetric recovery. Their assumption of objects being aligned with coordinate axes which seems unrealistic in practice. Moreover, it is not clear how to incorporate learned shape priors for complex real-world objects which were shown to be beneficial for many tasks (e.g., in [54]). Firman et al. [20] use a structured-output regression forest to complete unobserved geometry of tabletop-sized objects. A generative model proposed by Wu et al. [84], termed Deep Belief Network, learns joint probability distribution $p(\mathbf{x}, y)$ of complex 3D shapes $\mathbf{x}$ across various object categories y.

End-to-end learning of stochastic motion control policies for active object and scene categorization is proposed by Jayaraman and Grauman [33]. Their CNN policy successively proposes views to be captured with RGB camera to minimize categorization error, and they use a look-ahead error as an unsupervised regularizer on the classification objective. Andreopoulos et al. [1] solve the problem of an active search for an object in a 3D environment. While they minimize the classification error of a single yet unknown voxel containing the searched object, we minimize the expected reconstruction error of all voxels. Also, their action space is significantly smaller than ours because they consider only local viewpoint changes at the next position while the SSL planning chooses from tens of thousands of rays over a longer horizon.

Modelling and position controlling of an UAV

Each multicopter has already a control algorithm to ensure stable flight conditions. However, often this kind of control is not usable for navigating to a specific position. To really control the translation of an UAV, it is necessary to develop an individual controller. For designing a suitable controller, it is useful to establish a mathematical model of the system first. In [77], [42], and [4] the design process for a mathematical model of a quadrocopter is presented. This kind of models are adjustable for other types of multicopters like Hexacopter [2] or Octocopter. To navigate to a specific position with an UAV, different kinds of control algorithms can be used like PID- or state-controller. Some different kinds of controllers a presented in [67].

Mapping of UAV and UGV point clouds

A basic prerequisite for many tasks, such as navigation, mapping or cooperation of UAV and UGV, is the robot localization. When working with three-dimensional point clouds, the registration is significantly affected by the success of an exact localization [29]. Due to the aim of this work, to localize the UAV and UGV together in a global map, a registration method has to be found that can handle point clouds from different sources. In this context, it is important that the methods for registration as well as the generation of vision-based point clouds can be combined.

Vision-based SLAM

In order to perform visual odometry, only keypoints are selected, which make a robust correspondence search possible. While some methods compute complex features ([34, 11]), new developments increasingly use image points directly ([53, 15, 21]). Direct approaches have the advantage that they are not reduced to certain feature points but can exploit all image points to determine the odometry and depth values and thus provide more dense reconstructions of the environment. Depending on how many image points are utilized, the approaches can be divided into dense and semi-dense methods.

An example of a semi-dense approach is the SVO algorithm, which is presented in the work of [21]. The method uses point features, but these are not explicitly extracted. Rather they are an implicit result of a direct motion estimation. The initialization of the pose is achieved by minimizing the photometric error. LSD-SLAM [16] provides another direct approach. Based on the odometry method of [15], the algorithm generates globally consistent maps of the environment by means of graph optimization in large-area environments. Similar to the SVO algorithm, a probabilistic representation of the depth map is also used here to model inaccuracies. [48] also uses a probabilistic approach, but the method is based on a feature-based monocular SLAM system ([49]). Furthermore, in contrast to SVO and LSD-SLAM, the depth values of a reference image are not filtered over many individual images, but only key images are used for the reconstruction.

[71] presents one of the first real-time methods and provides dense reconstructions with a monocular camera. The tracking of the camera is based on the approach of [34]. The reconstruction is carried out using several key images. By expanding to several images, regions that would be hidden in two images or would be outside the corresponding image can also be reconstructed with a higher probability. DTAM ([53]) also provides dense reconstructions in real-time. In order to estimate the depth values, the method performs a global energy reduction over many individual images. REMODE ([59]) is a method for the reconstruction of dense point clouds, which integrates a Bayesian estimate into the optimization process. By modeling uncertainties of measurement for each pixel, regularization can be controlled precisely and inaccuracies in the localization can be reduced. Real-time capability is achieved through a CUDA-based implementation. For the pose estimation, the method of [21] is used. One of the recent developments of dense reconstructions is DPPTAM [9]. The approach reconstructs high textured regions with a semi-dense approach and low textured regions by approximation of surfaces. Thereby the assumption is made that homogeneously colored image regions form a plane, which can be determined by superpixels ([18]).

The procedures described so far fall under the category of online procedures, i.e. they are real-time capable and can deliver first results during

camera recording. In contrast, offline procedures require all collected recordings in advance and then carry out the corresponding calculations. In [23], a pipeline for reconstruction is presented that combines all necessary processing steps in a software framework called MVE. The framework is also capable of reconstructing texturized surfaces. A nice collection of relvant frameworks gives the "Awesome 3D reconstruction list". [11]

SLAM with 3D Point Clouds

In last year's TRADR deliverable we presented the state-of-the-art techniques for performing localization in 3D point clouds. This overview is also available in Dubé et al. [12] and Cadena et al. [7]. In this section we review previously proposed methods related to the core aspects of our approach: multi-robot SLAM, efficient point cloud segmentation, and data-driven methods for 3D point clouds. Note that this literature review combines material from our publications listed in Annexes 2.7, 2.8, and 2.9.

Multi-robot SLAM A thorough survey on multi-robot SLAM can be found in [68]. There is a significant amount of works proposing solutions to the SLAM problem for multiple robots equipped with cameras or 2D LiDAR [22, 32, 35] but much fewer works consider 3D LiDAR sensors [52, 45]. Nagatani et al. [52] propose to merge digital elevation maps obtained from three robots where inter-robot constraints are found on the basis of *submap* matching by assuming little drift and a known good estimate of the relative transformation between the robots. The map-merging strategy is performed offline and the experiment only consider a small environment. Michael et al. [45] present a strategy for generating a 3D map of a building damaged by an earthquake. The maps are locally built on each robot using a technique which assumes the environment to be composed of walls and horizontal ground planes. The two maps are merged afterwards, providing a good initial guess for the relative robot transformation which is then refined by ICP. Without place recognition, the two aforementioned solutions are not applicable to online multi-robot SLAM and cannot correct for drift which might occur in the single maps. Global place recognition techniques for 3D point clouds based on global descriptors [3, 88] and keypoint descriptors [66, 25] are presented but rarely integrated in a full online SLAM system, let alone a multi-robot one.

Incremental point cloud segmentation Closely related to our work, Whelan et al. [81] proposes an incremental region growing method for segmenting dense point cloud maps. Segmentation is done only once for each input cloud with a merging step afterwards. Only planar segments were

[11] https://github.com/openMVG/awesome_3DReconstruction_list

considered whereas our generic region growing algorithm allows for different tuples of growing policies. Tateno et al. [74] merge RGB-D data into a *global segmentation map* that is maintained by matching and propagating segments extracted from the current depth map. Similarly, Finman et al. [19] propose to segment the depth maps using an incremental variation of the graph-based Felzenszwalb algorithm. A voting algorithm is proposed for recomputing parts of the segmented map given new data. None of the above works proposed a solution for retrieving models based on the generated segments. Contrastingly, we show through multiple experiments that our incremental region growing algorithm can effectively be leveraged for localization.

Data-driven methods for 3D point clouds In recent years, CNNs have become the state-of-the-art method for generating learning-based descriptors, due to their ability to find complex patterns in data [36]. When working with 3D point clouds, methods based on CNNs achieve impressive performances in applications such as object detection [17, 44, 64, 39, 85, 82, 61], semantic segmentation [64, 39, 61, 75], and 3D object generation [83].

Recently, a handful of works proposing the use of CNNs for localization in 3D point clouds have started to appear [86, 13]. First, Zeng et al. [86] propose to extract data-driven 3D keypoint descriptors (3DMatch) which are robust to changes in point of view. Although impressive retrieval performances are demonstrated using an RGB–D sensor in indoor environments, it is not clear whether this method is applicable in real-time in large-scale outdoor environments. Elbaz et al. [13] propose to describe local subsets of points using a deep neural network autoencoder. The authors state that the implementation has not been optimized for real-time operation and no timings have been provided. Contrastingly, our work presents a data-driven segment-based localization method that can operate in real-time and that allows map reconstruction and semantic extraction capabilities.

To achieve this reconstruction capability, the architecture of our descriptor was inspired by autoencoders in which an encoder network compresses the input to a small dimensional representation, and a decoder network attempts to decompress the representation back into the original input. The compressed representation can be used as a descriptor for performing 3D object classification [5]. Brock et al. [5] also present successful results using variational autoencoders for reconstructing voxelized 3D data. Different configurations of encoding and decoding networks have also been proposed for reconstructing and completing 3D shapes and environments [26, 10, 76].

While autoencoders present an interesting opportunity of simultaneously accomplishing both compression and feature extraction tasks, optimal performance at both is not guaranteed. As we show in our work (See Annex 2.9, encoding and feature extraction can have conflicting goals when robustness to

changes in point of view is desired. In our work, we combine the advantages of the encoding-decoding architecture of autoencoders with a technique proposed by Parkhi et al. [56]. The authors address the face recognition problem by first training a CNN to classify people in a training set and afterwards use the second to last layer as a descriptor for new faces. This classification-based method is an alternative to training networks using contrastive loss [6] or triplet loss [79]. We use the resulting segment descriptors in the context of SLAM to achieve better performance, as well as significantly compressed maps that can easily be stored, shared, and reconstructed.

References

[1] Alexander Andreopoulos, Stephan Hasler, Heiko Wersing, Herbert Janssen, John K. Tsotsos, and Edgar Körner. Active 3D object localization using a humanoid robot. *IEEE Transactions on Robotics*, 27 (1):47–64, 2011. ISSN 15523098. doi: 10.1109/TRO.2010.2090058.

[2] Carlos A Arellano-Muro, Luis F Luque-Vega, Bernardino Castillo-Toledo, and Alexander G Loukianov. Backstepping control with sliding mode estimation for a hexacopter. In *Electrical Engineering, Computing Science and Automatic Control (CCE), 2013 10th International Conference on*, pages 31–36. IEEE, 2013.

[3] Michael Bosse and Robert Zlot. Place recognition using keypoint voting in large 3D lidar datasets. In *IEEE Int. Conf. on Robotics and Automation*, 2013.

[4] Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4393–4398. IEEE, 2004.

[5] Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. In *Workshop on 3D Deep Learning, NIPS*, 2016.

[6] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[7] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J.J. Leonard. Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age. *IEEE Trans. on Robotics*, 32(6):1309–1332, 2016.

[8] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Computer Vision – ECCV 2016: 14th European Conference on*, pages 628–644, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46484-8. doi: 10.1007/978-3-319-46484-8_38. URL `http://dx.doi.org/10.1007/978-3-319-46484-8_38`.

[9] Alejo Concha and Javier Civera. Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[10] Angela Dai, Charles Ruizhongtai Qi, and Matthias Niessner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *IEEE Conf. on Computer Vision and Pattern Recognition*, July 2017.

[11] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. Monoslam: Real-time single camera SLAM. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1052–1067, 2007.

[12] Renaud Dubé, Daniel Dugas, Elena Stumm, Juan Nieto, Roland Siegwart, and Cesar Cadena. SegMatch: Segment based place recognition in 3D point clouds. In *IEEE Int. Conf. on Robotics and Automation*, pages 5266–5272. IEEE, 2017.

[13] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *IEEE Conf. on Computer Vision and Pattern Recognition*, July 2017.

[14] D. Endo, Y. Okada, K. Nagatani, and K. Yoshida. Path following control for tracked vehicles based on slip-compensating odometry. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2871–2876, Oct 2007. doi: 10.1109/IROS.2007.4399228.

[15] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[16] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision (ECCV)*, September 2014.

[17] Martin Engelcke, Dushyant Rao, Dominic Zeng Wang, Chi Hay Tong, and Ingmar Posner. Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks. In *IEEE Int. Conf. on Robotics and Automation*, pages 1355–1361. IEEE, 2017.

[18] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *Int. Journal of Computer Vision*, 59(2):167–181, 2004.

[19] Ross Finman, Thomas Whelan, Michael Kaess, and John J Leonard. Efficient incremental map segmentation in dense rgb-d maps. In *IEEE Int. Conf. on Robotics and Automation*, pages 5488–5494. IEEE, 2014.

[20] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5431–5440, June 2016. doi: 10.1109/CVPR.2016.586.

[21] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[22] Dieter Fox, Jonathan Ko, Kurt Konolige, Benson Limketkai, Dirk Schulz, and Benjamin Stewart. Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339, 2006.

[23] Simon Fuhrmann, Fabian Langguth, and Michael Goesele. MVE – a multiview reconstruction environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, volume 6, No, 7, page 8, 2014.

[24] M. Gianni, M. A. R. Garcia, F. Ferri, and F. Pirri. Terrain contact modeling and classification for atvs. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 186–192, May 2016. doi: 10.1109/ICRA.2016.7487132.

[25] Karl Granström, Thomas B Schön, Juan I Nieto, and Fabio T Ramos. Learning to close loops from range data. *The Int. Journal of Robotics Research*, 30(14):1728–1754, 2011.

[26] Vitor Guizilini and Fabio Ramos. Learning to reconstruct 3d structures for occupancy mapping. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.

[27] W. Guo, Y. Mu, and X. Gao. Step-climbing ability research of a small scout wheel-track robot platform. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2097–2102, Dec 2015. doi: 10.1109/ROBIO.2015.7419083.

[28] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017. doi: 10.1109/ICCV.2017.322. URL `https://doi.org/10.1109/ICCV.2017.322`.

[29] Dirk Holz, Alexandru E Ichim, Federico Tombari, Radu B Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-D. *IEEE Robotics & Automation Magazine*, 22(4):110–124, 2015.

[30] Seunghoon Hong, Junhyuk Oh, Honglak Lee, and Bohyung Han. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3204–3212, 2016. doi: 10.1109/CVPR.2016.349. URL `https://doi.org/10.1109/CVPR.2016.349`.

[31] Hamidreza Houshiar and Andreas Nüchter. 3d point cloud compression using conventional image compression for efficient data transmission. In *Information, Communication and Automation Technologies (ICAT), 2015 XXV International Conference on*, pages 1–8. IEEE, 2015.

[32] Andrew Howard. Multi-robot simultaneous localization and mapping using particle filters. *The International Journal of Robotics Research*, 25(12):1243–1256, 2006.

[33] Dinesh Jayaraman and Kristen Grauman. Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In *Computer Vision – ECCV 2016: 14th European Conference on*, pages 489–505. Springer International Publishing, Cham, 2016. ISBN 978-3-319-46454-1. doi: 10.1007/978-3-319-46454-1_30. URL `http://dx.doi.org/10.1007/978-3-319-46454-1_30`.

[34] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[35] P. Koch, S. May, M. Schmidpeter, M. Kühn, C. Pfitzner, C. Merkl, R. Koch, M. Fees, J. Martin, and A. Nüchter. Multi-Robot Localization and Mapping based on Signed Distance Functions. *Journal of Intelligent and Robotic Systems*, 83(3):409–428, September 2016.

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] Vladimír Kubelka, Lorenz Oswald, François Pomerleau, Francis Colas, Tomáš Svoboda, and Michal Reinstein. Robust data fusion of multimodal sensory information for mobile robots. *Journal of Field Robotics*, 32(4):447–473, 2015. ISSN 1556-4967. doi: 10.1002/rob.21535. URL `http://dx.doi.org/10.1002/rob.21535`.

[38] Anh Tuan Le, D. C. Rye, and H. F. Durrant-Whyte. Estimation of track-soil interactions for autonomous tracked vehicles. In *Proceedings of International Conference on Robotics and Automation*, volume 2, pages 1388–1393 vol.2, Apr 1997. doi: 10.1109/ROBOT.1997.614331.

[39] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3D lidar using fully convolutional network. In *Robotics: Science and Systems*, 2016.

[40] Y. Li, S. Ge, H. Fang, C. Chu, and Y. Liu. Effects of the fiber releasing on step-climbing performance of the articulated tracks robots. In *2009*

IEEE International Conference on Robotics and Biomimetics (ROBIO), pages 818–823, Dec 2009. doi: 10.1109/ROBIO.2009.5420567.

[41] Jinguo Liu, Yuechao Wang, Shugen Ma, and Bin Li. Analysis of stairs-climbing ability for a tracked reconfigurable modular robot. In *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.*, pages 36–41, June 2005. doi: 10.1109/SSRR.2005.1501238.

[42] Teppo Luukkonen. Modelling and control of quadcopter. *Independent research project in applied mathematics, Espoo*, 2011.

[43] J. L. Martínez, A. Mandow, J. Morales, S. Pedraza, and A. García-Cerezo. Approximating kinematics for tracked mobile robots. *The International Journal of Robotics Research*, 24(10):867–878, 2005. doi: 10.1177/0278364905058239. URL `http://dx.doi.org/10.1177/0278364905058239`.

[44] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D convolutional neural network for real-time object recognition. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[45] Nathan Michael, Shaojie Shen, Kartik Mohta, Yash Mulgaonkar, Vijay Kumar, Keiji Nagatani, Yoshito Okada, Seiga Kiribayashi, Kazuki Otake, Kazuya Yoshida, et al. Collaborative mapping of an earthquake-damaged building via ground and aerial robots. *Journal of Field Robotics*, 29(5):832–841, 2012.

[46] Aron Monszpart, Nicolas Mellado, Gabriel J. Brostow, and Niloy J. Mitra. RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103:1–103:12, July 2015. ISSN 0730-0301. doi: 10.1145/2766995. URL `http://doi.acm.org/10.1145/2766995`.

[47] Carlos Moreno, Yilin Chen, and Ming Li. A dynamic compression technique for streaming kinect-based point cloud data. In *Computing, Networking and Communications (ICNC), 2017 International Conference on*, pages 550–555. IEEE, 2017.

[48] Raúl Mur-Artal and Juan D Tardós. Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM. *Proceedings of Robotics: Science and Systems, Rome, Italy*, 1, 2015.

[49] Raúl Mur-Artal, J. M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.

[50] K. Nagatani, N. Tokunaga, Y. Okada, and K. Yoshida. Continuous acquisition of three-dimensional environment information for tracked vehicles on uneven terrain. In *2008 IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 25–30, Oct 2008. doi: 10.1109/SSRR.2008.4745872.

[51] Keiji Nagatani, Hiroaki Kinoshita, Kazuya Yoshida, Kenjiro Tadakuma, and Eiji Koyanagi. Development of leg-track hybrid locomotion to traverse loose slopes and irregular terrain. *Journal of Field Robotics*, 28 (6):950–960, 2011. ISSN 1556-4967. doi: 10.1002/rob.20415. URL `http://dx.doi.org/10.1002/rob.20415`.

[52] Keiji Nagatani, Yoshito Okada, Naoki Tokunaga, Seiga Kiribayashi, Kazuya Yoshida, Kazunori Ohno, Eijiro Takeuchi, Satoshi Tadokoro, Hidehisa Akiyama, Itsuki Noda, et al. Multirobot exploration for search and rescue missions: A report on map building in robocuprescue 2009. *Journal of Field Robotics*, 28(3):373–387, 2011.

[53] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327. IEEE, 2011.

[54] D. T. Nguyen, B. S. Hua, M. K. Tran, Q. H. Pham, and S. K. Yeung. A field model for repairing 3d shapes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5676–5684, June 2016. doi: 10.1109/CVPR.2016.612.

[55] L. H. Pan, C. N. Kuo, C. Y. Huang, and J. J. Chou. The claw-wheel transformable hybrid robot with reliable stair climbing and high maneuverability. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 233–238, Aug 2016. doi: 10.1109/COASE.2016.7743386.

[56] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *BMVC*, volume 1, No. 3, page 6, 2015.

[57] M. Pecka, V. Šalanský, K. Zimmermann, and T. Svoboda. Autonomous flipper control with safety constraints. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2889–2894, Oct 2016. doi: 10.1109/IROS.2016.7759447.

[58] Jesse Pentzer, Sean Brennan, and Karl Reichard. Model-based prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation. *Journal of Field Robotics*, 31(3): 455–476, 2014. ISSN 1556-4967. doi: 10.1002/rob.21509. URL `http://dx.doi.org/10.1002/rob.21509`.

[59] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[60] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, June 2016. doi: 10.1109/CVPR.2016.609.

[61] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, July 2017.

[62] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788, 2016. doi: 10.1109/CVPR.2016.91. URL `https://doi.org/10.1109/CVPR.2016.91`.

[63] M. Reinstein and M. Hoffmann. Dead Reckoning in a Dynamic Quadruped Robot Based on Multimodal Proprioceptive Sensory Information. *IEEE Transactions on Robotics*, 29(2):563–571, April 2013. doi: 10.1109/TRO.2012.2228309.

[64] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[65] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2484–2493, June 2015. doi: 10.1109/CVPR.2015.7298863.

[66] Timo Röhling, Jennifer Mack, and Dirk Schulz. A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[67] Francesco Sabatino. Quadrotor control: modeling, nonlinearcontrol design, and simulation, 2015.

[68] Sajad Saeedi, Michael Trentini, Mae Seto, and Howard Li. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics*, 33(1):3–46, 2016.

[69] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11,

November 2012. ISSN 0730-0301. doi: 10.1145/2366145.2366199. URL http://doi.acm.org/10.1145/2366145.2366199.

[70] J. Simanek, M. Reinstein, and V. Kubelka. Evaluation of the ekf-based estimation architectures for data fusion in mobile robots. *IEEE/ASME Transactions on Mechatronics*, 20(2):985–990, April 2015. ISSN 1083-4435. doi: 10.1109/TMECH.2014.2311416.

[71] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition (Proc. DAGM)*, pages 11–20, Darmstadt, Germany, September 2010.

[72] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, Dec 2015. doi: 10.1109/ICCV.2015.114.

[73] Minhyuk Sung, Vladimir G. Kim, Roland Angst, and Leonidas Guibas. Data-driven structural priors for shape completion. *ACM Trans. Graph.*, 34(6):175:1–175:11, October 2015. ISSN 0730-0301. doi: 10.1145/2816795.2818094. URL http://doi.acm.org/10.1145/2816795.2818094.

[74] Keisuke Tateno, Federico Tombari, and Nassir Navab. Real-time and scalable incremental segmentation on dense slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 4465–4472. IEEE, 2015.

[75] Lyne P. Tchapmi, Christopher B. Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *International Conference on 3D Vision (3DV)*, 2017.

[76] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen. Shape completion enabled robotic grasping. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2442–2447, 2017.

[77] Holger Voos. Entwurf eines flugreglers für ein vierrotoriges unbemanntes fluggerätcontrol systems design for a quadrotor uav. *at-Automatisierungstechnik Methoden und Anwendungen der Steuerungs-, Regelungs-und Informationstechnik*, 57(9):423–431, 2009.

[78] Lujia Wang, Luyu Wang, Yinting Luo, and Ming Liu. Point-cloud compression using data independent method—a 3d discrete cosine transform approach. In *Information and Automation (ICIA), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[79] Kilian Q Weinberger, John Blitzer, and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in neural information processing systems*, pages 1473–1480, 2006.

[80] Martin Weinmann, Boris Jutzi, and Clément Mallet. Semantic 3D scene interpretation: a framework combining optimal neighborhood size selection with relevant features. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2(3):181, 2014.

[81] Thomas Whelan, Lingni Ma, Egor Bondarev, PHN de With, and John McDonald. Incremental and batch planar simplification of dense point cloud maps. *Robotics and Autonomous Systems*, 69:3–14, 2015.

[82] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 3109–3118, 2015.

[83] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016.

[84] Zhirong Wu, S. Song, A. Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, June 2015. doi: 10.1109/CVPR.2015.7298801.

[85] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D shapenets: A deep representation for volumetric shapes. In *IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015.

[86] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from rgb-d reconstructions. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[87] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S. C. Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3127–3134, June 2013. doi: 10.1109/CVPR.2013.402.

[88] Yan Zhuang, Nan Jiang, Huosheng Hu, and Fei Yan. 3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments. *IEEE Transactions on Instrumentation and Measurement*, 62(2):438–450, 2013.

[89] K. Zimmermann, P. Zuzánek, M. Reinstein, T. Petříček, and V. Hlaváč. Adaptive traversability of partially occluded obstacles. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3959–3964, May 2015. doi: 10.1109/ICRA.2015.7139752.

2 Annexes

2.1 Pecka-TIE2017, "Controlling Robot Morphology from Incomplete Measurements"

Bibliography Pecka, Martin and Zimmermann, Karel and Reinstein, Michal and Svoboda, Tomáš. "Controlling Robot Morphology from Incomplete Measurements" In *IEEE Transactions on Industrial Electronics*, Special issue on: on Motion Control for Novel Emerging Robotic Devices and Systems. Vol 64, No 2, 2017.

Abstract Mobile robots with complex morphology are essential for traversing rough terrains in Urban Search & Rescue missions (USAR). Since teleoperation of the complex morphology causes high cognitive load of the operator, the morphology is controlled autonomously. The autonomous control measures the robot state and surrounding terrain which is usually only partially observable, and thus the data are often incomplete. We marginalize the control over the missing measurements and evaluate an explicit safety condition. If the safety condition is violated, tactile terrain exploration by the body-mounted robotic arm gathers the missing data.

Relation to WP Describes an approach for automatic robot control on rough terrain. Contributes to the robot perception suite. T1.7.

Availablity Unrestricted. Included in the public version of this deliverable. `https://arxiv.org/abs/1612.02739`.

2.2 Pecka-IROS-2017, "Fast simulation of vehicles with non-deformable tracks"

Bibliography Martin Pecka, Karel Zimmermann, and Tomas Svoboda. "Fast simulation of vehicles with non-deformable tracks". In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*

Abstract This paper presents a novel technique that allows for both computationally fast and sufficiently plausible simulation of vehicles with non-deformable tracks. The method is based on an effect we have called Contact Surface Motion. A comparison with several other methods for simulation of tracked vehicle dynamics is presented with the aim to evaluate methods that are available off-the-shelf or with minimum effort in general-purpose robotics simulators. The proposed method is implemented as a plugin for the open-source physics-based simulator Gazebo using the Open Dynamics Engine.

Relation to WP Describes a novel model of a tracked robot for Gazebo simulator. Contributes to the robot perception suite. T1.7.

Availablity Unrestricted. Included in the public version of this deliverable. `https://arxiv.org/abs/1703.04316`

2.3 Zimmermann-ICCV-2017, "Learning for Active 3D Mapping"

Bibliography Karel Zimmermann, Tomáš Petříček, Vojtěch Šalanský, and Tomáš Svoboda. "Learning for Active 3D Mapping". In *Internation Conference on Computer Vision (ICCV2017)*

Abstract We propose an active 3D mapping method for depth sensors, which allow individual control of depth-measuring rays, such as the newly emerging solid-state lidars. The method simultaneously (i) learns to reconstruct a dense 3D occupancy map from sparse depth measurements, and (ii) optimizes the reactive control of depth-measuring rays. To make the first step towards the online control optimization, we propose a fast prioritized greedy algorithm, which needs to update its cost function in only a small fraction of possible rays. The approximation ratio of the greedy algorithm is derived. An experimental evaluation on the subset of the KITTI dataset demonstrates significant improvement in the 3D map accuracy when learning-to-reconstruct from sparse measurements is coupled with the optimization of depth measuring rays.

Relation to WP Describes a novel metod for creating local 3D maps from sparse 3D measurements. Contributes to the robot perception suite. T1.7.

Availability Unrestricted. Included in the public version of this deliverable. `https://arxiv.org/abs/1708.02074`

2.4 Azayev-MScThesis-2018, "Deep Learning for Autonomous Control of Robot's Flippers in Simulation"

Bibliography Teymur Azayev. Deep Learning for Autonomous Control of Robot's Flippers in Simulation. Master thesis, Czech Technical University, January 2018. Advisor: Karel Zimmermann.

Abstract Neural networks have seen increasing use in various robotic tasks such as locomotion largely due to advanced in Deep Learning techniques and Reinforcement Learning algorithms. We examine several Deep Learning approaches to learning a semi-autonomous locomotion policy for a ground based

search and rescue robot using only front facing RGBD camera and proprioceptive data. A supervised learning approach is suggested and implemented for the case where we only have a real robot and no simulated environment. We also suggest a method to deal with potential issues of multimodal action distributions using an alternative loss proxy based on Generative Adversarial Networks. Reactive as well as recurrent policies implemented using RNNs are compared. A simulator is used to train policies for the robot using Deep Reinforcement Learning. All policies are trained end-to-end, using convolutional neural networks for high dimensional image inputs. We examine the performance of policies trained with variously shaped rewards such as low control effort and smooth locomotion. Experiments are performed on the real robot using a learned RNN policy in the simulator and observe that the policy is transferable with no finetuning to the real environment, albeit, with some performance degradation. We also suggest two potential methods of domain transfer based on image modification using Gram matrix matching and Generative Adversarial Networks.

Relation to WP Describes a novel metod for controling robot flippers from 3D data directly. Contributes to the robot perception suite. T1.7.

Availablity Unrestricted. Included in the public version of this deliverable. `https://dspace.cvut.cz/handle/10467/74103`

2.5 Petricek-PhDThesis-2017, "Coupled Learning and Planning for Active 3D Mapping"

Bibliography Tomáš Petříček. Coupled Learning and Planning for Active 3D Mapping. PhD Thesis, Czech Technical University 2017. Advisors: Tomas Svoboda, Karel Zimmermann.

Abstract Autonomous robots, including those deployed in search and rescue operations or autonomous vehicles, must build and maintain accurate representations of the surroundings to operate efficiently and safely in human environment. These representations, or maps, should encompass both low-level information about geometry of the scene and high-level semantical information, including recognized categories or individual objects. In the first part we propose a method of 3D object recognition based on matching local invariant features, which is further extended for 3D point cloud registration task and evaluated on challenging real-world datasets. The method builds on a multi-stage feature extraction pipeline composed of sparse keypoint detection to reduce complexity of further stages, establishing local reference frames as a means to achieve invariance with respect to rigid transformations without sacrificing descriptiveness of the underlying 3D shape, and a compact description of the shape based on area-weighted normal projections.

For a moderate overlap between the laser scans, the registration method provides a superior registration accuracy compared to state-of-the-art methods including Gen- eralized ICP, 3D Normal-Distribution Transform, Fast Point-Feature Histograms, and 4-Points Congruent Sets. In the second part, two tasks from the area of active 3D mapping are being solved—namely, simultaneous exploration and segmentation with a mobile robot in a search and rescue scenario, and active 3D mapping using a sensor with steerable depth-measuring rays, with applications in autonomous driving. For these tasks, we assume that the localization is provided by an external source. In the simultaneous exploration and segmentation task, we consider a mobile robot exploring an unknown environment along a known path, using a static panoramic sensor providing RGB and depth measurements, and controlling a narrow field-of-view thermal camera mounted on a pan-tilt unit. The task is to control the sensor along the path to maximize accuracy of segmentation of the surroundings into human body and background categories. Since demanding optimal control does not allow for online replanning, we rather employ the optimal planner offline to provide guiding trajectories for learning a CNN-based control policy in a guided Q-learning framework. A policy initialization is proposed which takes advantage of a special structure of the task and allows efficient learning of the policy. In the active 3D mapping task, our method simultaneously learns to reconstruct a dense 3D occupancy map from sparse measurements and optimizes the reactive control of depth-measuring rays. We propose a fast prioritized greedy algorithm to solve the control subtask online, which needs to update the cost function in only a small fraction of possible rays in each iteration. An approximation ratio of the algorithm is derived. We experimentally demonstrate, using publicly available KITTI dataset, that accuracy of the 3D improves significantly when learning-to-reconstruct is coupled with the optimization of depth measuring rays.

Relation to WP Contributes to the robot perception suite. T1.7.

Availablity Unrestricted. Included in the public version of this deliverable.

`http://ptak.felk.cvut.cz/tradr/visuals/bagfile_crawler/Petricek-PhD-Thesis-2017.pdf`

2.6 Hollmannova-TR-CTU-2018, "RealSense and Elevation Mapping for Terrain Mapping on TRADR Robot"

Bibliography Dita Hollmannová. RealSense and Elevation Mapping for Terrain Mapping on TRADR Robot. Technical Report, CTU, January 2018. Advisor: Tomas Svoboda.

Abstract Experimental evaluation of an elevation mapping ROS package with the TRADR Lidar mapping suite. The main goal is to investigate the possibility of using RealSense sensor as the main depth measuring device for the TRADR adaptive traversal node. The RealSense is both less accurate and precise but delivers depth measure with much higher frame rates.

Relation to WP Contributes to the robot perception suite. T1.7.

Availablity Unrestricted. Included in the public version of this deliverable.

2.7 Dubé-IROS-2017, "An Online Multi-Robot SLAM System for 3D LiDARs"

Bibliography Renaud Dubé, , Abel Gawel, Hannes Sommer, Juan Nieto, Roland Siegwart, and Cesar Cadena. "An Online Multi-Robot SLAM System for 3D LiDARs". In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, (pp. 1004-1011).

Abstract Using multiple cooperative robots is advantageous for time critical search and rescue missions as they permit rapid exploration of the environment and provide higher redundancy than using a single robot. A considerable number of applications such as autonomous driving and disaster response could benefit from merging mapping data from several agents. Online multi-robot localization and mapping has mainly been addressed for robots equipped with cameras or 2D LiDARs. However, in unstructured and ill-lighted real-life scenarios, a mapping system can potentially benefit from a rich 3D geometric solution. In this work, we present an online localization and mapping system for multiple robots equipped with 3D LiDARs. This system is based on incremental sparse pose-graph optimization using sequential and place recognition constraints, the latter being identified using a 3D segment matching approach. The result is a unified representation of the world and relative robot trajectories. The complete system runs in real-time and is evaluated with two experiments in different environments: one urban and one disaster scenario. The system is available open source and easy-to-run demonstrations are publicly available.

Relation to WP This work contribute to Tasks T1.7 and T1.8 in proposing a real-time multi-robot SLAM system. We present an evaluation on data collected during a TRADR mission at the Knepper powerplant in Dortmund.

Availablity Unrestricted. Included in the public version of this deliverable[12].

2.8 Dubé-RAL-2018, "Incremental Segment-Based Localization in 3D Point Clouds"

Bibliography Renaud Dubé, , Mattia G. Gollub, Hannes Sommer, Igor Gilitschenski, Roland Siegwart, Cesar Cadena, and Juan Nieto. "Incremental Segment-Based Localization in 3D Point Clouds". Accepted for *IEEE Robotics and Automation Letters (RA-L)*, 2018.

Abstract Localization in 3D point clouds is a highly challenging task due to the complexity associated with extracting information from 3D data. This paper proposes an incremental approach addressing this problem efficiently. The presented method first accumulates the measurements in a dynamic voxel grid and selectively updates the point normals affected by the insertion. An incremental segmentation algorithm, based on region growing, tracks the evolution of single segments which enables an efficient recognition strategy using partitioning and caching of geometric consistencies. We show that the incremental method can perform global localization at 10Hz in a urban driving environment, a speedup of x7.1 over the compared batch solution. The efficiency of the method makes it suitable for applications where real–time localization is required and enables its usage on cheaper, low–energy systems. Our implementation is available open source along with instructions for running the system[13].

Relation to WP This work contribute to Task T1.8 by presenting an efficient approach to localization in 3D point clouds. This novel incremental solution addresses the real-time performance requirements of multi-robot and long-term missions.

Availablity Unrestricted. Included in the public version of this deliverable[14].

2.9 Dubé-2018, "*SegMap*: 3D Segment Mapping using Data-Driven Descriptors"

Bibliography Renaud Dubé, Andrei Cramariuc, Juan Nieto, Roland Siegwart, and Cesar Cadena. "*SegMap*: 3D Segment Mapping using Data-Driven

[12]The paper is available at `http://ieeexplore.ieee.org/abstract/document/8202268/`

[13]The implementation is available at `https://github.com/ethz-asl/segmatch` and a video demonstration is available at `https://www.youtube.com/watch?v=cHfs3HLzc2Y`.

[14]The paper is available at `http://www.gilitschenski.org/igor/publications/201801-ral-incremental_segmatch/ral18-incremental_segmatch.pdf`

Descriptors". Submitted to *Robotics: Science and Systems (RSS)*, 2018.

Abstract When performing localization and mapping, working at the level of structure can be advantageous in terms of robustness to environmental changes and differences in illumination. This paper presents *SegMap*: a *map representation* solution to the localization and mapping problem based on the extraction of segments in 3D point clouds. In addition to facilitating the computationally intensive task of processing 3D point clouds, working at the level of segments addresses the data compression requirements of real-time single- and multi-robot systems. While current methods extract descriptors for the single task of localization, *SegMap* leverages a data-driven descriptor in order to extract meaningful features that can also be used for reconstructing a dense 3D map of the environment and for extracting semantic information. This is particularly interesting for navigation tasks and for providing visual feedback to end-users such as robot operators, for example in search and rescue scenarios. These capabilities are demonstrated in multiple urban driving and search and rescue experiments. Our method leads to an increase of area under the ROC curve of 28.3% over current state of the art using eigenvalue-based features. We also obtain very similar reconstruction capabilities to a model specifically trained for this task. The *SegMap* implementation will be made available open-source along with easy to run demonstrations.

Relation to WP This work contributes to Task T1.7 by presenting a novel 3D *map representation* solution to SLAM for multi-robot systems. Thanks to its data-driven descriptor this approach addresses multiple search & rescue-related challenges: large-scale and long-term missions, real-time performances, limitted communication bandwidth, and providing end-users 3D visual feedback.

Availablity Restricted, submitted to RSS 2018. Included in the private version of this deliverable use consistent word for private with other deliverables.

2.10 Surmann-2017, "3D mapping for multi hybrid robot cooperation"

Bibliography Hartmut Surmann, Nils Berninger1, and Rainer Worst: 3D Segment Mapping using Data-Driven Descriptors". *IROS 2017*, 2017.

Abstract This paper presents a novel approach to build consistent 3D maps for multi robot cooperation in USAR environments. The sensor streams from unmanned aerial vehicles (UAVs) and ground robots (UGV) are fused

in one consistent map. The UAV camera data are used to generate 3D point clouds that are fused with the 3D point clouds generated by a rolling 2D laser scanner at the UGV. The registration method is based on the matching of corresponding planar segments that are extracted from the point clouds. Based on the registration, an approach for a globally optimized localization is presented. Apart from the structural information of the point clouds, it is important to mention that no further information is required for the localization. Two examples show the performance of the overall registration.

Relation to WP This work contributes to Task T1.8 by presenting a novel 3D solution to SLAM for hybrid multi-robot systems and different sensor streams from vision and Lidar.

Availablity Unrestricted. Included in the public version of this deliverable[15].

[15]The paper is available at `https://doi.org/10.1109/IROS.2017.8202217`

2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Fast Simulation of Vehicles with Non-deformable Tracks

Martin Pecka[1,2], Karel Zimmermann[2], and Tomáš Svoboda[1,2]

***Abstract*— This paper presents a novel technique that allows for both computationally fast and sufficiently plausible simulation of vehicles with non-deformable tracks. The method is based on an effect we have called *Contact Surface Motion*. A comparison with several other methods for simulation of tracked vehicle dynamics is presented with the aim to evaluate methods that are available off-the-shelf or with minimum effort in general-purpose robotics simulators. The proposed method is implemented as a plugin for the open-source physics-based simulator Gazebo using the Open Dynamics Engine.**

I. INTRODUCTION

Tracked vehicles are often preferred over the wheeled ones in tasks where traversing complicated terrain is needed, such as in Urban Search and Rescue missions. Tracks provide higher stability, better traction and help the vehicle traverse holes in the underlying terrain.

It is common in robotics research that the initial development of algorithms is first conducted in a simulator or game engine to avoid excessive wear of the real vehicle. In this phase, approximate simulation methods usually suffice, differing by the level of approximation and computation time. General-purpose simulators like Gazebo, V-REP, Webots, MORSE and Actin are often used for this task [1], providing various approximate motion models implemented in their physics engines (ODE, Bullet, Havoc).

Simulation of wheels is straightforward in these simulators, thus all of them provide means to simulate wheeled vehicles, including skid-steer motion of multi-wheel vehicles. However, there is no straightforward approach for tracked vehicle simulation, thus this motion model is not available in most simulators. After an exhaustive search, only two simulators were found providing a tracked robot in its robot model library – the commercial simulators Webots and V-REP. However, none of these implementations is both plausible on difficult terrain and computationally light.

The most plausible and general simulation methods for rubber belts are based on finite elements analysis, where the belt is subdivided in many small elements that interact in a defined way. We omit this class of methods in this work due to their inherent excessive computational complexity which makes them impractical for quick algorithm prototyping. Further argument for omitting these methods is that none of the most used open-source dynamics engines used in robotics supports simulation based on finite elements.

In this paper, we present a novel technique for non-deformable tracks simulation, which we implemented in the open-source simulator Gazebo [2]. The method provides a fast, simple and plausible simulation of non-deformable tracks with minimal changes to the simulator code and no changes to its physics engine (ODE). The method was successfully used in our prior work [3] for assessing safety of actions a tracked vehicle can perform. We would like to emphasize that our motivation is to have a fast and plausible method that can be easily integrated into existing robotics simulators and does not require implementation of state-of-the art physics engine components (which are usually absent in the robotics simulators).

We compare this method to other already known motion models. Finally, we propose a set of metrics that allow to compare the methods in terms of plausibility, computational time, and the range of track types that can be simulated by each of the respective techniques.

II. TYPES OF CATERPILLAR TRACKS

To clearly specify the type of vehicles this work is focused on, a short taxonomy of track types follows.

Based on the material the track is made of, the two basic types are *metal* tracks and *rubber* tracks. Metal tracks are usually made of many small *track plates* connected together with hinge-like joints. Rubber tracks are made of a continuous steel-reinforced band of rubber.

Another distinctive feature of different track types is the deformability of the outer shape of the track. The deformable track systems need a set of inner (sometimes also outer) wheels keeping the track approximately in the required shape and providing suspension (see Figure 1). The track can bend

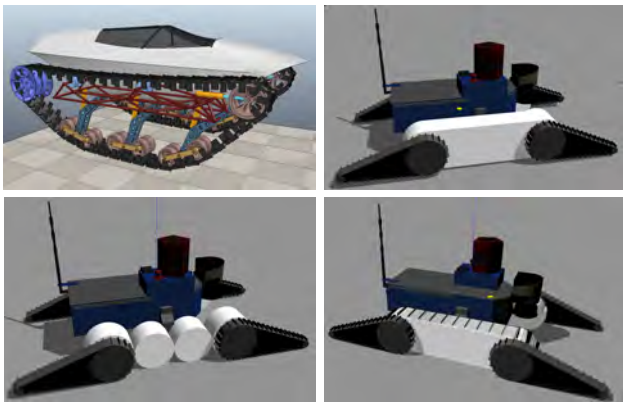


Fig. 1. **Track models.** ***Top left***: A vehicle with chain-like deformable tracks. This is the model available in model database of the V-REP simulator (courtesy of Qi Wang). ***Top right***: Non-deformable track model used for the proposed method. ***Bottom left***: Track approximated by 4 wheels. ***Bottom right***: Track made of 2 cm plates with grousers.

[1]Czech Technical University in Prague, Czech Institute of Informatics Robotics and Cybernetics
[2]Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics

in between the wheels, hence the name *deformable tracks*. Metal tracks are usually deformable, and also deformable rubber tracks exist.

Non-deformable tracks have solid guides (infills), which prevent the outer belt shape from bending and deformation (see Figure 1). This design is often chosen for rubber tracks, and it is the type this comparison is focused on.

A special category—*conveyor belts* and *escalators*—may be added to this taxonomy. In many design principles they are similar to the tracks for vehicles, but the main difference is they are always fixed to the environment and thus have no dynamics as a whole.

Independently from the above categories, tracks can be equipped with *grousers*. These protrusions enlarge the contact surface and help to increase traction in soft materials (depicted in Figure 1).

III. RELATED WORK

Depending on the purpose of the simulation, either very precise and detailed, or approximate models can be used. The former ones have been studied extensively in literature, whereas the approximate models, due to their triviality and inaccuracy, have not been examined profoundly despite their frequent use in nowadays robotics.

A. Precise Models

Simulation of the deformable tracks can be completely set up using existing robotics simulators – the track consists of a set of solid track plates connected with hinge joints, several wheels and, possibly, suspension of the wheels. All these components are available in simulators like Gazebo, Webots or V-REP. However, this type of simulations is both computationally intensive and very unstable for the high number of constrained dynamic elements [4]. Only the V-REP simulator provides a reliable simulation of this type, and many parameters have to be very finely tuned for it to work. Sokolov et al. [5] tried to implement this method in Gazebo, but the reported results are unsatisfactory.

When the general-purpose simulators fail, specialized simulators were developed to simulate the deformable track dynamics. Wallin et al. [6] compared several formulations of the mechanical joints when applied to metal tracks. They conclude that each formulation has its advantages and disadvantages and has to be chosen with respect to the specific use-case.

As discussed in the introduction, considerable effort is devoted to simulation of tracks using the Finite Elements Analysis [7], [8]. But the precision and computational demands are of higher orders than the methods we focus on.

In agriculture and military research, the track-soil interaction is of high interest (mainly due to sinkage of the track plates). Most of these works seem to only consider planar motion of the vehicle [9], [10], [11] and mainly concentrate on computing correct sinkage-induced behavior. Yamakawa and Watanabe [12] provide a fully three-dimensional simulation taking into account the track-soil interactions and wheel suspension.

B. Approximate Models

Common feature of the models described in the previous section is that they properly simulate some effects, but are either very computationally intensive, or neglect some other important effects (they e.g. assume motion on flat ground with small obstacles only).

We are not aware of any approximate model for the deformable track type, because its behavior is highly nonlinear and it essentially requires to model the individual parts of the track separately. The rest of this section thus concentrates on approximate models for non-deformable tracks.

In some environments, only flat ground is present (e.g. in household robotics or storehouse helper robots). Then there is effectively only a very small difference between a tracked robot and a 4-wheel robot with skid-steer control.

In some cases, the tracks can be treated completely passive and the robot motion can be roughly estimated by setting zero friction to the track surface, and pushing the robot with a virtual force instead of driving the tracks. This force can be applied via a P(ID) controller, so that the robot achieves the desired velocity and keeps it. However, the usual effects of friction can not be simulated. Consequently, the robot can not stand on a tilted plane without control force (which the real robot can do).

When negotiation of obstacles needs to be accounted for, the 4-wheel approximation would fail because the robot could not support itself on obstacle edges by the middle parts of the tracks. In this case, the problem is often solved by putting more virtual intersecting wheels inside the track. This approach has been tested by Sokolov et al. [13], and is available as a predefined model in V-REP and Webots simulators. The model still uses the skid-steer wheel control with synchronized wheel velocities on each side. On one hand, it has problems imitating the skid-steer behavior properly. On the other hand, the robot is able to overcome some obstacles and can support itself by any part of the track.But the geometry of such model does not correspond to the real geometry, which is why these models cannot plausibly simulate e.g. climbing up a staircase. We have observed in Section V that this model also gets stuck in some cases where the real robot would continue going. These models also do not work very well with the standard *friction pyramid* approximation of friction direction – it is instead needed to use the more precise (and more computationally expensive) *friction cone* model [1].

The V-REP simulator offers another method of approximate simulation, which is only suitable for conveyor belts and other static elements. It bypasses the physics by directly setting linear velocity of the whole conveyor belt mechanism, letting it interact with other bodies, and resetting all forces that acted on it afterwards. This way, the conveyor belt can exert forces on objects colliding with it, but at the same time, it stays on its place unaffected by any kind of dynamics (because the forces are zeroed-out each simulation step).

C. Skid-steer Motion

The slippage in the skid-steer behavior is an essential part of motion of tracked vehicles. While it automatically emerges from the precise simulation models as a result of track tension and other forces acting on the individual parts of the track, a kinematic model is also available for approximate or kinematics-only simulations.

Martínez et al. [14] define virtual points called *Instantaneous Centers of Rotation* (ICR) which depend on the desired turning radius and on a coefficient called *steering efficiency*. The robot follows a circular path centered at the ICR and if the steering efficiency is equal to 1, the motion is the same as the motion of a geometrically equal differential-drive wheeled vehicle.

Janarthanan et al. [15] extend this theory for tracked vehicles with road wheels.

IV. Model Based on Contact Surface Motion

Our novel method exploits the dynamic simulation formulation as *Linear Complementarity Problem* (LCP), which is used in ODE [16] and other robotics simulators. It does not, however, depend on any particular LCP solver implementation.

A. LCP Formulation

The dynamic simulation problem is an application of Newton's second law:

$$\mathbf{F} = Ma = \frac{d(M\dot{\mathbf{q}})}{dt} \tag{1}$$

where t is time, $\mathbf{F}$ is the force acting on the dynamic system, M is the mass and inertia matrix, and $\dot{\mathbf{q}}$ is the linear and angular velocity of the bodies (which is the derivative of the system state $\mathbf{q}$). The force $\mathbf{F}$ is split into *external force* $\mathbf{F_e}$ and *constraint force* $\mathbf{F_c}$ [4], which is a set of forces generated by joint constraints that keep joint constraints valid in the next time step.

The constraints are written in the form

$$\dot{C}(\mathbf{q}) = J\dot{\mathbf{q}} \geq 0 \tag{2}$$

where J is the *constraint Jacobian*. An observation in [4] states that the direction of the constraint force is given by J, so it is sufficient to search for the constraint force magnitude λ (so that $\mathbf{F_c} = J\ \lambda$).

In simulation, the derivative is discretized into short time steps Δt (usually 1 ms) and the state of the system is integrated step-by-step using Euler's integration [4]. The state of the system in the next time step $n+1$ can be expressed as

$$\mathbf{q_{n+1}} = \mathbf{q_n} + \mathbf{v_{n+1}}\Delta t$$

where the new velocity vector $\mathbf{v_{n+1}}$ (corresponding to $\dot{\mathbf{q}}$ in the continuous setting) is obtained from Equation 1:

$$\begin{aligned}\mathbf{v_{n+1}} &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + \mathbf{F_c})\Delta t \\ &= \mathbf{v_n} + M^{-1}(\mathbf{F_e} + J\lambda)\Delta t\end{aligned}$$

The unknown constraint force magnitude λ is the solution of the following LCP [4]:

$$\begin{aligned} JM^{-1}J^T\lambda\Delta t + J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ \text{given } \lambda \geq 0,\ J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t) &\geq 0 \\ (J(\mathbf{v_n} + M^{-1}\mathbf{F_e}\Delta t))^T\lambda &= 0 \end{aligned}$$

B. Contact Constraint Equations

In each time step, when links L_1 and L_2 collide, a set of contact points $\{C_i\}_{i=0}^{N}$ is generated at places where the links touch or penetrate each other. Every contact point is assigned a *contact joint*, which is a temporary constraint between L_1 and L_2. The set of constraints yielded by the contact joint consists of a position constraint (repelling the two links from each other along the contact normal), and a velocity constraint for friction (stopping parallel motion of the two links), which often utilizes the Coulomb friction representation [17], [18].

Linear velocity of L_1 is denoted by $\mathbf{v_1}$, angular velocity by $\omega_\mathbf{1}$, and $\mathbf{r_{1i}}$ is the vector from the center of L_1 to C_i; respective definitions hold for L_2. Further, $\mathbf{t_i}$ denotes the main tangential friction direction (which is perpendicular to the contact normal).

The approximate velocity constraint for Coulomb friction at contact point C_i with friction coefficient μ_i is [17]:

$$\frac{\partial C_i}{\partial t} = (\mathbf{v_2} + \omega_2 \times \mathbf{r_{2i}} - (\mathbf{v_1} + \omega_1 \times \mathbf{r_{1i}})) \cdot \mathbf{t_i} = 0 \tag{3}$$

$$-\mu_i \leq \lambda_i \leq \mu_i \tag{4}$$

which can be interpreted as "stop any motion in direction $\mathbf{t_i}$". The LCP solver tries to find magnitude of the friction force in direction $-\mathbf{t_i}$ (which is bounded by μ_i) that would satisfy this equation.

C. Contact Surface Motion Model

With the previous definitions, our novel method can be described as a modification of Equation 3. To account for the track velocity v_t, Equation 3 is adjusted to:

$$\frac{\partial C_i}{\partial t} = v_t$$

which might be interpreted as "find a force that would keep relative motion of L_1 and L_2 at velocity v_t". With this change, the model will move just by applying the modified friction constraints and setting v_t.

Nevertheless, this model is not able to correctly simulate grousers. If the real track has grousers, one way to add a similar effect to the simulation is to increase the friction coefficient. This method proved useful in our previous work [3] where we heavily utilized the simulator to find a control policy suitable also for the real robot.

There are more precise models for contacts with friction [18], but the practical experiments have shown that even the friction pyramid approximation used in ODE is sufficient for our method to work.

This method can be also easily used for tracks of various shapes. The only requirement is to be able to compute the normals of contact points on the tracks.

D. Enabling Skid-Steer Motion

The last part to be defined is the friction direction $\mathbf{t_i}$. If only forward motion is required, it can be simply set to be parallel to the tracks. However, this setting causes problems when the robot is to turn around using skid-steering motion (since the friction forces are not consistent with the turning maneuver).

Here we connect the dynamic simulation with the kinematic model of tracked vehicle motion by Martínez et al. introduced in Section III-C. The whole vehicle is said to be following a circular path centered at ICR (or driving straight if ICR is in infinity). Thus, we know the desired trajectory of all contact points on the track, and we set each direction $\mathbf{t_i}$ to be tangent to this trajectory, see Figure 2.

This model has been successfully used in our previous work [3]. Implementation of the proposed method (and some other) has been offered to the Gazebo community [2].

V. COMPARISON OF MODELS

In this section, a comparison of methods of modeling non-deformable tracks is presented.

A. Tested Models

The tested models are described in the following sections (and depicted in Figure 1). Each model is shortly introduced, and an abbreviation for it is defined, which is used throughout the rest of the text and figures. All the tested models differ only in representation of the main tracks – all other properties, such as mass, inertia, shape etc. were the same for all models.

With each of the models, identification of the most realistic set of parameters was done. The optimized parameters were always *linear* and *angular gain* – ratios that convert control inputs from simulator to velocity commands for the models. Other parameters were added only for the models they make sense with, and consist of *steering efficiency* and friction coefficients in the first and second friction direction.

First, we tried to manually find a suitable set of parameters and estimated the ranges for each of them. Then we did 5 iterations of optimization, in each of which we examined 5 samples from a multivariate Gaussian distribution centered on the so far best set of parameters (with covariance derived from the estimated ranges). Examination of each sample consisted of traversing all defined scenarios with model settings taken from the sample, and summing up the weighted metric values (defined further in this section). To account for the uncertainty in the simulator, each traversal was tried 3 times and the metric value was averaged over these trials.

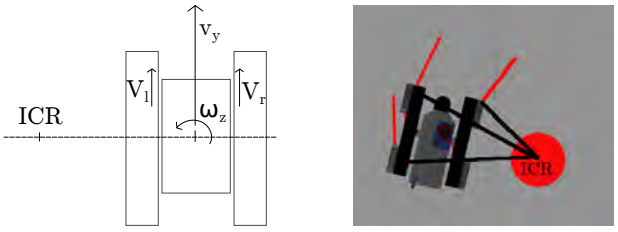


Fig. 2. **Instantaneous Center of Rotation.** ***Left***: A schematic view of the ICR. If the vehicle doesn't slip to the sides, ICR lies always on the depicted horizontal line passing through the centers of the tracks [14]. The distance of ICR from the center depends on forward velocity v_y and angular velocity ω_z (inverse kinematics), or the speeds of the left and right track V_l and V_r (forward kinematics). ***Right***: Computed directions of the friction forces $\mathbf{t_i}$ (red lines) for the case where ICR lies in the center of the red disk. The friction forces are perpendicular to the (black) lines connecting the contact points with ICR.

1) Model based on Contact Surface Motion: This is the novel model shortened as **CSM**.

2) Wheels instead of tracks: Model with 4 wheels instead of each track (**4wheels**) or 8 wheels (**8wheels**). All wheels are velocity-controlled using a skid-steer wheel control mechanism (with wheels on each side synchronized in velocity).

3) Subdivision to plates: Model with belt subdivided into 10 cm plates (**plates10**) or 2 cm plates (**plates2**) interconnected by hinge joints, plus sprocket and idler wheels. Versions with grousers attached to the track plates are shortened as **plates10g** and **plates2g**. The inner space of the track is filled with a solid box which can collide with the track plates, thus emulating the non-deformability of the track. Only the sprocket wheel is controlled, using torque control. This model requires more tuning in the simulator. To simplify it, the sprocket wheel is represented by a cylinder with infinite friction with the track plates (so that it efficiently transfers force to them without the need to model the teeth and their interaction with the plates). Further, lateral motion of track plates has to be avoided (otherwise, they would slip off the track very easily). This would be best done with a planar joint, which is however not available in Gazebo/ODE. As a workaround, placing two virtual vertical plates to the sides of each track (that collide only with the track plates) yields a similar behavior (although it is not ideal).

4) No friction: Model with zero friction between the tracks and ground (**no_friction**). The collision shape of the track is the same as in the **CSM** model, but the friction of the track is set to zero, and the whole model is force-controlled by applying a virtual force at its center of mass. The applied force is always perpendicular to the vertical axis of the robot.

5) The real robot: The **real** robot was also part of the test. It is the Absolem platform used in Urban Search and Rescue project TRADR [19]. Position of the robot in 6D space was measured by an IMU combined with track and laser odometry [20].

B. Test Scenarios

The models were tested in the following scenarios. Each scenario specifies a different metric showing how successful the model was, and was selected specifically to discover weak points of the models. All the scenarios start with the robot in rest, no initial speed, forces or torques. A view on the obstacles in the scenarios is provided in Figure 3.

CPU time was measured in all scenarios. It represents the (real-world) time difference between the start of first scenario execution, and the end of the last scenario execution (so it is summed up over all scenarios for each model). The

TABLE I

NUMERICAL COMPARISON OF THE SIMULATION METHODS.

	Metric	**csm** (proposed)	**4wheels**	**8wheels**	**no_friction**	**plates10**	**plates2**	**plates10g**	**plates2g**
Straight	d_t	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	$\mathbf{0.1 \pm 0.0}$	0.2 ± 0.1	1.6 ± 0.0	1.3 ± 0.0	1.6 ± 0.1	0.5 ± 0.0
Rotate	d_ω	$\mathbf{0.1 \pm 0.0}$	0.5 ± 0.0	1.6 ± 0.0	$\mathbf{0.1 \pm 0.1}$	1.4 ± 0.6	1.0 ± 0.1	2.5 ± 0.2	3.1 ± 0.0
Circular	$\sum d_t$	157.8 ± 5.7	$\mathbf{45.5 \pm 1.7}$	116.9 ± 6.4	47.4 ± 2.2	210.5 ± 30.6	189.5 ± 4.7	564.9 ± 36.3	195.7 ± 6.5
Back&forth	d_{st}	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	1.3 ± 0.1	0.1 ± 0.0	2.6 ± 0.2	0.3 ± 0.0
Ramp	$\sum d_\omega$	$\mathbf{1.8 \pm 0.0}$	2.2 ± 0.1	$\mathbf{1.8 \pm 0.0}$	2.0 ± 0.0	4.6 ± 1.1	2.6 ± 0.4	10.5 ± 4.1	34.7 ± 1.9
	$\sum d_t$	8.0 ± 2.7	$\mathbf{4.6 \pm 1.3}$	5.3 ± 0.8	16.8 ± 4.3	36.3 ± 1.6	61.7 ± 0.2	36.9 ± 3.5	121.9 ± 1.1
Staircase	$\sum d_\omega$	14.0 ± 0.8	10.2 ± 0.1	10.7 ± 0.3	17.1 ± 0.4	36.0 ± 31.3	$\mathbf{8.4 \pm 1.8}$	25.1 ± 11.1	45.5 ± 1.8
	$\sum d_t$	$\mathbf{12.1 \pm 1.0}$	16.3 ± 2.3	15.2 ± 0.9	13.0 ± 3.7	111.5 ± 5.9	86.8 ± 2.5	14.4 ± 7.0	163.0 ± 1.6
Stand on st.	d_{st}	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	0.2 ± 0.0	0.2 ± 0.0	0.6 ± 0.6	0.2 ± 0.0	0.2 ± 0.2	0.2 ± 0.0
	d_ω	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.1 ± 0.0	0.4 ± 0.4	0.1 ± 0.0	0.2 ± 0.2	0.1 ± 0.0
Pallet	$\sum d_\omega$	$\mathbf{27.6 \pm 0.9}$	27.7 ± 12.5	28.3 ± 17.7	31.8 ± 1.0	164.4 ± 151.5	47.6 ± 4.7	83.0 ± 13.1	64.3 ± 2.6
	$\sum d_t$	49.9 ± 2.5	116.9 ± 76.2	157.8 ± 46.6	$\mathbf{45.2 \pm 3.4}$	508.3 ± 146.3	181.1 ± 3.8	198.6 ± 72.7	78.4 ± 1.2
CPU time	time	38.9 ± 1.4	47.5 ± 1.7	82.4 ± 3.3	$\mathbf{33.0 \pm 1.3}$	254.9 ± 4.9	2282.6 ± 112.7	203.5 ± 8.3	2241.3 ± 31.8

Numerical results of the conducted experiments. Each model-scenario pair was executed 10 times, and the averages and standard deviations of the defined metrics are shown in the table. Shorthand d_t means the *distance to target point* metric (units are meters), d_{st} is distance from start. Term d_ω denotes the smallest angular offset from target roll-pitch-yaw orientation (units are radians). Terms $\sum d_t$ and $\sum d_\omega$ stand for the *sum of positional errors* or *sum of angular errors* respectively (with units meters and radians). *CPU time* (in last row) is not a scenario, but as it is aggregated over all scenarios for each model, we display it as a row of values. The duration of all scenarios in simulation time is 110 seconds, so a run-time of 30 seconds means the simulation ran at $\frac{11}{3}$ real-time speed on the test notebook. Best results in each scenario are highlighted in bold for better orientation.

simulators were running with high process priority without an upper bound on performance. The time complexity could be probably lowered for most of the models by adjusting the dynamics engine for the particular case; our measurements show CPU time needed by the implementation in the stock simulator without any code modifications.

Where a metric refers to the error from real robot trajectory, it means the scenario was traversed with the real robot, and the trajectory was recorded as a reference.

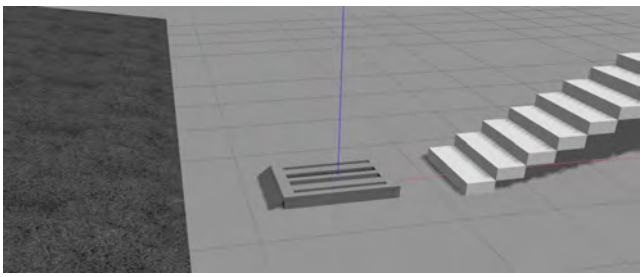


Fig. 3. **Obstacles used in test scenarios.** Obstacles that appear in the test scenarios (from the left): ramp, pallet, staircase. Also flat ground was used in scenarios. The models of the obstacles are 1:1 models of the obstacles traversed by the real robot.

1) Straight drive: Drive straight on a building floor using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: distance from point $(3.0, 0.0, 0.0)^T$.

2) Rotating in place: Keep the center at one place while rotating at $0.6\ rad.s^{-1}$ for 10 seconds. Metric: Angular distance from heading $6.0\ rad$, metric distance from the starting point.

3) Circular path: Follow a circular path by driving left track at velocity $0.1\ m.s^{-1}$ and right track at velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz.

TABLE II

SUMMARY RESULTS

	CSM	**Wheels**	**Plates**	**No friction**
Computation speed	✓	✓	×	✓
Plausibility on flat surfaces	✓	✓	✓	✓
Plausibility on rough terrain	✓	×	✓	×
Non-deformable tracks	✓	✓	✓	✓
Deformable tracks	×	×	✓	×
Grousers	×	×	✓	×

This table presents an overview based on the results of the conducted experiments. Sign "✓" means that the model is suitable for/supports the given use-case. Sign "×" means that the method is not suitable for/does not support the given use-case.

4) Ramp: Drive straight on a tilted ramp using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

5) Staircase: Climb down a staircase using velocity $0.3\ m.s^{-1}$ for 10 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

6) Stand on staircase: Stand on a staircase with no control commands for 10 seconds. Metric: Distance from the starting point, angular offset from the starting orientation.

7) Pallet: Climb over a pallet using velocity $0.1\ m.s^{-1}$ for 30 seconds. Metric: Sum of positional errors (from real robot trajectory) sampled at 10 Hz, sum of angular errors sampled at 10 Hz.

8) Back and forth: Drive using velocity $0.2\ m.s^{-1}$ back and forth 10 times, with 2 seconds between every direction switch. Metric: distance from the starting point.

C. Test results

Each model was tested 10 times in each scenario, and the values of the metrics were averaged over these tests.

The detailed results are shown in Table I. A summary extracted from the test results is given in Table II.

From the table, it follows that the track plate models are slower by an order of magnitude or two than the other models. We have also observed, that the 10 cm plates are too rough approximation of the smoothly curved belt, and the resulting model's motion could be described as "bumpy". Last observation for track plate models is that without grousers, the robot is often not able to climb up the pallet. That, however, corresponds to the expected real behavior of a belt without grousers.

The wheeled models are computationally fast and provide good plausibility in most scenarios. They suffer from unrealistic slippage in the *stand on staircase* scenario, because the friction forces have unrealistic directions. The *pallet* scenario showed to be a big problem for these methods—if a sharp edge (e.g. a step or pallet edge) touches the track in a point where neighboring wheels intersect, the model suddenly stops moving as a result of unrealistic forces and their directions. We think it is not a bug in our implementation, since the same behavior was also observed with the wheeled track model available in V-REP simulator (which even uses a different dynamics engine—Bullet).

The *no_friction* model provided good results in all tested scenarios, except *stand on staircase*. That failure is obviously caused by the missing friction between tracks and ground. It was the fastest tested model.

The proposed *Contact Surface Motion* model was the second fastest tested model. It provided good results in all tested scenarios except *circular path*. Here, the parameter optimization was not able to find a set of parameters that would provide good performance for both *rotate in place* and *circular path*; with the best set of parameters, the robot was turning too quickly in the *circular path* scenario. Together with *no_friction*, only these two models traversed the pallet without problems.

VI. CONCLUSION AND FUTURE WORK

Simulation of tracked vehicles is a complicated task even when it is narrowed down only to simulation of non-deformable tracks. The presented *Contact Surface Motion* model proved to be one of the fastest methods that still provide highly plausible results in most cases. It is the first computationally-light method allowing the use of precise geometry of the tracks while keeping plausible dynamic behavior. It can be utilized not only for simulation of tracked vehicles, but also for conveyor belts, treadmills and any other kind of moving planar surfaces.

The proposed set of metrics for comparison of the simulation models showed as a practical test for discovering weak and strong points of each model. Once the pull request to Gazebo [2] is merged, the testing world and obstacles [21] can be utilized by others to compare with their models.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union under grant agreement FP7-ICT-609763 TRADR; from the Czech Science Foundation under Project GA14-13876S, and by the Grant Agency of the CTU Prague under Project SGS16/161/OHK3/2T/13.

REFERENCES

[1] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending Open Dynamics Engine for Robotics Simulation," in *Lecture Notes Artificial Intell.*, vol. 6472 LNAI, 2010, pp. 38–50.

[2] Gazebo pull request "Added support for tracked vehicles". [Online]. Available: https://bitbucket.org/osrf/gazebo/pull-requests/2652

[3] M. Pecka, K. Zimmermann, and T. Svoboda, "Autonomous Flipper Control with Safety Constraints," in *IEEE Proc. Int. Robots Syst. (IROS)*. Los Alamitos, USA: IEEE, 2016, pp. 2889–2894.

[4] B. Kenwright and G. Morgan, "Practical Introduction to Rigid Body Linear Complementary Problem (LCP) Constraint Solvers," in *Alg. and Archit. Gaming Design*. IGI Global, 2012, pp. 159–201.

[5] M. Sokolov, I. Afanasyev, R. Lavrenov, A. Sagitov, L. Sabirova, and E. Magid, "Modelling a Crawler-type UGV for Urban Search and Rescue in Gazebo Environment," in *Int. Conf. Artificial Life Robotics (ICAROB)*, 2017.

[6] M. Wallin, A. K. Aboubakr, P. Jayakumar, M. D. Letherwood, D. J. Gorsich, A. Hamed, and A. A. Shabana, "A Comparative Study of Joint Formulations: Application to Multibody System Tracked Vehicles," *Nonlinear Dynamics*, vol. 74, no. 3, pp. 783–800, 2013.

[7] S. G. Arias, "Finite Element Analysis of Rubber Treads on Tracks to Simulate Wear Development," in *3DS Simulia Comm. Conf.*, 2012.

[8] Z. D. Ma and N. C. Perkins, "A Super-Element of Track-Wheel-Terrain Interaction for Dynamic Simulation of Tracked Vehicles," *Multibody System Dynamics*, vol. 15, no. 4, pp. 347–368, 2006.

[9] G. Ferretti and R. Girelli, "Modelling and Simulation of an Agricultural Tracked Vehicle," *J. Terramechanics*, vol. 36, no. 3, pp. 139–158, 1999.

[10] B. Janarthanan, C. Padmanabhan, and C. Sujatha, "Longitudinal Dynamics of a Tracked Vehicle: Simulation and Experiment," *J. Terramechanics*, vol. 49, no. 2, pp. 63–72, 2012.

[11] D. Rubinstein and R. Hitron, "A Detailed Multi-Body Model for Dynamic Simulation of Off-Road Tracked Vehicles," *J. Terramechanics*, vol. 41, no. 2-3, pp. 163–173, 2004.

[12] J. Yamakawa and K. Watanabe, "A Spatial Motion Analysis Model of Tracked Vehicles with Torsion Bar Type Suspension," *J. Terramechanics*, vol. 41, no. 2-3, pp. 113–126, 2004.

[13] M. Sokolov, R. Lavrenov, A. Gabdullin, I. Afanasyev, and E. Magid, "3D Modelling and Simulation of a Crawler Robot in ROS/Gazebo," in *Proc. Int. Conf. Control, Mechatron. and Autom.*, 2016, pp. 61–65.

[14] J. L. Martínez, A. Mandow, J. Morales, S. Pedraza, and A. García-Cerezo, "Approximating Kinematics for Tracked Mobile Robots," *Int. J. Robotics Research*, vol. 24, no. 10, pp. 867–878, 2005.

[15] B. Janarthanan, C. Padmanabhan, and C. Sujatha, "Lateral Dynamics of Single Unit Skid-Steered Tracked Vehicle," *Int. J. Automotive Technology*, vol. 12, no. 6, pp. 865–875, dec 2011.

[16] R. Smith *et al.*, "Open Dynamics Engine," 2005, http://www.ode.org/ode.html.

[17] J. Trinkle and J.-S. Pang, "On Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction," *J. Appl. Math. and Mechanics*, vol. 77, pp. 267–279, 1997.

[18] D. M. Kaufman, S. Sueda, D. L. James, and D. K. Pai, "Staggered Projections for Frictional Contact in Multibody Systems," *ACM Trans. Graph. (SIGGRAPH)*, vol. 27, no. 5, pp. 164:1–164:11, 2008.

[19] G.-J. M. Kruijff et al., "Designing, Developing, and Deploying Systems to Support Human-Robot Teams in Disaster Response," *Advanced Robotics*, vol. 28, no. 23, pp. 1547–1570, 2014.

[20] V. Kubelka and M. Reinstein, "Complementary Filtering Approach to Orientation Estimation Using Inertial Sensors Only," in *Proc. IEEE Int. Conf. Rob. Automat.*, 2012, pp. 599–605.

[21] Webpage containing all the data and configurations needed to replicate the experiments. [Online]. Available: http://cmp.felk.cvut.cz/~peckama2/simulation_quality/

Learning for Active 3D Mapping

Karel Zimmermann, Tomáš Petříček, Vojtěch Šalanský, and Tomáš Svoboda
Czech Technical University in Prague, Faculty of Electrical Engineering
zimmerk@fel.cvut.cz

Abstract

We propose an active 3D mapping method for depth sensors, which allow individual control of depth-measuring rays, such as the newly emerging solid-state lidars. The method simultaneously (i) learns to reconstruct a dense 3D occupancy map from sparse depth measurements, and (ii) optimizes the reactive control of depth-measuring rays. To make the first step towards the online control optimization, we propose a fast prioritized greedy algorithm, which needs to update its cost function in only a small fraction of possible rays. The approximation ratio of the greedy algorithm is derived. An experimental evaluation on the subset of the KITTI dataset demonstrates significant improvement in the 3D map accuracy when learning-to-reconstruct from sparse measurements is coupled with the optimization of depth measuring rays.

1. Introduction

Development of autonomous vehicles such as self-driving cars or ground robots has attracted substantial attention of the robotics community in the last few years. One of the reasons is that an accurate 3D perception, which is an essential component for many fundamental capabilities such as emergency braking, predictive active damping or self-localization from offline maps [12], has finally become possible. Since state-of-the-art rotating lidars are very expensive, heavy and contain moving parts prone to mechanical wear, several manufacturers have announced the development of cheaper, lighter, smaller and motionless solid-state lidars (SSL), which should become available soon [1].

In contrast to rotating lidars, the SSL uses an optical phased array as a transmitter of depth measuring light pulses. Since the built-in electronics can independently steer pulses of light by shifting its phase as it is projected through the array, the SSL can focus its attention on the parts of the scene important for the current task. Task-driven reactive control steering hundreds of thousands of rays per second using only an on-board computer is a challenging problem, which calls for highly efficient parallelizable algorithms. As a first step towards this goal, we propose an active mapping method for SSL-like sensors, which simultaneously (i) learns to *reconstruct a dense 3D voxel-map* from sparse depth measurements and (ii) optimize the reactive *control of depth-measuring rays*, see Figure 1. The proposed method is evaluated on a subset of the KITTI dataset [5], where sparse SSL measurements are artificially synthesized from captured lidar scans, and compared to a state-of-the-art 3D reconstruction approach [3].

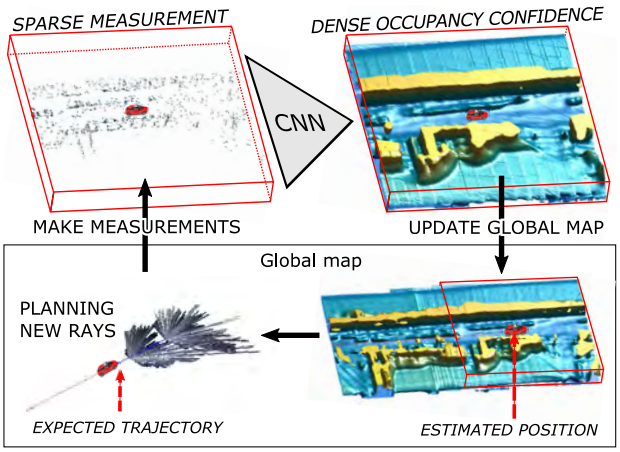


Figure 1. **Active 3D mapping with Solid State Lidar.** Iteratively learned deep convolutional network reconstructs local dense occupancy map from sparse depth measurements. The local map is registered to a global occupancy map, which in turn serves as an input for the optimization of depth-measuring rays along the expected vehicle trajectory. The dense occupancy maps are visualized as isosurfaces.

The main contribution of this paper lies in proposing a computationally tractable approach for very high-dimensional active perception task, which couples learning of the 3D reconstruction with the optimization of depth-measuring rays. Unlike other approaches such as active object detection [6] or segmentation [7], SSL-like reactive control has significantly higher dimensionality of the state-action space, which makes a direct application of unsupervised reinforcement learning [6] prohibitively expen-

sive. Keeping the on-board reactive control in mind, we propose prioritized greedy optimization of depth measuring rays, which in contrast to a naïve greedy algorithm re-evaluates only $1/500$ rays in each iteration. We derive the approximation ratio of the proposed algorithm.

The 3D mapping is handled by an iteratively learned convolution neural network (CNN), as CNNs proved their superior performance in [3, 17]. The iterative learning procedure stems from the fact that both (i) the directions in which the depth should be measured and (ii) the weights of the 3D reconstruction network are unknown. We initialize the learning procedure by selecting depth-measuring rays randomly to learn an initial 3D mapping network which estimates occupancy of each particular voxel. Then, using this network, depth-measuring rays along the expected vehicle trajectory can be planned based on the expected reconstruction (in)accuracy in each voxel. To reduce the training-planning discrepancy, the mapping network is re-learned on optimized sparse measurements and the whole process is iterated until validation error stops decreasing.

2. Previous work

High performance of image-based models is demonstrated in [14], where a CNN pooling results from multiple rendered views outperforms commonly used 3D shape descriptors in object recognition task. Qi *et al*. [10] compare several volumetric and multi-view network architectures and propose an anisotropic probing kernel to close the performance gap between the two approaches. Our network architecture uses a similar design principle.

Choy *et al*. [3] proposed a unified approach for single and multi-view 3D object reconstruction which employs a recurrent neural architecture. Despite providing competitive results in the object reconstruction domain, the architecture is not suitable for dealing with high-dimensional outputs due to its high memory requirements and would need significant modifications to train with full-resolution maps which we use. We provide a comparison of this method to ours in Sec. 6.2, in a limited setting.

Model-fitting methods such as [13, 15, 11] rely on a manually-annotated dataset of models and assume that objects can be decomposed into a predefined set of parts. Besides that these methods are suited mostly for man-made objects of rigid structure, fitting of the models and their parts to the input points is computationally very expensive; e.g., minutes per input for [13, 15]. Decomposition of the scene into plane primitives as in [8] does not scale well with scene size (quadratically due to candidate pairs) and could not most likely deal with the level of sparsity we encounter.

Geometrical and physical reasoning comprising stability of objects in the scene is used by Zheng *et al*. [18] to improve object segmentation and 3D volumetric recovery. Their assumption of objects being aligned with coordinate axes which seems unrealistic in practice. Moreover, it is not clear how to incorporate learned shape priors for complex real-world objects which were shown to be beneficial for many tasks (e.g., in [9]). Firman *et al*. [4] use a structured-output regression forest to complete unobserved geometry of tabletop-sized objects. A generative model proposed by Wu *et al*. [17], termed Deep Belief Network, learns joint probability distribution $p(\mathbf{x}, y)$ of complex 3D shapes $\mathbf{x}$ across various object categories y.

End-to-end learning of stochastic motion control policies for active object and scene categorization is proposed by Jayaraman and Grauman [6]. Their CNN policy successively proposes views to capture with RGB camera to minimize categorization error. The authors use a look-ahead error as an unsupervised regularizer on the classification objective. Andreopoulos *et al*. [2] solve the problem of an active search for an object in a 3D environment. While they minimize the classification error of a single yet apriori unknown voxel containing the searched object, we minimize the expected reconstruction error of all voxels. Also, their action space is significantly smaller than ours because they consider only local viewpoint changes at the next position while the SSL planning chooses from tens of thousands of rays over a longer horizon.

3. Overview of the active 3D mapping

We assume that the vehicle follows a known path consisting of L discrete positions and a depth measuring device (SSL) can capture at most K rays at each position. The set of rays to be captured at position l is denoted J_l.

We denote $\mathbf{Y}$ the global ground-truth occupancy map, $\hat{\mathbf{Y}}$ its estimate, and $\mathbf{X}$ the map of the sparse measurements. All these map share common global reference frame corresponding to the first position in the path. For each of these maps there are local counterparts $\mathbf{y}_l, \hat{\mathbf{y}}_l$, and $\mathbf{x}_l$, respectively. Local maps corresponding to position l all share a common reference frame which is aligned with the sensor and captures its local neighborhood. The global ground-truth map $\mathbf{Y}$ is used to synthesize sensor measurements $\mathbf{x}_l$ and to generate local ground-truth maps $\mathbf{y}_l$ for training.

The active mapping pipeline, consisting of a measure-reconstruct-plan loop, is depicted in Fig. 1 and detailed in Alg. 1. Neglecting sensor noise, the set of depth-measuring rays obtained from the planning, the measurements $\mathbf{x}_l$, and the resulting reconstruction $\hat{\mathbf{Y}}$ can all be seen as a deterministic function of mapping parameters θ and $\mathbf{Y}$. If we assume that that ground-truth maps $\mathbf{Y}$ come from a probability distribution, both learning of θ and planning of the depth-measuring rays approximately minimize common objective

$$\mathbb{E}_{\mathbf{Y}}\{\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}(\theta, \mathbf{Y}))\}, \tag{1}$$

where $\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = \sum_i w_i \log(1 + \exp(-Y_i \hat{Y}_i))$ is the weighted logistic loss, $Y_i \in \{-1, 1\}$ and $\hat{Y}_i \in \mathbb{R}$ denote

Algorithm 1 Active mapping

1: Initialize position $l \leftarrow 0$ and select depth-measuring rays randomly.
2: Measure depth in the directions selected for position l and update global sparse measurements $\mathbf{X}$ and dense reconstruction $\hat{\mathbf{Y}}$ with these measurements.
3: Obtain local measurements $\mathbf{x}_l$ by interpolating $\mathbf{X}$.
4: Compute local occupancy confidence $\hat{\mathbf{y}}_l = \mathbf{h}_\theta(\mathbf{x}_l)$ using the mapping network $\mathbf{h}_\theta$.
5: Update global occupancy confidence $\hat{\mathbf{Y}} \leftarrow \hat{\mathbf{Y}} + \hat{\mathbf{y}}_l$.
6: Plan depth-measuring rays along the expected vehicle trajectory over horizon L given reconstruction $\hat{\mathbf{Y}}$.
7: Repeat from line 2 for next position $l \leftarrow l + 1$.

the elements of $\mathbf{Y}$ and $\hat{\mathbf{Y}}$, respectively, corresponding to voxel i. In learning, $w_i \geq 0$ are used to balance the two classes, *empty* with $Y_i = -1$ and *occupied* with $Y_i = 1$, and to ignore the voxels with unknown occupancy. We assume independence of measurements and use, for corresponding voxels i, additive updates of the occupancy confidence $\hat{Y}_i \leftarrow \hat{Y}_i + h_i(\mathbf{x}_l)$ with $h_i(\mathbf{x}_l) \approx \log(\Pr(Y_i = 1|\mathbf{x}_l)/\Pr(Y_i = -1|\mathbf{x}_l))$. $\Pr(Y_i = 1|\mathbf{x}_l)$ denotes the conditional probability of voxel i being occupied given measurements $\mathbf{x}_l$ and $\sigma(\hat{Y}_i) = 1/(1 + e^{-\hat{Y}_i})$ is its current estimate.

4. Learning of 3D mapping network

The learning is defined as approximate minimization of Equation 1. Since (i) the result of planning $\mathbf{x}_l(\theta, \mathbf{Y})$ is not differentiable with respect to θ and (ii) we want to reduce variability of training data[1], we locally approximate the criterion around a point θ^0 as

$$\mathbb{E}_{\mathbf{Y}}\{\sum_l \mathcal{L}(\mathbf{y}_l, \mathbf{h}_\theta(\mathbf{x}_l(\theta^0, \mathbf{Y})))\} \quad (2)$$

by fixing the result of planning in $\mathbf{x}_l(\theta^0, \mathbf{Y})$. The learning then becomes the following iterative optimization

$$\theta^t = \arg\min_\theta \mathbb{E}_{\mathbf{Y}}\{\sum_l \mathcal{L}(\mathbf{y}_l, \mathbf{h}_\theta(\mathbf{x}_l(\theta^{t-1}, \mathbf{Y})))\}, \quad (3)$$

where minimization in particular iterations is tackled by Stochastic Gradient Descent. Learning is summarized in Alg. 2.

Note, that in order to achieve (i) local optimality of the criterion and (ii) statistical consistency of the learning process (i.e., that the training distribution of sparse measurements $\mathbf{x}_l$ corresponds to the one obtained by planning), one would have to find a fixed point of Equation 3. Since there are no guarantees that any fixed point exists, we instead iterate the minimization until validation error is decreasing.

The mapping network consists of 6 convolutional layers

[1] We introduce a canonical frame by using the local maps instead of the global ones, which helps the mapping network to capture local regularities.

Algorithm 2 Learning of active mapping

1: Initialize $t \leftarrow 0$ and obtain dataset $D_0 = \{(\mathbf{x}_l, \mathbf{y}_l)\}_l$ by running the pipeline with the rays being selected randomly, instead of using the planner.
2: Train the mapping network on D_t to obtain $\mathbf{h}_t$ with parameters θ^t.
3: Obtain $D_{t+1} = \{(\mathbf{x}_l(\theta^t, \mathbf{Y}), \mathbf{y}_l)\}_l$ by running Alg. 1 and using $\mathbf{h}_{\theta^t}$ for mapping.
4: Set $t \leftarrow t + 1$ and repeat from line 2 until validation error stops decreasing.

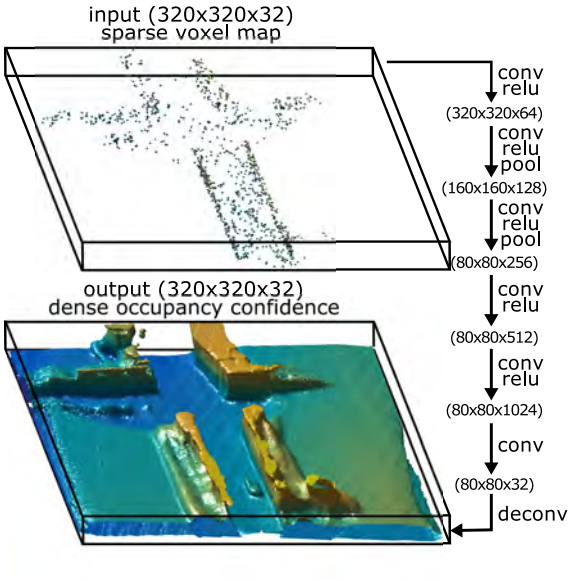


Figure 2. Architecture of the mapping network. **Top:** An example input with sparse measurements, showing only the occupied voxels. **Bottom:** The corresponding reconstructed dense occupancy confidence after thresholding. **Right:** Schema of the network architecture, composed from the convolutional layers (denoted *conv*), linear rectifier units (*relu*), and pooling (*pool*) and upsampling layers (*deconv*).

with 5×5 kernels followed by linear rectifier units (elementwise $\max\{x, 0\}$) and, in 2 cases, by max pooling layers with 2×2 kernels and stride 2, see Fig. 2. In the end, there is an fourfold upsampling layer so that the output has same size as input. The network was implemented in *MatConvNet* [16].

5. Planning of depth measuring rays

Planning at position l searches for a set of rays J, which approximately minimizes the expected logistic loss $\mathcal{L}(\mathbf{Y}, \mathbf{h}_{\theta^t}(\mathbf{x}_{l+L}))$ between ground truth map $\mathbf{Y}$ and reconstruction obtained from sparse measurements $\mathbf{x}_{l+L}$ at the horizon L. The result of planning is the set of rays J, which will provide measurements for a sparse set of vox-

els. This set of voxels is referred to as *covered* by J and denoted as $C(J)$. While the mapping network is trained *offline* on the ground-truth maps, the planning have to search the subset of rays *online* without any explicit knowledge of the ground-truth occupancy $\mathbf{Y}$. Since it is not clear how to directly quantify the impact of measuring a subset of voxels on the reconstruction $\mathbf{h}_{\theta^t}(\mathbf{x}_{l+L})$, we introduce simplified reconstruction model $\hat{\mathbf{h}}(J, \hat{\mathbf{Y}})$, which predicts the loss based on currently available map $\hat{\mathbf{Y}}$. This model conservatively assumes that the reconstruction in covered voxels $i \in C(J)$ is correct (i.e. $\mathcal{L}\big(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})\big) = 0$) and reconstruction of not covered voxels $i \notin C(J)$ does not change (i.e. $\mathcal{L}\big(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})\big) = \mathcal{L}(Y_i, \hat{Y}_i)$). Given this reconstruction model, the expected loss simplifies to:

$$\sum_i \mathcal{L}\big(Y_i, \hat{h}_i(J, \hat{\mathbf{Y}})\big) = \sum_{i \notin C(J)} \mathcal{L}(Y_i, \hat{Y}_i) \tag{4}$$

Since the ground-truth occupancy of voxels is apriori unknown, neither the voxel-wise loss nor the coverage are known. We model the expected loss in voxel i as

$$\mathcal{L}(Y_i, \hat{Y}_i) \approx \mathbb{E}_{Y_i \sim \mathrm{B}(\sigma(\hat{Y}_i))} \mathcal{L}(Y_i, \hat{Y}_i) = \mathcal{H}(\mathrm{B}(\sigma(\hat{Y}_i))) = \epsilon_i, \tag{5}$$

where $\mathcal{H}(\mathrm{B}(p))$ is the entropy of the Bernoulli distribution with parameter p, denoting the probability of outcome 1 from the possible outcomes $\{-1, 1\}$. The vector of concatenated losses ϵ_i is denoted $\boldsymbol{\epsilon}$.

The length of particular rays is also unknown, therefore coverage $C(J)$ of voxels by particular rays cannot be determined uniquely. Consequently, we introduce probability p_{ij} that voxel i will not be covered by ray $j \in J$. This probability is estimated from currently available map $\hat{\mathbf{Y}}$ as the product of (i) the probability that the voxels on ray j which lie between voxel i and the sensor are unoccupied and (ii) the probability that at least one of the following voxels or the voxel i itself are occupied. If ray j does not intersect voxel i, then $p_{ij} = 1$. The vector of probabilities p_{ij} for ray j is denoted $\mathbf{p}_j$. Assuming that rays J are independent measurements, the expected loss is modeled as $\boldsymbol{\epsilon}^\mathsf{T} \prod_{j \in J} \mathbf{p}_j$.

The planning searches for the set $J = J_1 \cup \cdots \cup J_L$ of subsets $J_1 \dots J_L$ of depth-measuring rays for the following L positions, which minimize the expected loss, subject to budget constraints $|J_1| \leq K, \dots |J_L| \leq K$

$$J^* = \arg\min_J \ \boldsymbol{\epsilon}^\mathsf{T} \prod_{j \in J} \mathbf{p}_j, \text{ s.t. } |J_1| \leq K, \dots |J_L| \leq K, \tag{6}$$

where $|J_l|$ denotes cardinality of the set J_l.

This is a non-convex combinatorial problem[2] which needs to be solved online repeatedly for millions of potential rays. We tried several convex approximations, however the high-dimensional optimization has been extremely time consuming and the improvement with respect to the significantly faster greedy algorithm was negligible. As a consequence of that, we have decided to use the greedy algorithm. We first introduce its simplified version (Alg. 3) and derive its properties, the significantly faster prioritized greedy algorithm (Alg. 4) is explained later.

[2] In our experiments, the number of possible combinations is greater then 10^{2000}.

We denote the list of available rays at position l as V_l. At the beginning, the list of all available rays is initialized as follows $V = V_1 \cup \cdots \cup V_L$. Alg. 3 successively builds the set of selected rays J. In each iteration the best ray j^* is selected, added into J and removed from V. The position from which the ray j^* is chosen is denoted l^*. If the budget K of l^* is reached, all rays from V_{l^*} are removed from V.

In order to avoid multiplication of all selected rays at each iteration, we introduce the vector $\mathbf{b}$, which keeps voxel loss. Vector $\mathbf{b}$ is initialized as $\mathbf{b} = \boldsymbol{\epsilon}$ and whenever ray j is selected, voxel losses are updated as follows $\mathbf{b} = \mathbf{b} \odot \mathbf{p}_j$, where $\odot$ denotes element-wise multiplication.

Algorithm 3 Greedy planning

Require: Set of available rays V and budget K

$J \leftarrow \emptyset$ ▷ Initialization
$\mathbf{b} \leftarrow \boldsymbol{\epsilon}$
while $\neg(V = \emptyset)$ **do**
  $j^* \leftarrow \arg\min_{j \in V} \mathbf{b}^\mathsf{T} \mathbf{p}_j$ ▷ Add the best ray
  $J \leftarrow J \cup j^*$
  $\mathbf{b} \leftarrow \mathbf{b} \odot \mathbf{p}_j$ ▷ Update voxel costs
  $V \leftarrow V \setminus j^*$ ▷ Remove j^* from V
  if $|J_{l^*}| = K$ **then**
    $V \leftarrow V \setminus V_{l^*}$ ▷ Close position
  end if
end while
return Set of selected rays J

The rest of this section is organized as follows: Section 5.1 shows the upper bound for the approximation ratio of the greedy algorithm. Section 5.2 introduces the prioritized greedy algorithm, which in each iteration needs to re-evaluate the cost function $\mathbf{b}^\mathsf{T} \mathbf{p}_j$ only for a small fraction of rays.

5.1. Approximation ratio of the greedy algorithm

We define the approximation ratio of a minimization algorithm to be $\rho = \frac{f}{\mathrm{OPT}}$, where f is the cost function achieved by the algorithm and OPT is the optimal value of the cost function. Given ρ, we know that the algorithm provides solution whose value is at most ρ OPT. In this section we derive the upper bound of the approximation ratio $\mathrm{UB}(\rho)$ of Algorithm 3. Figure 3 shows values of $\mathrm{UB}(\rho)$ for different number of positions L.

The greedy algorithm successively selects rays that reduce the cost function the most. To show how cost function differs from OPT, an upper bound on the cost function need to be derived. Let us suppose that in the beginning

of an arbitrary iteration we have voxel losses given by vector $\mathbf{b}$, the following lemma states that for arbitrary voxel i, there always exists a ray j, that reduces the cost function to $\sum_i b_i(1-\frac{1}{K}) + \frac{\text{OPT}}{K}$, where $\text{OPT} = \mathbf{1}^\top \prod_{j=1}^K \mathbf{p}_j = \mathbf{1}^\top \mathbf{p}^{\text{OPT}}$ is the unknown optimum value of the cost function which is achievable by K rays $\mathbf{p}_1 \dots \mathbf{p}_K$.

Lemma 5.1. *If for some rays* $\prod_{j=1}^K p_{ij} = p_i^{\text{OPT}}$ *then*

$$\forall_{\mathbf{0}\leq\mathbf{b}\leq\mathbf{1}} \exists_j \sum_{i=1}^V p_{ij} b_i \leq \sum_{i=1}^V b_i(1-\frac{1}{K}) + \frac{\text{OPT}}{K} \quad (7)$$

Proof: We know that there is optimal solution consisting from K rays. Without loss of generality we assume that $\prod_{j=1}^K p_{ij} = p_i^{\text{OPT}}$ holds for first K rays, then

$$\forall_i \sum_{j=1}^K p_{ij} \leq K - 1 + p_i^{\text{OPT}}. \quad (8)$$

This holds for an arbitrary positive scaling factor b_i, therefore

$$\forall_i \sum_{j=1}^K p_{ij} b_i \leq (K - 1 + p_i^{\text{OPT}}) b_i. \quad (9)$$

We sum up inequalities over all voxels i

$$\sum_{i=1}^V \sum_{j=1}^K p_{ij} b_i \leq \sum_{i=1}^V (K - 1 + p_i^{\text{OPT}}) b_i. \quad (10)$$

We switch sums in the left hand side of the inequality to obtain addition of K terms as follows

$$\sum_{i=1}^V p_{i1} b_i + \dots + \sum_{i=1}^V p_{iK} b_i \leq \sum_{i=1}^V (K - 1 + p_i^{\text{OPT}}) b_i \quad (11)$$

Hence, we know that at least one of these K terms has to be smaller than or equal to $\frac{1}{K}$ of the right hand side

$$\begin{aligned} \exists_j \sum_{i=1}^V p_{ij} b_i &\leq \frac{1}{K} \sum_{i=1}^V (K - 1 + p_i^{\text{OPT}}) b_i = \\ &= \sum_{i=1}^V b_i(1-\frac{1}{K}) + \frac{1}{K} \sum_{i=1}^V p_i^{\text{OPT}} b_i \leq \\ &\leq \sum_{i=1}^V b_i(1-\frac{1}{K}) + \sum_{i=1}^V \frac{p_i^{\text{OPT}}}{K} = \quad (12) \\ &= \sum_{i=1}^V b_i(1-\frac{1}{K}) + \frac{\text{OPT}}{K} \qquad \square \end{aligned}$$

Especially, if there is only one position, all optimal K rays $\mathbf{p}_1 \dots \mathbf{p}_K$ are either already selected or still available. This assumption allows to derive the following upper bound on the cost function of the greedy algorithm f^K after K iterations for $L = 1$.

Theorem 5.1. *Upper bound* $\text{UB}(f^K) \geq f^K$ *of the greedy algorithm after* K *iterations is*

$$\text{UB}(f^K) = E\frac{1}{e} + \text{OPT}\left(1 - \frac{1}{e}\right), \quad (13)$$

where $E = \sum_{i=1}^V \epsilon_i$ *and* e *is Euler number.*

Proof: We prove the upper bound by complete induction. In the beginning no ray is selected, per-voxel loss is $b_i^0 = \epsilon_i$ and the value of the cost function $f^0 = \sum_{i=1}^V b_i^0 = E$. Using Lemma 5.1, we know that there exists ray j such that $\sum_{i=1}^V p_{ij} b_i^0 \leq \sum_{i=1}^V b_i^0(1-\frac{1}{k}) + \frac{\text{OPT}}{K}$, therefore we know that

$$\begin{aligned} f^1 &= \sum_{i=1}^V p_{ij} b_i^0 \leq \sum_{i=1}^V b_i^0 \left(1 - \frac{1}{K}\right) + \frac{\text{OPT}}{K} = \\ &= E\left(1 - \frac{1}{K}\right) + \frac{\text{OPT}}{K}. \quad (14) \end{aligned}$$

Greedy algorithm continues by updating the per-voxel loss $b_i^1 = b_i^0 p_{ij}$. In the second iteration there are two possible cases: (i) we have either used the optimal ray in the first iteration, then the situation is better and we know there is $(K-1)$ rays which achieves optimum, or (ii) we have not selected the optimal ray in the first iteration, therefore we have still K rays which achieves the optimum. Since the cost function reduction in the latter case gives the upper bound on the cost function reduction in the former one, we assume that there is still k optimal rays available, therefore there exists ray j such that

$$\begin{aligned} f^2 &= \sum_{i=1}^V p_{ij} b_i^1 \leq \sum_{i=1}^V b_i^1 \left(1 - \frac{1}{k}\right) + \frac{\text{OPT}}{K} \leq \\ &\leq E\left(1 - \frac{1}{K}\right)^2 + \frac{\text{OPT}}{K}\left(\left(1 - \frac{1}{K}\right) + 1\right). \quad (15) \end{aligned}$$

We assume that the following holds

$$f^{t-1} \leq E\left(1 - \frac{1}{K}\right)^{t-1} + \frac{\text{OPT}}{K} \sum_{u=0}^{t-2} \left(1 - \frac{1}{K}\right)^u \quad (16)$$

and prove the inequality for f^t. Using the assumption (16) and Lemma 5.1, the following inequalities hold

$$\begin{aligned} f^t &\leq \sum_{i=1}^V b_i^{t-1} \left(1 - \frac{1}{K}\right) + \frac{\text{OPT}}{K} \leq \\ &\leq \left[E\left(1 - \frac{1}{K}\right)^{t-1} + \frac{\text{OPT}}{K} \sum_{u=0}^{t-2} \left(1 - \frac{1}{K}\right)^u\right] \left(1 - \frac{1}{K}\right) + \frac{\text{OPT}}{K} \\ &= E \underbrace{\left(1 - \frac{1}{K}\right)^t}_{\alpha_t^K} + \text{OPT} \underbrace{\frac{1}{K} \sum_{u=0}^{t-1} \left(1 - \frac{1}{K}\right)^u}_{\beta_t^K} \quad (17) \end{aligned}$$

Since $\alpha_t^K + \beta_t^K = 1$ [3] and $\alpha_K = \left(1 - \frac{1}{K}\right)^K \leq \frac{1}{e}$, the upper bound for cost function of the greedy algorithm in Kth iteration is $f^K \leq E\frac{1}{e} + \text{OPT}\left(1 - \frac{1}{e}\right)$ □

Theorem 5.1 reveals that the approximation ratio of the greedy algorithm $\rho = \frac{f^K}{\text{OPT}}$ after K iterations has following upper bound

$$\rho \leq \frac{\text{OPT}(\frac{E}{\text{OPT}}\frac{1}{e} + \left(1 - \frac{1}{e}\right))}{\text{OPT}} \leq \frac{E}{\text{LB}(\text{OPT})e} + \left(1 - \frac{1}{e}\right) \quad (18)$$

We can simply find $\text{LB}(\text{OPT})$ by considering for each voxel the best K rays independently.

So far we have assumed that the greedy algorithm chooses only K rays and that all rays are available in all iterations. Since there are L positions and the greedy algorithm can choose only K rays at each position, some rays may be no longer available when choosing $(K+1)$th ray. In the worst case possible, the rays from the most promising position will become unavailable. Since we have not chosen optimal rays we can no longer achieve OPT. Nevertheless, we can still choose from rays which achieve a new optimum.

We introduce $\overline{\text{OPT}}_v$ as the optimum achievable after closing v positions. Obviously $\overline{\text{OPT}}_0 = \text{OPT}$. Let us assume that, when the first position is closed we cannot lose more than R_1, therefore $\overline{\text{OPT}}_1 = \text{OPT} + R_1$. Without any additional assumption, R_1 could be arbitrarily large. We discuss potential assumptions later. Similarly $\overline{\text{OPT}}_2 = \text{OPT} + R_1 + R_2$, and $\overline{\text{OPT}}_v = \text{OPT} + \sum_{l=1}^{v} R_l$. The following theorem states the upper bound for f^{LK} as a function of $\overline{\text{OPT}}_v$.

Theorem 5.2. *Upper bound* $\text{UB}(f^{LK}) \geq f^{LK}$ *of the greedy algorithm after* LK *iterations is*

$$\text{UB}(f^{LK}) = E\frac{1}{e} + \sum_{u=0}^{L-1} \gamma_u \overline{\text{OPT}}_u, \quad (19)$$

where $\gamma_u = \left(1 - \sqrt[L]{\frac{1}{e}}\right)\left(\sqrt[L]{\frac{1}{e}}\right)^{L-1-u}$

Proof. We start from the result (17) shown in the proof of Theorem 5.1. Since there is LK rays achieving optimum $\overline{\text{OPT}}_0 = \text{OPT}$, the cost function f^K in Kth iteration is bounded as follows

$$f^K \leq E\underbrace{\left(1 - \frac{1}{LK}\right)^K}_{\alpha_K^{LK}} + \overline{\text{OPT}}_0 \underbrace{\frac{1}{LK}\sum_{u=0}^{K-1}\left(1 - \frac{1}{LK}\right)^u}_{\beta_K^{LK}} \quad (20)$$

In the $(K+1)$th iteration, there are two possible cases: (i) rays from some position l become not available and there is $K(L-1)$ rays available which can achieve a new optimum which is not higher than $\overline{\text{OPT}}_1$ or (ii) all rays are available and there is still LK rays which achieve $\overline{\text{OPT}}_0 = \text{OPT}$.

[3] $\beta_t^K = \frac{1}{K}\sum_{u=0}^{t-1}\left(1 - \frac{1}{K}\right)^u = (1-a)\sum_{u=0}^{t-1} a^u = 1 - a^t = 1 - \left(1 - \frac{1}{K}\right)^t = 1 - \alpha_t^K$ for $a = \left(1 - \frac{1}{K}\right)$.

Noticing that the upper bound is increasing in $\overline{\text{OPT}}_0$ and L, we can cover both cases by considering there is still LK rays which achieves $\overline{\text{OPT}}_1$, therefore

$$f^{K+1} \leq (E\alpha_K^{LK} + \overline{\text{OPT}}_0\beta_K^{LK})(1 - \frac{1}{LK}) + \frac{\overline{\text{OPT}}_1}{LK} = \\ = E\alpha_{K+1}^{LK} + \overline{\text{OPT}}_0\beta_K^{LK}(1 - \frac{1}{LK}) + \frac{\overline{\text{OPT}}_1}{LK} \quad (21)$$

We can now continue up to the iteration $2K$ in which the upper bound is as follows

$$f^{2K} \leq E\alpha_{2K}^{LK} + \overline{\text{OPT}}_0\beta_K^{LK}\alpha_K^{LK} + \overline{\text{OPT}}_1\beta_K^{LK} \quad (22)$$

For $(2K+1)$th iteration the situation is similar as for $(K+1)$th iteration. In order to cover both cases, we consider that there is LK rays which achieves $\overline{\text{OPT}}_2$ and continue up to the $3k$th iteration, which yields the following upper bound

$$f^{3K} \leq E\alpha_{3K}^{LK} + \overline{\text{OPT}}_0\beta_K^{LK}\alpha_{2K}^{LK} + \\ + \overline{\text{OPT}}_1\beta_K^{LK}\alpha_K^{LK} + \overline{\text{OPT}}_2\beta_K^{LK} \quad (23)$$

Finally after LK iterations the upper bound is

$$f^{LK} \leq E\alpha_{LK}^{LK} + \beta_K^{LK}\sum_{u=0}^{L-1}\alpha_{(L-1-u)K}^{LK}\overline{\text{OPT}}_u \leq \\ \leq E\frac{1}{e} + \sum_{u=0}^{L-1}\left(1 - \sqrt[L]{\frac{1}{e}}\right)\left(\sqrt[L]{\frac{1}{e}}\right)^{L-1-u}\overline{\text{OPT}}_u. \quad (24)$$

The last inequality stems from the fact that $(\alpha_K^{LK})^L = \alpha_{LK}^{LK} \leq \frac{1}{e}$ and that $\alpha_K^{LK} + \beta_K^{LK} = 1$. □

Finally we derive the upper bound of the approximation ratio $\rho = f^{LK}/\text{OPT}$.

Theorem 5.3. *Upper bound of the approximation ratio is*

$$\rho \geq \frac{E}{\text{LB}(\text{OPT})}\frac{1}{e} + \sum_{u=0}^{L-1}\gamma_u\left(1 + \frac{\sum_{v=1}^{u} R_v}{\text{LB}(\text{OPT})}\right) \quad (25)$$

where $\text{LB}(\text{OPT})$ *is lower bound of the* OPT.

Proof:

$$\rho = \frac{f^{LK}}{\text{OPT}} \leq \frac{\text{UB}(f^{LK})}{\text{OPT}} = \frac{E\frac{1}{e} + \sum_{u=1}^{L}\gamma_u\overline{\text{OPT}}_u}{\text{OPT}} = \\ = \frac{\text{OPT}(\frac{E}{\text{OPT}}\frac{1}{e} + \sum_{u=1}^{L}\gamma_u\frac{\overline{\text{OPT}}_u}{\text{OPT}})}{\text{OPT}} = \\ = \frac{E}{\text{OPT}}\frac{1}{e} + \sum_{u=1}^{L}\gamma_u\frac{\text{OPT} + \sum_{v=1}^{u}R_v}{\text{OPT}} \leq \quad (26) \\ \leq \frac{E}{\text{LB}(\text{OPT})}\frac{1}{e} + \sum_{u=0}^{L-1}\gamma_u\left(1 + \frac{\sum_{v=1}^{u}R_v}{\text{LB}(\text{OPT})}\right) \quad \square$$

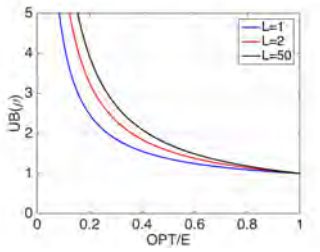


Figure 3. $\mathrm{UB}(\rho)$ as a function of $\frac{\mathrm{OPT}}{E}$ ratios with $R_v \leq \frac{V}{L}$.

The approximation ratio depends on the OPT, if $\mathrm{OPT} = 0$ then $\rho = \infty$, if $\mathrm{OPT} = E$ then $\rho = 1$. If we make an assumption that each position covers only $\frac{1}{L}$ fraction of voxels, then $R_v \leq \frac{V}{L}$. Figure 3 shows values of $\mathrm{LB}(\rho)$ for different ratios of $\frac{\mathrm{OPT}}{E}$ for this case.

5.2. Prioritized greedy planning

In practice we observed a significant speed up of the greedy planning (Alg. 3) by imposing prioritized search for $\arg\min_j \mathbf{b}^\mathsf{T}\mathbf{p}_j$. Namely, let us denote Δ_j^k the decrease of the expected reconstruction error achieved by selecting ray j in iteration k, $\Delta_j^k = \sum_i (b_i^{k-1} - b_i^k) = \sum_i b_i^{k-1}(1-p_{ij})$, and show that it is non-increasing. For $p_{ij}, p_{ij'} \in [0,1]$ and $b_i^{k-1} \geq 0$ it follows that $b_i^{k-1}(1-p_{ij}) \geq b_i^{k-1}p_{ij'}(1-p_{ij})$. Summing the inequalities for all voxels i, we get

$$\Delta_j^k = \sum_i b_i^{k-1}(1-p_{ij}) \geq \sum_i b_i^{k-1}p_{ij'}(1-p_{ij}) = \Delta_j^{k+1} \quad (27)$$

for an arbitrary ray j' selected in iteration k. Note that $\Delta_j^k \geq \Delta_j^{k+a}$ for any $a \geq 1$.

Now, when we search for j maximizing Δ_j^k in decreasing order of $\Delta_j^{k-a_j}$, $a_j \geq 1\ \forall j$, we can stop once $\Delta_j^k > \Delta_{j'}^{k-a_{j'}}$ for the next ray j' because none of the remaining rays can be better than j. Moreover, we can take advantage of the fact that all the remaining rays including j remained sorted when updating the priority for the next iteration. The proposed planning is detailed in Alg. 4.

The number of re-evaluations of Δ_j in Alg. 4 was approximately $500\times$ smaller than in Alg. 3. Despite the sorting took about a $1/10$ of the computation time, the prioritized planning was about $30\times$ faster and took 0.3s on average using a single-threaded implementation.

Algorithm 4 Prioritized greedy planning

Require: Set of rays $V = \{1, \ldots, N\}$ at positions L, budget K, voxel costs $\mathbf{b}$, probability vectors $\mathbf{p}_j\ \forall j \in V$, mapping from ray to position $\lambda\colon V \mapsto L$

$J_l \leftarrow \emptyset\ \forall l \in L$ ▷ No rays selected
$\Delta_j \leftarrow \infty\ \forall j \in V$ ▷ Force recompute
$S \leftarrow (1, \ldots, N)$ ▷ Sequence of ray indices, $S(n)$ denotes the nth element in the sequence, $S(m{:}n)$ the subsequence from the mth to the nth element.
while $S \neq \emptyset$ **do**
  for $n \in (1, \ldots, |S|)$ **do**
    $\Delta_{S(n)} \leftarrow \mathbf{b}^\mathsf{T}(\mathbf{1} - \mathbf{p}_{S(n)})$
    if $n < |S| \wedge \Delta_{S(n)} \geq \Delta_{S(n+1)}$ **then**
      break
    end if
  end for
  Sort subsequence $S(1:n)$ s.t. $\Delta_{S(n')} \geq \Delta_{S(n'+1)}$
  Merge sorted subsequences $S(1:n-1)$ and $S(n:|S|)$
  $j^* \leftarrow S(1), l^* \leftarrow \lambda(j^*)$
  $J_{l^*} \leftarrow J_{l^*} \cup \{j^*\}$ ▷ Add the best ray
  $\mathbf{b} \leftarrow \mathbf{b} \odot \mathbf{p}_{j^*}$ ▷ Update voxel costs
  if $|J_{l^*}| = K$ **then**
    $S \leftarrow S \setminus \{j : \lambda(j) = l^*\}$ ▷ Close position
  else
    $S \leftarrow S \setminus \{j^*\}$ ▷ Remove j^* from S
  end if
end while
return Selected rays J_l at every position $l \in L$

6. Experiments

Dataset All experiments were conducted on selected sequences from categories *City* and *Residential* from the KITTI dataset [5]. We first brought the point clouds (captured by the Velodyne HDL-64E laser scanner) to a common reference frame using the localization data from the inertial navigation system (OXTS RT 3003 GPS/IMU) and created the ground-truth voxel maps from these. The voxels traced from the sensor origin towards each measured point were updated as empty except for the voxels incident with any of the end points which were updated as occupied for each incident end point. The dynamic objects were mostly removed in the process since the voxels belonging to these objects were also many times updated as empty while moving. All maps used axis-aligned voxels of edge size $0.2\,\mathrm{m}$.

For generating the sparse measurements, we consider an SSL sensor with the field of view of $120°$ horizontally and $90°$ vertically discretized in $160 \times 120 = 19200$ directions. At each position, we select $K = 200$ rays and ray-trace in these directions until an occupied voxel is hit or the maximum distance of $48\mathrm{m}$ is reached. Only the rays which end up hitting an occupied voxel produce valid measurements, as is the case with the time-of-flight sensors. Local maps $\mathbf{x}_l$ and $\mathbf{y}_l$ contain volume of $64\mathrm{m} \times 64\mathrm{m} \times 6.4\mathrm{m}$ discretized into $320 \times 320 \times 32$ voxels.

6.1. Active 3D mapping

In this experiment, we used 17 and 3 sequences from the *Residential* category for training and validation, respectively, and 13 sequences from the *City* category for testing. We evaluate the iterative planning-learning procedure described in Sec. 4. For learning the mapping networks, we

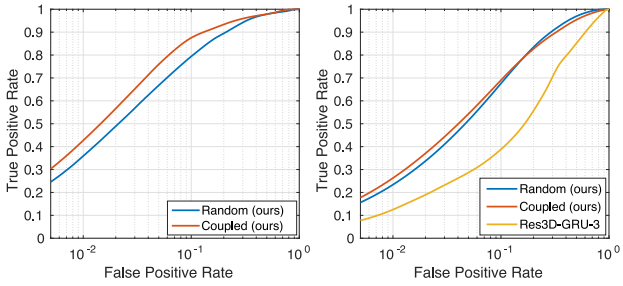


Figure 4. ROC curves of occupancy prediction from active 3D mapping on test sets. **Left:** *Random* denotes the global occupancy $\hat{\mathbf{Y}}$ obtained by using $\mathbf{h}_{\theta^0}$ with random sparse measurements, *Coupled* the occupancy obtained by using $\mathbf{h}_{\theta^3}$ with the prioritized greedy planning. The voxels which are more than 1m from what could possibly be measured are excluded, together with the false positives which can be attributed to discretization error (in 1-voxel distance from an occupied voxel). **Right:** *Random* denotes the local occupancy maps $\hat{\mathbf{y}}_l$ obtained by using $\mathbf{h}_{\theta^0}$, *Coupled* the maps obtained by using $\mathbf{h}_{\theta^1}$, and *Res3D-GRU-3* denotes the reconstruction obtained by the network adapted from [3].

used learning rate $\alpha = 10^{-3}(1/8)^{\lceil i/10 \rceil}$ based on epoch number i, batch size 1, and momentum 0.99. Networks $\mathbf{h}_{\theta^0}, \ldots, \mathbf{h}_{\theta^3}$ were trained for 20 epochs.

The ROC curves shown in Fig. 4 (left) are computed using ground-truth maps $\mathbf{Y}$ and predicted global occupancy maps $\hat{\mathbf{Y}}$. The performance of the $\mathbf{h}_{\theta^3}$ network (denoted *Coupled*) significantly outperforms the $\mathbf{h}_{\theta^3}$ network (*Random*), which shows the benefit of the proposed iterative planning-mapping procedure. Examples of reconstructed global occupancy maps are shown in Fig. 5. Note that the valid measurements covered around 3% of the input voxels.

6.2. Comparison to a recurrent image-based architecture

We provide a comparison with the image-based reconstruction method of Choy *et al*. [3]. Namely, we modify their residual *Res3D-GRU-3* network to use sparse depth maps of size 160×120 instead of RGB images. The sensor pose corresponding to the last received depth map was used for reconstruction. The number of views were fixed to 5, with $K = 200$ randomly selected depth-measuring rays in each image. For this experiment, we used 20 sequences from the *Residential* category—18 for training, 1 for validation and 1 for testing. Since the *Res3D-GRU-3* architecture is not suited for high-dimensional outputs due to its high memory requirements, we limit the batch size to 1 and the size of the maps to $128 \times 128 \times 32$, which corresponds to $16 \times 16 \times 4$ recurrent units. Our mapping network was trained and tested on voxel maps instead of depth images.

The corresponding ROC curves, computed from local maps $\mathbf{y}_l$ and $\hat{\mathbf{y}}_l$, are shown in Fig. 4 (right). Both $\mathbf{h}_{\theta^0}$ and $\mathbf{h}_{\theta^1}$ networks outperforms the *Res3D-GRU-3* network.

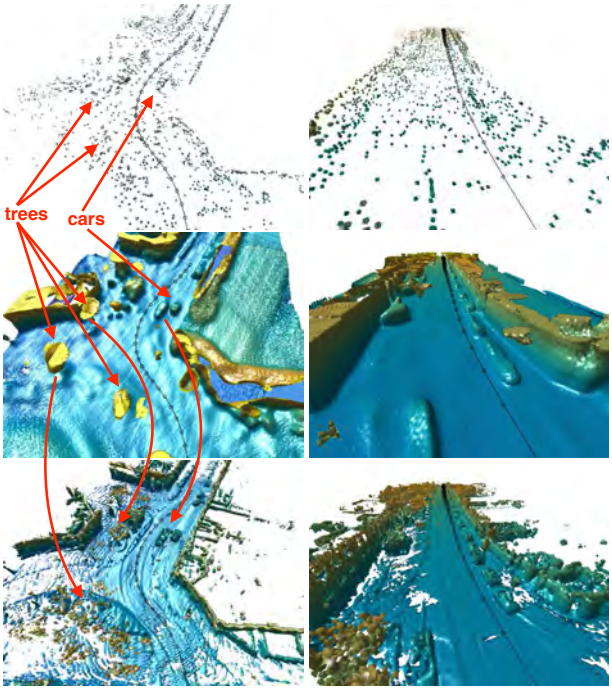


Figure 5. Examples of global map reconstruction. **Top:** Sparse measurement maps $\mathbf{X}$. **Middle:** Reconstructed occupancy maps $\hat{\mathbf{Y}}$ in form of isosurface. **Bottom:** Ground-truth maps $\mathbf{Y}$. The black line denotes trajectory of the car.

We attribute this result mostly to the fact that our method is implicitly provided the known trajectory, while the *Res3D-GRU-3* network is not. Another reason may be the ray-voxel mapping which is also known implicitly in our case, compared to [3].

7. Conclusions

We have proposed a computationally tractable approach for the very high-dimensional active perception task. The proposed 3D-reconstruction CNN outperforms a state-of-the-art approach by 20% in recall, and it is shown that when learning is coupled with planning, recall increases by additional 8% on the same false positive rate. The proposed prioritized greedy planning algorithm seems to be a promising direction with respect to on-board reactive control since it is about $30\times$ faster and requires only $1/500$ of ray evaluations compared to a naïve greedy solution.

Acknowledgment

The research leading to these results has received funding from the European Union under grant agreement FP7-ICT-609763 TRADR and No. 692455 Enable-S3, from the Czech Science Foundation under Project 17-08842S, and from the Grant Agency of the CTU in Prague under Project SGS16/161/OHK3/2T/13.

References

[1] E. Ackerman. Quanergy announces $250 solid-state LIDAR for cars, robots, and more. In *IEEE Spectrum*, January 2016. 1

[2] A. Andreopoulos, S. Hasler, H. Wersing, H. Janssen, J. K. Tsotsos, and E. Körner. Active 3D object localization using a humanoid robot. *IEEE Transactions on Robotics*, 27(1):47–64, 2011. 2

[3] C. B. Choy, D. Xu, J. Gwak, K. Chen, and S. Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Computer Vision – ECCV 2016: 14th European Conference on*, pages 628–644, Cham, 2016. Springer International Publishing. 1, 2, 8

[4] M. Firman, O. M. Aodha, S. Julier, and G. J. Brostow. Structured prediction of unobserved voxels from a single depth image. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5431–5440, June 2016. 2

[5] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, Sept. 2013. 1, 7

[6] D. Jayaraman and K. Grauman. Look-ahead before you leap: End-to-end active recognition by forecasting the effect of motion. In *Computer Vision – ECCV 2016: 14th European Conference on*, pages 489–505. Springer International Publishing, Cham, 2016. 1, 2

[7] A. K. Mishra, Y. Aloimonos, L. F. Cheong, and A. Kassim. Active visual segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):639–653, April 2012. 1

[8] A. Monszpart, N. Mellado, G. J. Brostow, and N. J. Mitra. RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM Trans. Graph.*, 34(4):103:1–103:12, July 2015. 2

[9] D. T. Nguyen, B. S. Hua, M. K. Tran, Q. H. Pham, and S. K. Yeung. A field model for repairing 3d shapes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5676–5684, June 2016. 2

[10] C. R. Qi, H. Su, M. Niener, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view CNNs for object classification on 3D data. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5648–5656, June 2016. 2

[11] J. Rock, T. Gupta, J. Thorsen, J. Gwak, D. Shin, and D. Hoiem. Completing 3d object shape from one depth image. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2484–2493, June 2015. 2

[12] H. Seifa and X. Hub. Autonomous driving in the iCity—HD maps as a key challenge of the automotive industry. *Autonomous Robots*, 2(2):159–162, 2016. 1

[13] C.-H. Shen, H. Fu, K. Chen, and S.-M. Hu. Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11, Nov. 2012. 2

[14] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 945–953, Dec 2015. 2

[15] M. Sung, V. G. Kim, R. Angst, and L. Guibas. Data-driven structural priors for shape completion. *ACM Trans. Graph.*, 34(6):175:1–175:11, Oct. 2015. 2

[16] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *Proceedings of the 23rd ACM International Conference on Multimedia*, MM '15, pages 689–692, New York, NY, USA, 2015. ACM. 3

[17] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, June 2015. 2

[18] B. Zheng, Y. Zhao, J. C. Yu, K. Ikeuchi, and S. C. Zhu. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3127–3134, June 2013. 2

3D mapping for multi hybrid robot cooperation

Hartmut Surmann[1] and Nils Berninger[1] and Rainer Worst[2]

Abstract— **This paper presents a novel approach to build consistent 3D maps for multi robot cooperation in USAR environments. The sensor streams from unmanned aerial vehicles (UAVs) and ground robots (UGV) are fused in one consistent map. The UAV camera data are used to generate 3D point clouds that are fused with the 3D point clouds generated by a rolling 2D laser scanner at the UGV. The registration method is based on the matching of corresponding planar segments that are extracted from the point clouds. Based on the registration, an approach for a globally optimized localization is presented. Apart from the structural information of the point clouds, it is important to mention that no further information is required for the localization. Two examples show the performance of the overall registration.**

(a) UAVs (b) UGV

Fig. 1. UAVs and UGV of the TRADR project after the earthquake in Amatrice / Italy 2016.

I. Introduction

Despite the growing technological advances, coping with disaster scenarios is still a major challenge for robots and humans. After 48 hours the probability of rescuing people from a collapsed building is drastically reduced [1]. The EU project TRADR develops novel science & technology for human-robot teams to assist in disaster response efforts, over multiple sorties during a mission. Various kinds of robots collaborate with human team members to explore the environment and to gather physical samples (fig. 1). The goal is to enable the team to gradually develop its understanding of the disaster area over multiple, possibly asynchronous sorties (persistent environment models), to improve team members' understanding of how to work in the area and to improve team-work. The fusion of different sensor streams of different semi-autonomous robots (UAV and UGV) in one consistent map is the basis for building the environment models. The UGV can be equipped with several sensors, e.g. tilting laser scanners and even actuators due to its higher payload. The UAV is equipped with only a few light sensors to reduce the weight and to increase the flight time. According to the current state of the UAV market, it is only possible to obtain important information in real-time by using a monocular camera. Furthermore, additional sensors such as GPS are available for localization, but a reliable position determination cannot always be ensured depending on the environment, e.g. close to buildings. The nature of the resulting data, which differ due to different sensors, is a major challenge. In order to use the data collaboratively, representations and algorithms have to be found that can process data from different sources (more or less in real-time).

For this work, a UGV with a laser scanner and a UAV with a monocular camera are used. While a laser scanner can directly provide distance information, more complex methods – generally known as Structure from Motion (SfM) – are used for extracting distance information from camera recordings. The difficulty when combining the data is finding corresponding regions that allow a robust transformation between the two point clouds. Naively, this should be possible with standard point-based scan matching. Unfortunately, point-based ICP fails due to the differences of the point clouds from the different sensors. Laser scanners compute precise radial point clouds whereas SfM or multi-view stereo algorithms compute less precise, erroneous, and non equally distributed dense point clouds, focused on brightness differences and textures. Therefore, geometrical structures have to be described as invariant as possible against the individual disturbances of the different sensors.

The objective of this work is the development of a method for a typical rescue scenario. The first responder arrives at the disaster site and uses the UAV to get an overview, i.e. images and an initial 3D point cloud. Humans and the UGVs use this initial sensor streams, which allow the localization of the UGV in a vision-based map. The approach uses surfaces that abstract from the underlying data structure and hence can compensate disturbances while still containing sufficient information for the motion estimation. The resulting 3D map combines the information from both sensors and thus has a higher information content. The collected data of the UAV have to be processed by a SfM method independent of the UGV. Subsequently, the results of the processing can be provided to the UGV for a first localization.

The remaining paper is organized as follows: The next section summarizes state of the art methods that can be

TRADR is funded by EU-FP7-ICT grant No. 609763. TRADR website: http://www.tradr-project.eu/.

[1]University of Applied Science Gelsenkirchen, Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven 53757 Sankt Augustin, Germany hartmut.surmann@w-hs.de

[2]Fraunhofer Institute for Intelligent Analysis and Information Systems IAIS, Schloss Birlinghoven 53757 Sankt Augustin, Germany rainer.worst@iais.fraunhofer.de

used to generate point clouds from camera recordings. Various registration methods are also reviewed and section III presents the selected registration method. An example of how we used this method for globally optimized localization is given in section IV. Several results of our experiments are shown in section V. Videos can be found at youtube www.youtube.com/watch?v=xAVR5aFv8VY.

II. Related work

A basic prerequisite for many tasks, such as navigation, mapping or cooperation of UAV and UGV, is the robot localization. When working with three-dimensional point clouds, the registration is significantly affected by the success of an exact localization [2]. Due to the aim of this work, to localize the UAV and UGV together in a global map, a registration method has to be found that can handle point clouds from different sources. In this context, it is important that the methods for registration as well as the generation of vision-based point clouds can be combined.

A. Vision-based SLAM

In order to perform visual odometry, only keypoints are selected, which make a robust correspondence search possible. While some methods compute complex features ([3], [4]), new developments increasingly use image points directly ([5]–[7]). Direct approaches have the advantage that they are not reduced to certain feature points but can exploit all image points to determine the odometry and depth values and thus provide more dense reconstructions of the environment. Depending on how many image points are utilized, the approaches can be divided into dense and semi-dense methods.

An example of a semi-dense approach is the SVO algorithm, which is presented in the work of [7]. The method uses point features, but these are not explicitly extracted. Rather they are an implicit result of a direct motion estimation. The initialization of the pose is achieved by minimizing the photometric error. LSD-SLAM [8] provides another direct approach. Based on the odometry method of [6], the algorithm generates globally consistent maps of the environment by means of graph optimization in large-area environments. Similar to the SVO algorithm, a probabilistic representation of the depth map is also used here to model inaccuracies. [9] also uses a probabilistic approach, but the method is based on a feature-based monocular SLAM system ([10]). Furthermore, in contrast to SVO and LSD-SLAM, the depth values of a reference image are not filtered over many individual images, but only key images are used for the reconstruction.

[11] presents one of the first real-time methods and provides dense reconstructions with a monocular camera. The tracking of the camera is based on the approach of [3]. The reconstruction is carried out using several key images. By expanding to several images, regions that would be hidden in two images or would be outside the corresponding image can also be reconstructed with a higher probability. DTAM ([5]) also provides dense reconstructions in real-time. In order to estimate the depth values, the method performs a global energy reduction over many individual images. REMODE ([12]) is a method for the reconstruction of dense point clouds, which integrates a Bayesian estimate into the optimization process. By modeling uncertainties of measurement for each pixel, regularization can be controlled precisely and inaccuracies in the localization can be reduced. Real-time capability is achieved through a CUDA-based implementation. For the pose estimation, the method of [7] is used. One of the recent developments of dense reconstructions is DPPTAM [13]. The approach reconstructs high textured regions with a semi-dense approach and low textured regions by approximation of surfaces. Thereby the assumption is made that homogeneously colored image regions form a plane, which can be determined by superpixels ([14]).

The procedures described so far fall under the category of online procedures, i.e. they are real-time capable and can deliver first results during camera recording. In contrast, offline procedures require all collected recordings in advance and then carry out the corresponding calculations. In [15], a pipeline for reconstruction is presented that combines all necessary processing steps in a software framework called MVE. The framework is also capable of reconstructing texturized surfaces.

B. Registration methods

Methods for registration can be divided roughly into point-based or iterative and feature-based methods ([2], [16]). An example of a known iterative method is the ICP-algorithm, which has already been implemented in several variants. According to [17] the transformation is determined by minimizing the Euclidean distance of the found point correspondences. The search for corresponding points and the calculation of the associated transformation for the alignment of these points is finally repeated iteratively until predefined limits have been reached. A disadvantage of iterative methods, however, is that they can converge to a local minimum under certain assumptions, such as an insufficient overlay of the scenes [2]. In addition, they can be sensitive to outliers and can be very computationally intensive with large amounts of data [18]. If several point clouds have to be registered, the generated scene must also be globally consistent. To achieve better results, it is common that feature-based methods are used for the initial registration and iterative methods are used for refining the already estimated transformation [2]. Features can be described by feature descriptors that incorporate geometric structures. If surfaces are used as a geometric structure, a high compression rate and thus a fast correspondence search can be achieved [16].

The work of [19] introduces a SLAM algorithm based on the registration of planar segments. The algorithm for the extraction of planar segments is based on the work of [20], which takes up the region-growing algorithm of [21] and adapts it by optimizations for the use in a SLAM system. For correspondence search and registration, the work of [22] is used. The presented MUMC-algorithm (Minimally Uncertain Maximum Consensus) maximizes geometric consistency

while minimizing the resulting uncertainties. As shown in the work of [19], both faster and more robust results can be obtained in comparison to an ICP-alorithm. [23] provides another plane-based registration method, which is based on the work of [19]. An approach that is also concerned with the registration of point clouds from different sensor groups is presented in [24]. As a first step, the method determines structural descriptors. For faster calculation, the descriptors are then projected into a subspace. A matching scheme is used to compare the descriptors and compute vote scores. The voting space is then used for place segmentation and for registration.

For this work, an algorithm is developed that is based on the approaches of [23] and [19]. The presented algorithm for surface extraction can be applied to unorganized point clouds and is fast in the calculation. The method of [19] has also proven itself in a test environment that is very close to a possible application area of this work.

III. Pose tracking

This section introduces the registration method used for relative localization. The first step is the segmention of planes from the source and the target point cloud as described in [23]. Afterwards corresponding planes and the associated transformation must be determined. The correspondence search is based on [23], but in contrast to the original algorithm, the area of the planes is determined by [25]. In addition, the correspondence search was extended by the examination of overlapping planes. This is done as follows: First, the transformation determined on the basis of corresponding planes is temporarily applied to the planes to be examined. Then the minimum and maximum coordinate values of each plane are determined and the vectors v_{min} and v_{max} are formed. Two planes ${}^{d}P$ and ${}^{m}P$ are overlapping when

$$ {}^{m}v_{min} < {}^{d}v_{max} + \varepsilon \quad \text{and} \quad {}^{d}v_{min} < {}^{m}v_{max} + \varepsilon \tag{1} $$

is satisfied. Here ε is a positive number that defines a tolerance range. The directions along the surface normals of the target planes can be ignored during verification.

The last step is to determine an optimal transformation from all corresponding planes as described in [19]. The rotation and translation is calculated in a separate step. A plane P will be defined by its oriented and normalized surface normal $\hat{n}$ and the distance d to the coordinate origin. If the correspondence set $\Omega = \{\langle {}^{m}P_{i1}, {}^{d}P_{i2}\rangle,\ i = 1, \ldots, N_\Omega\}$, which assigns every plane ${}^{d}P_{i2}$ of the source point cloud a corresponding plane ${}^{m}P_{i1}$ of the target point cloud, is known, the optimal rotation can be calculated by minimizing the following function:

$$ f(R) = \frac{1}{2}\sum_{i=1}^{N_\Omega} ||R\,{}^{d}\hat{n}_{i2} - {}^{m}\hat{n}_{i1}||^2. \tag{2} $$

The translation is expressed by the equation

$$ N_d^m t = d, \tag{3} $$

with

$$ N_{N_\Omega \times 3} = \begin{bmatrix} {}^{m}\hat{n}_1^T \\ \vdots \\ {}^{m}\hat{n}_{N_\Omega}^T \end{bmatrix} \quad \text{and} \quad d_{N_\Omega \times 1} = \begin{bmatrix} {}^{m}d_1 - {}^{d}d_1 \\ \vdots \\ {}^{m}d_{N_\Omega} - {}^{d}d_{N_\Omega} \end{bmatrix}. \tag{4} $$

The equation is solved by means of singular value decomposition. The singular value decomposition of the matrix N is given by

$$ N_{N_\Omega \times 3} = U_{N_\Omega \times N_\Omega} \Sigma_{N_\Omega \times 3} V_{3\times 3}^T. \tag{5} $$

Here the column vectors u_i of U are the left singular vectors and the column vectors v_i of V are the right singular vectors.

Σ is a $N_\Omega \times 3$ diagonal matrix, which contains the real positive singular values σ_i. Afterwards, a rank decision for the matrix N will be made, i.e. the rank r will be chosen so that $\sigma_r > 0$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$.

The best approximation of N is given by $\hat{N}_r$ with

$$ \hat{N}_r = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \tag{6} $$

The best translation estimation for rank r can finally be achieved by

$$ {}_{d}^{m}t = \sum_{i=1}^{r} \sigma_1^{-1} (u_i \cdot d)\, v_i. \tag{7} $$

If two laser point clouds are compared with one another, the ICP algorithm can be used for further refining the translation. The transformation already determined serves as an initial position estimation. Another option is given by the odometry estimation of the robot. If an additional translation estimation ${}_{d}^{m}t_e$ could be computed, it can be used to determine the missing translation directions v_i, $i = r+1, \ldots, 3$ and can be integrated with

$$ {}_{d}^{m}t = \sum_{i=1}^{r} \sigma_1^{-1} (u_i \cdot d)\, v_i + \sum_{i=r+1}^{3} \sigma_1^{-1} ({}_{d}^{m}t_e \cdot v_i)\, v_i \tag{8} $$

in the overall translation estimation.

IV. Localization

This section describes how the registration procedure described in section III can be used for localization and mapping. In robot localization, a distinction can be made between relative and absolute localization. In the case of relative localization, the changes in the respective current pose are determined from a known pose and thus the entire trajectory is built step by step. In the absolute localization, the pose is determined with respect to a given map. A disadvantage of the relative localization is that errors in the determination of the pose changes are accumulated and thus the estimated trajectory as well as the constructed map are not globally consistent. However, if the starting position

within a given map is known, the relative localization can be optimized. For each update step, the new pose is compared with the given map and a correction is made. The global map is provided by the UAV, which takes images during a first flight over the environment and generates a point cloud by means of a vision-based SLAM algorithm. If the absolute pose of the UGV is known in the global map, the map can be extended by the information of the laser scan and a more detailed map can be built step by step. This is useful, on the one hand, in low-textured regions, which cannot be covered by most camera-based methods. On the other hand, map areas such as interiors that are not accessible to the UAV or that are not visible in the event of a flyover due to occlusions can also be included in the global map. All processing steps involved are explained below; see fig. 2 for an overview of the whole process.

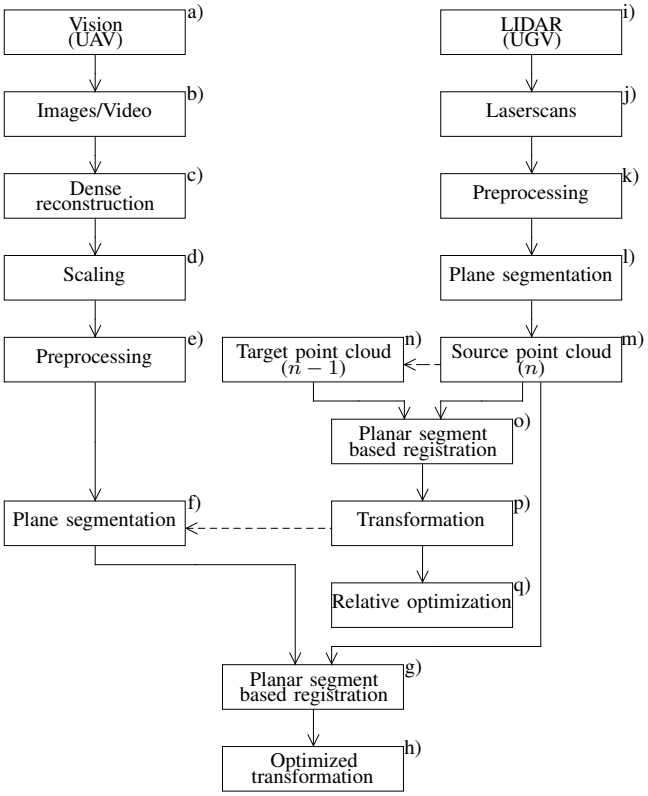


Fig. 2. Localization pipeline for laser point clouds with following global optimization step.

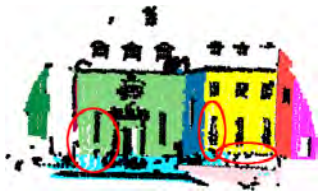
Steps a–c of fig. 2 describe the dense reconstruction with a suitable algorithm, e.g. MVE. After a reconstruction, the vision-based point clouds have to be scaled (fig. 2, d). This is necessary since the scaling factor for the reconstruction cannot be unambiguously determined when using a monocular camera. A correct scaling factor can be determined in several ways. For this work, the GPS coordinates recorded by the UAV during its flight are used. For the calculation, all positions estimated using the vision-based method are assigned to the nearest GPS coordinates and the Euclidean distances of adjacent points are calculated. The ratio of the average distances finally indicates the scaling factor.

As preparation for the plane segmentation, the point clouds are filtered through several processing steps (fig. 2, e and k). The aim of the preprocessing is to increase the robustness of the plane segmentation and thus the subsequent registration. By filtering, the point clouds are also reduced in size, which can considerably reduce the computational effort. For this work, a voxel grid filter and an outlier removal filter are applied, but additional filters can be added if necessary.

The relative pose of the UGV is updated with each new laser scan (fig. 2, l–p). First of all, a plane segmentation is carried out once for each new point cloud. Then the relative transformation between the last and the last but one point cloud is determined by means of a planar segment-based registration. For the initial laser point cloud, the assumption is made that it has approximately the correct pose with respect to the global map of the UAV. One way to determine the pose is by matching GPS coordinates. An exact pose is not necessary, since the initial pose is subsequently adjusted as part of the global optimization. If the initial pose was determined, an accumulation of the relative transformations can be used to estimate the current global pose. This pose will be optimized by aligning the associated point cloud with the global point cloud of the UAV. To achieve this, a planar segment-based registration is performed between the current laser point cloud and a section of the global vision-based point cloud (fig. 2, f–h). The position and size of the section is determined by the position and size of the current laser point cloud. Since the relative transformation is not always exact, the section is additionally expanded by a tolerance range. If the UGV moves in regions that are not or only slightly captured in the global point cloud of the UAV, a global optimization is not possible. In this case, a relative optimization can be carried out using a metascan algorithm (fig. 2, q). For this purpose the *simultaneous matching* algorithm from [26] was adapted for a surface-based approach and used for this work as follows:

1) The first point cloud that could no longer be optimized globally is defined as the master point cloud and determines the coordinate system. The already calculated relative transformation of a new point cloud serves as the initial registration of the relative optimization.
2) A list is initialized with the new point cloud.
3) The following three steps are repeated until the list contains no more elements:
    a) The first point cloud in the list is removed as the current point cloud from the list.
    b) If the current point cloud is not the master point cloud, then the neighbouring point clouds of the current point cloud are calculated. A point cloud is regarded as a neighbouring point cloud when a given minimum number of surfaces overlap with the surfaces of the current point cloud. All neighbouring point clouds are then grouped into a single point cloud and a planar segment-based registration with the current point cloud is

(a) Point cloud 1 (b) Point cloud 2 (c) Point cloud 3



(d) Point clouds 1 and 2

Fig. 3. Intermediate result of the relative optimization with three laser point clouds recorded on the site of Fraunhofer IAIS. The segmented surfaces are marked in color. The red dot indicates the respective position of the UGV. (c) shows the current point cloud, which was initially registered with point cloud (b). Point clouds 1 and 2 represent neighbouring point clouds of point cloud 3 and are summarized in (d). Point cloud 2 has gaps due to occlusions (see red markings). By a combination with point cloud 1, however, the gaps could be closed and thus a better calculation of the area size with respect to point cloud 3 could be carried out (see red markings in (d)).

performed.

c) If the calculated transformation changes the pose of the current point cloud by more than a pre-defined minimum, all neighbouring point clouds that are not already in the list are added to the list.

Figure 3 illustrates a possible result of the relative optimization.

So far, the assumption has been made that the pose of the first laser point cloud is approximately known with respect to the global point cloud. In the following, an approach to determine the initial pose is presented, which only uses the structural information from the point clouds. The procedure is orientated on [27] and can be described as follows:

1) The planes of the initial laser point clouds are segmented.
2) The global point cloud is divided into cells. The size of a cell is determined by the size of the laser point cloud plus a tolerance range. For each cell, a plane segmentation as well as a planar segment-based registration with the laser point cloud is done.
3) For each registration with a cell, the proportion of the match is calculated as follows:

$$r = \frac{|\mathcal{K}|}{max\left(|^m P|, |^d P|\right)}, \tag{9}$$

where $|\mathcal{K}|$ is the number of matching planes. The respective number of segmented planes of the laser point cloud and the point cloud set by the cell is given by $|^m P|$ and $|^d P|$. The better the current cell represents the position of the laser point cloud, the greater is the number of corresponding planes. The number $|\mathcal{K}|$ of the corresponding planes therefore corresponds to a large proportion of the maximum possible number of correspondences. For cells with few common planes, the proportion of correspondences is small compared to the possible number of correspondences. The cell that best represents the position of the laser point cloud is given by the largest value r. If r_1 is the largest determined value, the following criteria must be met for a unique match:

$$r_1 > \alpha \quad \text{and} \quad r_1 > \beta r_2. \tag{10}$$

α is a pre-defined threshold that r_1 must reach at least. r_2 is the second largest value given by equation 9. The ratio β defines the relationship between r_1 and r_2. If these criteria cannot be fulfilled, there are several cells with a similar agreement and the best position for the laser point cloud is not clearly determinable. In this case, further laser point clouds have to be collected and re-evaluated. The global position of the laser point cloud can finally be determined by the combination of the cell position and the transformation, which was calculated in the context of the planar segment-based registration.

TABLE I

PLANE SEGMENTATION OF THE LASER POINT CLOUD.

Nr.	Points	Preprocessing [s]	Segmentation [s]	Planes
1	103090	0.0090	0.1989	8
2	135233	0.0096	0.8091	9
3	182623	0.0171	1.2556	11
4	265629	0.0175	1.4302	14
5	286333	0.0248	2.5137	19
6	291043	0.0197	3.0201	29
7	289934	0.0188	2.8454	24

TABLE II

REGISTRATION OF THE LASER POINT CLOUD.

Pair	Registration [s]	Correspondences	ICP [s]
$1 \rightarrow 2$	0.0116	2	0.817343
$2 \rightarrow 3$	0.3643	8	0.231159
$3 \rightarrow 4$	0.0814	7	0.725623
$4 \rightarrow 5$	2.3613	18	1.02961
$5 \rightarrow 6$	4.6153	15	0.72616
$6 \rightarrow 7$	7.7108	17	0.839342

TABLE III

PLANE SEGMENTATION OF THE VISION-BASED POINT CLOUD SECTIONS.

Nr.	Points	Preprocessing [s]	Segmentation [s]	Planes
1	30745	0.0035	2.4906	53
2	49258	0.0067	6.9224	109
3	53586	0.0079	11.2961	145
4	48600	0.0072	10.2647	149
5	32366	0.0055	6.2082	111
6	28300	0.0050	6.4789	108
7	9754	0.0028	2.0793	61

TABLE IV

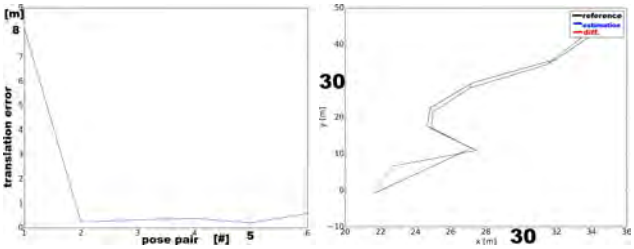
REGISTRATION OF THE LASER POINT CLOUDS WITH THE GLOBAL VISION-BASED POINT CLOUD.

Pair	Registration [s]	Correspondences	ICP [s]
1 Laser → Camera	0.012568	6	-
2 Laser → Camera	0.2772	7	0.480169
3 Laser → Camera	0.5608	11	-
4 Laser → Camera	0.4258	9	0.630182
5 Laser → Camera	0.5098	14	-
6 Laser → Camera	1.3474	20	-
7 Laser → Camera	0.0896	0	0.861286

TABLE V

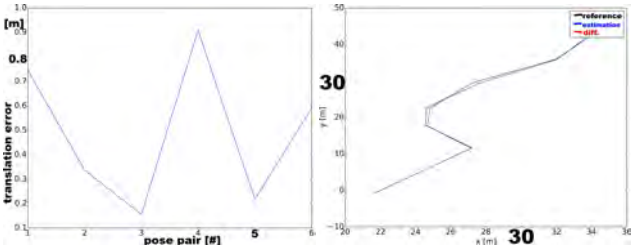
RELATIVE POSE ERROR AND ABSOLUTE TRAJECTORY ERROR OF THE RELATIVE LOCALIZATION.

Metric	E_{rmse} [m]	E_{min} [m]	E_{max} [m]
RPE	3.3579	0.2090	8.1820
ATE	3.1074	0.6780	7.4754



(a) RPE (b) ATE

Fig. 4. RPE and ATE of the relative localization.



(a) RPE (b) ATE

Fig. 5. RPE and ATE of the globally optimized localization.

V. EXPERIMENTS

In this section the results of the planar segment-based localization are evaluated. For the test environment the former site of the blast furnace Phoenix-West in Dortmund was selected (see fig. 8).

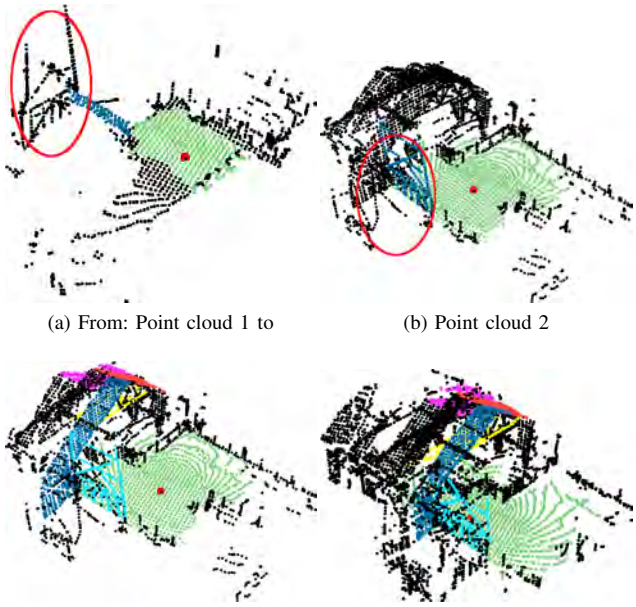
The UGV started near the entrance area of the factory building, drove further into the hall and finished the recordings there. A total of 7 laser scans were recorded. The relative pose error and absolute trajectory error (RPE / ATE) of the estimated trajectory after [28] were used as evaluation criteria for the localization. In order to obtain a reference trajectory of the UGV, adjacent laser point clouds were aligned relative to each other and the poses were subsequently refined with the SLAM framework 3DTK [29]. The vision-based point cloud for the following experiments was generated by MVE with images at a resolution of 640×480 pixels. MVE was choosen since it provides convincing results with respect to the estimated trajectory as well as the Mean Plane Variance (MPV) and Mean Map Entropy (MME), which were computed according to [30]. However the approach presented in this work is not limited to MVE.

The processing times of the pose tracking without further optimization steps are listed in the tables I and II. The RPE and ATE are represented in table V and fig. 4. The errors in the trajectory are caused by less accurate registration of the first two point clouds. The reason for this are inadequate structural elements, that do not allow accurate estimation of all directions of translation (see fig. 5). For the globally optimized trajectory, the results listed in table III and table IV were obtained. The deviations in the trajectory could be reduced by the optimization. In contrast to the relative localization, the global point cloud enabled the correct registration of the first two point clouds by additional structures (see fig. 6 and fig. 7, for the RPE and ATE). The overall registration result is shown in fig. 9. The initial localization was evaluated as a final test. The test sequence locates each laser point cloud in the global point cloud. The first five point clouds could be located correctly. The two last point clouds represented areas within the factory and had too little overlap with the global point cloud.

VI. CONCLUSION AND FUTURE WORK

The base for human to robot and robot to robot collaboration is a persistent environment model, which implies to fuse different sensor streams of different modalities. We present a novel approach for the plane-based localization of laser point clouds (UGV) in monocular vision point clouds (UAV). The method first performs a plane segmentation and then attempts to register neighbouring point clouds by means of corresponding planes. The method uses a global point cloud generated by the UAV's camera recordings for optimization. The evaluation showed that the relative localization provided a reliable registration and is therefore suitable as an initial estimation for global optimization. Point clouds, which had inadequate structures or slight overlaps with neighbouring point clouds, prevented accurate registration. Differences in the relative localization could be offset by the global optimization. A further important component of the global localization is the determination of the initial pose of the UGV. We suggested an automatic search of the start sector with subsequent registration. Areas with more than 50% overlap were successfully localized. When evaluating the vision-based procedures, it turned out that MVE ([15]) is best suited for planar segment-based registration.

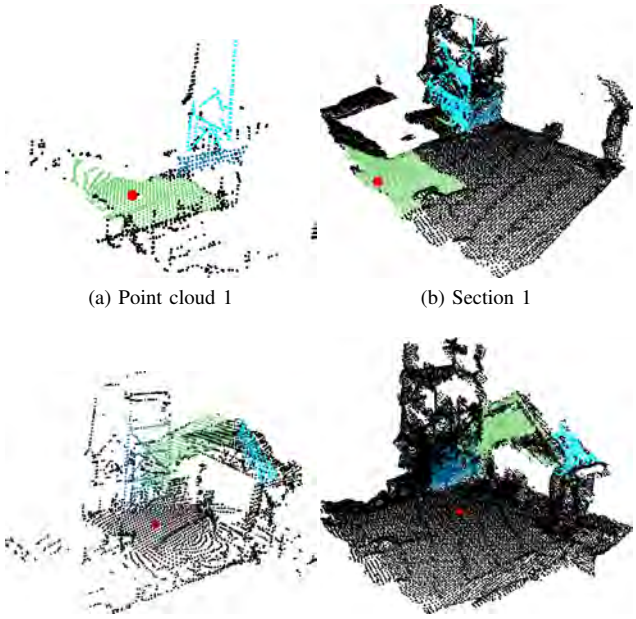
The localization method developed in this work will be extended for future work, e.g. by the utilization of additional



(a) From: Point cloud 1 to (b) Point cloud 2

(c) From: Point cloud 2 to (d) Point cloud 3

Fig. 6. Registration of the first three point clouds. Corresponding planes are randomly colored. The red marked area in image a could not be captured from the laser scanner in image b and thus offers no possibility for determining the direction of translation.



(a) Point cloud 1 (b) Section 1

(c) Point cloud 2 (d) Section 2

Fig. 7. Registered pairs of the globally optimized localization.

sensor information. For example, laser point clouds with color information can be generated by the camera on the UGV. For the correspondence search, this color information, in addition to the surface area, forms a further useful criterion for matching surfaces. When possible, GPS coordinates can also be used to support the localization.

REFERENCES

[1] R. R. Murphy, "Marsupial and shape-shifting robots for urban search and rescue," *Intelligent Systems and their Applications, IEEE*, vol. 15, no. 2, pp. 14–19, 2000.

[2] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-D," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[4] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera SLAM," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[5] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2320–2327.

[6] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision (ICCV)*, Sydney, Australia, December 2013.

[7] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[8] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, September 2014.

[9] R. Mur-Artal and J. D. Tardós, "Probabilistic semi-dense mapping from highly accurate feature-based monocular SLAM," *Proceedings of Robotics: Science and Systems, Rome, Italy*, vol. 1, 2015.

[10] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[11] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Pattern Recognition (Proc. DAGM)*, Darmstadt, Germany, September 2010, pp. 11–20.

[12] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

[13] A. Concha and J. Civera, "Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence," in *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[14] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[15] S. Fuhrmann, F. Langguth, and M. Goesele, "MVE – a multiview reconstruction environment," in *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, vol. 6, no. 7, 2014, p. 8.

[16] K. Pathak, N. Vaskevicius, J. Poppinga, M. Pfingsthorn, S. Schwertfeger, and A. Birk, "Fast 3D mapping by matching planes extracted from range sensor point-clouds," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 2009.

[17] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[18] D. Viejo and M. Cazorla, "3D plane-based egomotion for SLAM on semi-structured environment," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 2761–2766.

[19] K. Pathak, A. Birk, N. Vaskevicius, M. Pfingsthorn, S. Schwertfeger, and J. Poppinga, "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

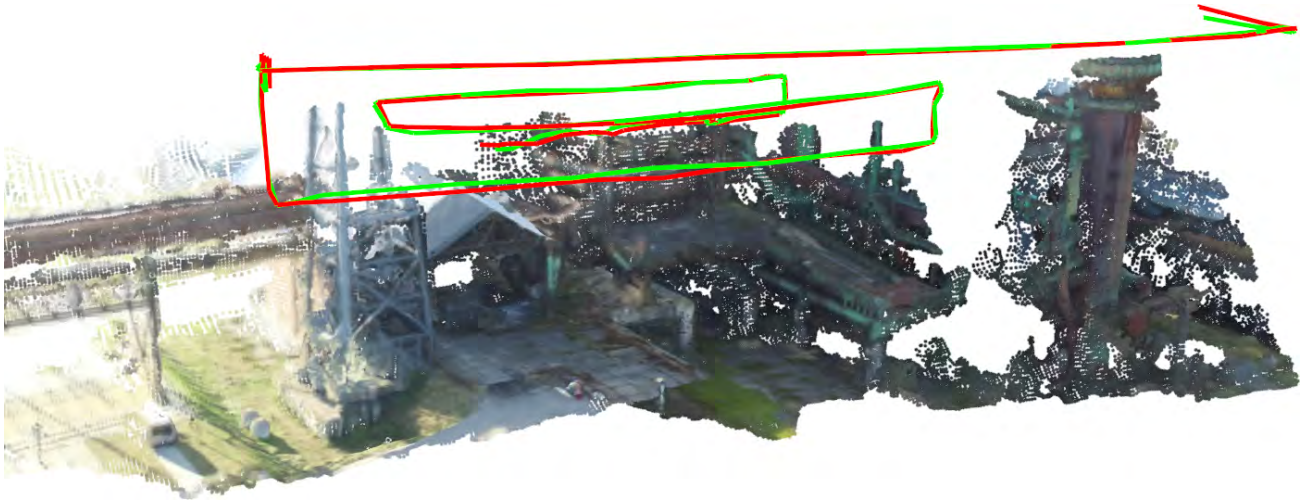


Fig. 8. Merged point cloud of the former site of the blast furnace Phoenix-West in Dortmund. The green trajectory was determined with MVE and the red trajectory with GPS. Both trajectories overlap to a large amount, which shows a good estimation of the trajectory. The point clouds were generated with MVE at a resolution of 640×480 pixels.

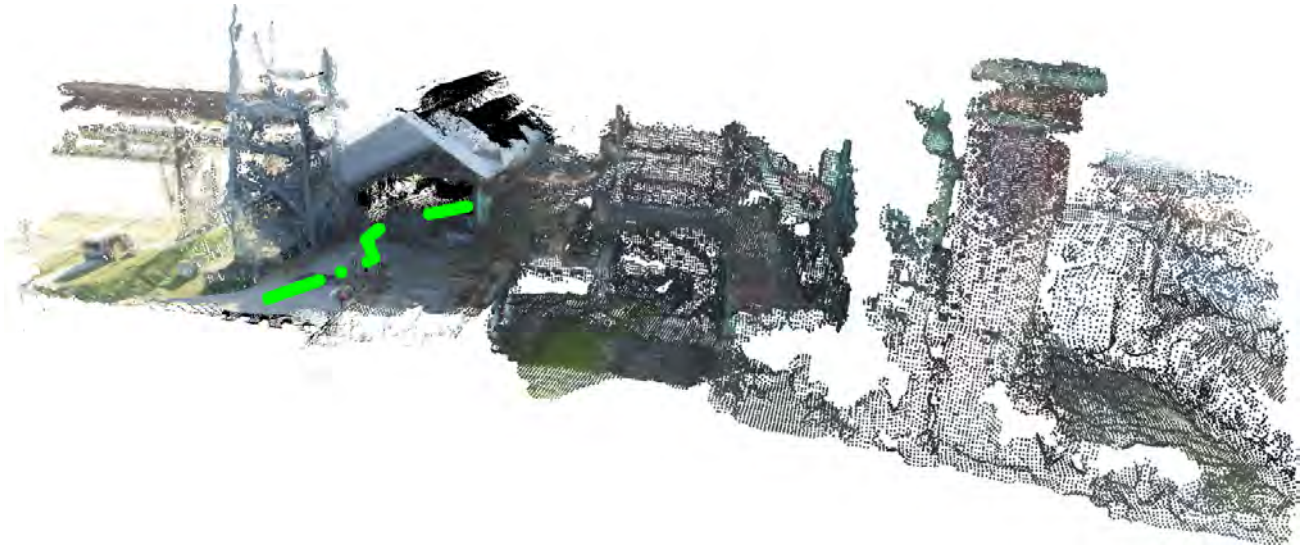


Fig. 9. Registered point clouds of the globally optimized localization. Points of the laser point clouds were dyed with the color information of the global point cloud. Within the factory no color information could be extracted. The estimated trajectory is shown in green.

[20] J. Poppinga, N. Vaskevicius, A. Birk, and K. Pathak, "Fast plane detection and polygonalization in noisy 3D range images," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 3378–3383.

[21] D. Hähnel, W. Burgard, and S. Thrun, "Learning compact 3D models of indoor and outdoor environments with a mobile robot," *Robotics and Autonomous Systems*, vol. 44, no. 1, pp. 15–27, 2003.

[22] K. Pathak, A. Birk, N. Vaskevicius, and J. Poppinga, "Fast registration based on noisy planes with unknown correspondences for 3-D mapping," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 424–441, 2010.

[23] J. Xiao, B. Adler, J. Zhang, and H. Zhang, "Planar segment based three-dimensional point cloud registration in outdoor environments," *Journal of Field Robotics*, vol. 30, no. 4, pp. 552–582, 2013.

[24] A. Gawel, T. Cieslewski, R. Dubé, M. Bosse, R. Siegwart, and J. Nieto, "Structure-based vision-laser matching," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 182–188.

[25] Z. C. Marton, R. B. Rusu, and M. Beetz, "On Fast Surface Reconstruction Methods for Large and Noisy Datasets," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, May 12-17 2009.

[26] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 181–198, 2003.

[27] T. Schmiedel, E. Einhorn, and H.-M. Gross, "IRON: A fast interest point descriptor for robust NDT-map matching and its application to robot localization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 3144–3151.

[28] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[29] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM – 3D mapping outdoor environments," *Journal of Field Robotics (JFR)*, vol. 24, no. 8-9, pp. 699–722, 2007.

[30] J. Razlaw, D. Droeschel, D. Holz, and S. Behnke, "Evaluation of registration methods for sparse 3D laser scans," in *Mobile Robots (ECMR), 2015 European Conference on*. IEEE, 2015, pp. 1–7.

An Online Multi-Robot SLAM System for 3D LiDARs

Renaud Dubé Abel Gawel Hannes Sommer Juan Nieto Roland Siegwart Cesar Cadena*

***Abstract*— Using multiple cooperative robots is advantageous for time critical Search and Rescue (SaR) missions as they permit rapid exploration of the environment and provide higher redundancy than using a single robot. A considerable number of applications such as autonomous driving and disaster response could benefit from merging mapping data from several agents. Online multi-robot localization and mapping has mainly been addressed for robots equipped with cameras or 2D LiDARs. However, in unstructured and ill-lighted real-life scenarios, a mapping system can potentially benefit from a rich 3D geometric solution. In this work, we present an online localization and mapping system for multiple robots equipped with 3D LiDARs. This system is based on incremental sparse pose-graph optimization using sequential and place recognition constraints, the latter being identified using a 3D segment matching approach. The result is a unified representation of the world and relative robot trajectories. The complete system runs in real-time and is evaluated with two experiments in different environments: one urban and one disaster scenario. The system is available open source and easy-to-run demonstrations are publicly available.**

I. INTRODUCTION

Teams of autonomous mobile robots offer several advantages compared to their single robot counterpart: robustness to single robot failure, quicker exploration of environments in time critical SaR missions, execution of tasks of high complexity and reduction of human risks and costs associated to disaster response [1, 2]. Accurate online localization and mapping is a crucial competency for enabling collaboration between multiple mobile robots. This is however a difficult task as stated by Saeedi et al. [3] which identified 10 major challenges to achieve online multi-robot Simultaneous Localization and Mapping (SLAM). The current work addresses the challenges of *closing loops*, *complexity* and *communication*. Inter-robot global associations are found and used to solve the full 3D SLAM problem with low time, memory and communication bandwidth requirements.

There are already existing vision-based multi-robot SLAM approaches [3] which can however become unreliable when strong changes in illumination occur, and in the presence of strong viewpoint variations [4]. In this work, we therefore consider 3D LiDARs for their ability to accurately represent the inherent 3D nature of our environment while providing higher robustness to changes in external illumination and in view-point.

*Authors are with the Autonomous Systems Lab, ETH, Zurich {rdube, gawela, sommerh, jnieto, rsiegwart, cesarc}@ethz.ch.

This work was supported by the European Union's Seventh Framework Programme for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763.

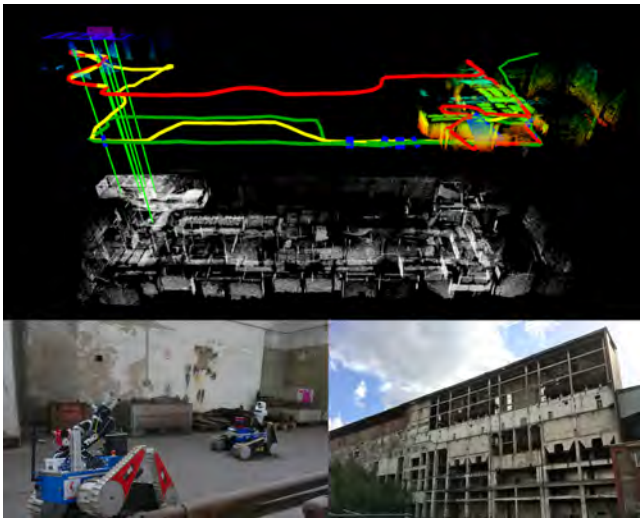


Fig. 1: Top: An illustration of the presented multi-robot SLAM system. The trajectories of three UGVs are estimated in real-time and shown in green, yellow and red. The target map M_t is shown below in white and an inter-robot place recognition is depicted with vertical green lines indicating segment matches. Bottom: Two skid-steering UGVs equipped with encoders, IMUs and rotating LiDARs and the decommissioned two-floors building considered in the power plant experiment of Section V-E.

Current multi-robot SLAM systems for 3D LiDARs do not offer a complete online solution [5, 6] which is perhaps due to the absence of efficient algorithms to perform global data association with dense 3D point clouds. This is contrastingly a well-studied problem in visual SLAM. Global place recognition techniques for 3D point clouds based on global descriptors [7, 8] and keypoint descriptors [9, 10] are presented but rarely integrated in a full online SLAM system, let alone a multi-robot one. Contrastingly, most of the current 3D LiDAR single-robot approaches propose to recognize places based on local *submap* matching. These local searches cannot correct for drift which occurs when long distances are travelled and large estimation errors accumulate. In the multi-robot case, finding robot-associations by performing global search based on *submaps* does not scale well with increasing number of robots and would require the raw LiDAR data to be transmitted, which reflect the aforementioned multi-robot challenges.

This paper presents an online 3D LiDAR SLAM system capable of simultaneously and accurately estimating multiple trajectories, as illustrated in Figure 1. To the best of our knowledge, this is the first proposed solution to the online multi-robot SLAM problem for 3D LiDARs. To achieve this, a pose-graph formulation is adopted by incorporating sequential and place recognition constraints. We perform intra and inter-robot place recognition by leveraging our

previously proposed and publicly available *SegMatch* algorithm [11] which was one key ingredient, along with significant implementation efforts, in order to achieve a full working system. The *SegMatch* technique is formed on the basis of partitioning point clouds into sets of segments which efficiently represent the environment by compact yet discriminative features. This compact representation is crucial for multi-robot applications as it reduces the required communication bandwidth as well as the complexity and the memory requirement of the overall system. This is reflected in the proposed system which offers real-time performance for the experiments considered in this paper.

To summarize, this paper presents the following contributions:

- A fully-integrated online multi-robot SLAM system for 3D LiDARs.
- An evaluation of the entire system in real-world, multi-robot automotive and disaster scenario experiments.
- An open-source implementation accompanied with easy-to-run demonstrations[1].

The remainder of the paper is structured as follows: Section II provides an overview of the related work in the field of 3D LiDAR-based SLAM and multi-robot SLAM. Section III and IV describe our online multi-robot 3D pose-graph SLAM system. The full system is evaluated in Section V, and Section VI finally concludes with a short discussion.

II. RELATED WORK

This section gives an overview of the related work in single-robot 3D LiDAR-based SLAM, with a focus on pose-graph based approaches, and present current solutions to the multi-robot mapping problem.

A. Single robot pose-graph SLAM

The recent survey of Cadena et al. [12] reviews common approaches to the SLAM problem. This work considers the pose-graph approach which was pioneered by Lu and Milios [13] and became increasingly popular in recent years with an active community performing research in this direction [14–19].

Several works propose to apply the pose-graph approach to perform 3D SLAM for single robots equipped with LiDAR sensors [15–19]. These works mainly differ in terms of the technique used for matching new 3D scans to previous ones. For example, Pathak et al. [15] propose to register subsequent 3D scans on the basis of large planar surfaces which leads to robust estimation of rotations and simplifies the pose-graph relaxation to handle 3D translations only. This assumption of an explicit plane model would however typically not hold for unstructured environments. Droeschel et al. [16] introduce a multi-resolution surface element representation for the 3D scans which are obtained by accumulating scans from a rotating 2D LiDAR sensor. Matching the scans through this surface elements representation allows for efficient registrations. The systems proposed in [17–19], and ours are all based on Iterative Closest Point (ICP) for registering successive 3D scans and for augmenting the pose-graph with the corresponding scan-matching constraints.

Although related to this work, none of the aforementioned approaches explicitly deal with loop-closures. Instead, scan matching is performed against *submaps* containing nodes in the vicinity of the robot, assuming that only little drift occurred. In our system, place recognitions are explicitly dealt with, allowing the fusion of maps from independent workers and enabling the joint exploration of larger environments.

B. Multi-robot pose-graph SLAM

A thorough survey on multi-robot SLAM can be found in [3]. There is a significant amount of works proposing solutions to the SLAM problem for robots equipped with cameras or 2D LiDAR but much fewer works consider 3D LiDAR sensors [5, 6, 20].

Nagatani et al. [5] propose to merge digital elevation maps obtained from three robots where inter-robot constraints are found on the basis of *submap* matching by assuming little drift and a known good estimate of the relative transformation between the robots. The map-merging strategy is performed offline and the experiment only consider a small environment. Michael et al. [6] present a strategy for generating a 3D map of a building damaged by an earthquake. The maps are locally built on each robot using a technique which assumes the environment to be composed of walls and horizontal ground planes. The two maps are merged afterwards, providing a good initial guess for the relative robot transformation which is then refined by ICP. The two aforementioned solutions are not applicable to online multi-robot SLAM and cannot correct for drift which might occur in the single maps. Finally, Kurazume et al. [20] propose to model large buildings using multiple robots, one of which is equipped with a 3D LiDAR. The other robots are used to improve the sensor's localization by using direct inter-robot detection. Although the idea is interesting, this system differs from ours in that a single 3D LiDAR is used.

As we could not find a single work presenting an online SLAM system for multiple robots equipped with 3D LiDARs, we briefly introduce major multi-robot systems based on vision, with a focus on the back-end particularities. The work of Kim et al. [21] addresses the multi-robot mapping problem based on incremental optimization of multiple relative pose-graphs. These graphs are linked through *anchor nodes* which allow a relative formulation of the inter-robot encounter detections using april-tags. *Anchor nodes* which are the equivalent of *base nodes* first introduced in [22] were also used in a multi-session vision-based SLAM system [23]. Anchor nodes are agnostic to the sensor used and help in the convergence of the back-end pose-graph optimization. Contrastingly, Konolige and Bowman [24] introduced *weak-links* to allow each robot's pose-graph to grow independently while keeping the problem constrained in the absence of

[1]The source code of the proposed system is open-sourced and demonstrations including a newly available dataset for multi-robot mapping in SaR scenarios are available at `https://github.com/ethz-asl/segmatch`. A video demonstration is available at `https://www.youtube.com/watch?v=JJhEkIA1xSE`

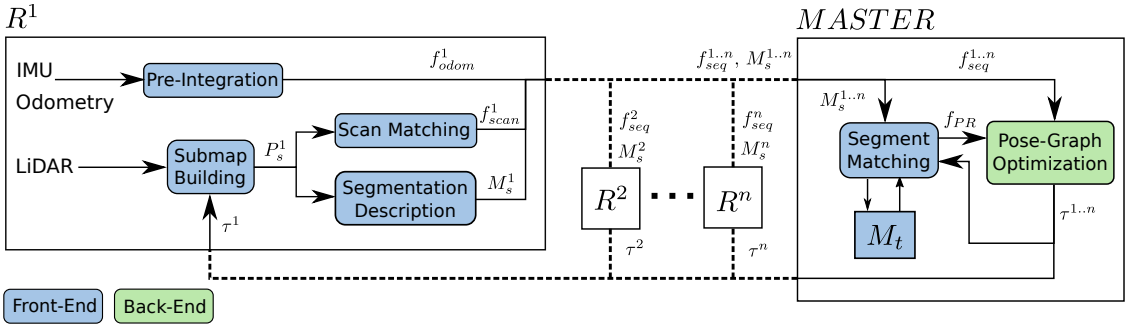


Fig. 2: A configuration of the proposed centralized multi-robot SLAM system where place recognition is based on a segment matching algorithm. The robots R^i locally compute odometry and scan-matching factors f^i_{odom} and f^i_{scan}. The 3D source point clouds P^i_s are segmented into maps M^i_s which are used by the master for generating place recognition factors f_{PR}. Once the pose-graph is optimized by the master, the trajectories updates $\boldsymbol{\tau}^i$ are used to maintain the target map M_t and transferred back to the robots. For simplicity, modules acting locally on each robot have been expanded only once.

inter-robot constraints. Our system is based on *weak-links* and could easily integrate *anchor nodes* instead if slow convergence is noticed during optimizations following inter-robot place recognitions.

III. System Architecture

This section introduces our multi-robot localization and mapping system based on 3D LiDAR and displacement measurements. As illustrated in Figure 2, the system is centralized such that a *master agent* is responsible of merging sensor information transmitted by multiple robots. At the core of the master agent lies an incremental pose-graph optimization back-end which is responsible for estimating the robot trajectories. On the other hand, the front-end is distributed between the robots and the master agent and is responsible for providing constraints to the optimization problem. For instance, each robot is responsible of computing sequential constraints, and of pre-processing the laser point clouds in order to compress the information to be transmitted over a communication channel of limited bandwidth.

The remainder of the section presents the back-end of the proposed multi-robot 3D SLAM system whereas Section IV details the front-end with the computation of LiDAR odometry, loop-closure and robot-association constraints.

A. Pose-graph formulation

The system is based on a pose-graph optimization approach [25] where a factor graph $G{=}(\mathcal{F}, \boldsymbol{\Theta}, \mathcal{E})$ is the underlying bipartite graph which connects all relevant elements of the system. It consists of factor nodes $\boldsymbol{f}_i{\in}\mathcal{F}$, variable nodes $\boldsymbol{\theta}_i{\in}\boldsymbol{\Theta}$ and edges $\epsilon_i{\in}\mathcal{E}$, that connect factor nodes with variable nodes. Variable nodes $\boldsymbol{\theta}_i$ are the states of the system and represent robot poses, i.e., $\boldsymbol{\theta}_i \in SE(3)$. Factor nodes $\boldsymbol{f}_i$ on the other hand are constraints between several poses.

The factor graph then defines the factorization of a function $\boldsymbol{f}(\boldsymbol{\Theta})$

$$\boldsymbol{f}(\boldsymbol{\Theta}) = \prod_i \boldsymbol{f}_i(\boldsymbol{\Theta}_i) \tag{1}$$

with $\boldsymbol{\Theta}_i$ being the subset of variables adjacent to the factor $\boldsymbol{f}_i$. Our system implements three different types of factor nodes, i.e., prior factors $\boldsymbol{f}_{prior}(\boldsymbol{\Theta}_0)$, sequential factors $\boldsymbol{f}_{seq,i}(\boldsymbol{\Theta}_i)$, and place recognition factors $\boldsymbol{f}_{PR,i}(\boldsymbol{\Theta}_i)$ which include both intra-robot loop-closures and inter-robot associations. Both $\boldsymbol{f}_{seq,i}(\boldsymbol{\Theta}_i)$ and $\boldsymbol{f}_{PR,i}(\boldsymbol{\Theta}_i)$ are always expressed as relative pose measurements which is particularly practical for multi-robot applications as the measurements are entirely independent from the fixed frame of reference. Finally, each factor is expressed in full 6 Degrees of Freedom (DOF).

B. Sparse incremental optimization

The aim of the back-end is to compute a Maximum A Posteriori (MAP) estimate of $\boldsymbol{f}(\boldsymbol{\Theta})$ given its observations $\tilde{\boldsymbol{z}}_i$ that minimizes a negative log-posterior $\boldsymbol{E}$. Assuming a Gaussian measurement model leads to Equations 2, 3 and 4.

$$\boldsymbol{f}_i(\boldsymbol{\Theta}_i) \propto \exp\left(-\frac{1}{2}||\boldsymbol{z}_i(\boldsymbol{\Theta}_i) - \tilde{\boldsymbol{z}}_i||^2_{\boldsymbol{\Omega}_i}\right) \tag{2}$$

$$\boldsymbol{E} = -\log \boldsymbol{f}(\boldsymbol{\Theta}) \tag{3}$$

$$\arg\min_{\boldsymbol{\Theta}}(\boldsymbol{E}) = \arg\min_{\boldsymbol{\Theta}}(\sum_i \boldsymbol{e}_i^T \boldsymbol{\Omega}_i \boldsymbol{e}_i) \tag{4}$$

where $e_i = \boldsymbol{z}_i(\boldsymbol{\Theta_i}) - \tilde{\boldsymbol{z}}_i$ is the error between the prediction function $\boldsymbol{z}_i(\boldsymbol{\Theta_i})$ and a measurement $\tilde{\boldsymbol{z}}_i$, with the information matrix $\boldsymbol{\Omega}_i$. In order to robustify the optimizer against false place recognitions, we add a Cauchy function as M-estimator to the place recognition factors as described in [26] which down-weights the effect of possibly wrong factors on the optimization objective $\boldsymbol{E}$.

As the prediction function $\boldsymbol{z}_i(\boldsymbol{\Theta})$ is nonlinear, we minimize the error $\boldsymbol{E}$ using nonlinear optimization with the Gauss-Newton algorithm. Specifically, we perform incremental update and optimization of the pose-graph using the iSAM2 algorithm [14] which allows for efficient variable re-ordering and relinearization using the Bayes tree.

Given that we use *weak links*, introduced by Konolige and Bowman [24], through the prior factor $\boldsymbol{f}_{prior}(\boldsymbol{\theta}_0)$, the Bayes tree approach is particularly suitable for updating the pose-graph resulting from the multi-robot problem. Once an inter-robot place recognition is detected, one prior is removed from the graph and iSAM2 allows for efficient re-ordering and relinearization of the variables.

IV. System front-end

In this section, we show one possible front-end configuration where sequential factors are created using ICP and displacement measurements and where the *SegMatch* algorithm is used for generating place recognition factors. As illustrated in the block diagram of Figure 2 the system is modular so that one could select and integrate different techniques for factors generation modules.

A. Sequential factors

This first front-end module is responsible for transforming 3D LiDAR and proprioceptive sensors measurements into sequential factors $\boldsymbol{f}_{seq,i}(\boldsymbol{\Theta}_i)$. In order to keep this module robust to failures when doing scan matching, we separate its contribution to the graph in two different types of factors i.e., odometry factors $\boldsymbol{f}_{odom,i}(\boldsymbol{\Theta}_i)$ and scan-matching factors $\boldsymbol{f}_{scan,i}(\boldsymbol{\Theta}_i)$.

A pose node $\boldsymbol{\theta}_i$ is added to the graph, along with one odometry and one scan-matching factor for every 3D laser-scan S_i, given that the robot travelled a minimum distance d_{min}. This minimum travel distance is a standard practice to help avoiding un-informative accumulation of data, and the growth of the factor graph, when the robot is not moving.

1) Odometry factors: Odometry factors $\boldsymbol{f}_{odom,i}(\boldsymbol{\Theta}_i)$ are based on displacements between consecutive robot poses as stated in Eq. 5 and are built using Eq. 2.

$$\boldsymbol{z}_{odom,i}(\boldsymbol{\Theta}_i) = \boldsymbol{\theta}_{i-1}^{-1} \oplus \boldsymbol{\theta}_i \tag{5}$$

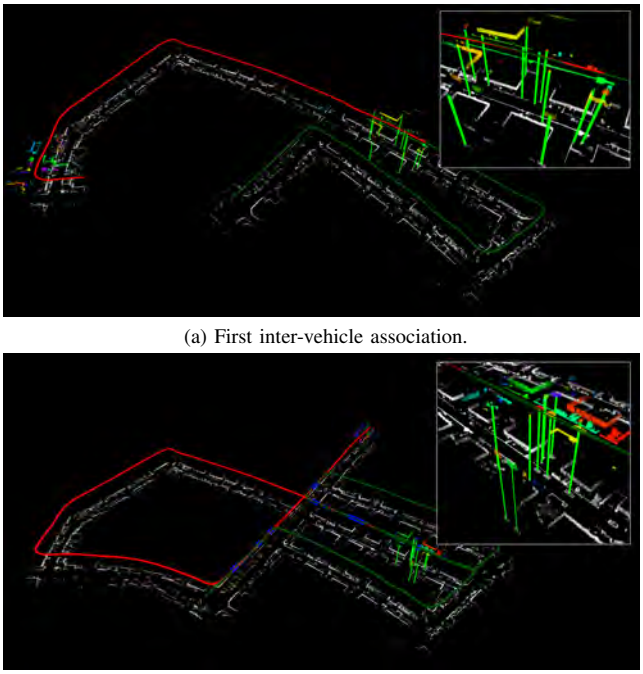
These odometry measurements can be obtained from different sources of proprioceptive sensors. For instance, for the UGVs illustrated in Figure 1, these constraints are obtained fusing wheel encoders and IMU data using an extended Kalman filter as proposed by Kubelka et al. [27].

2) Scan matching factors: The 3D LiDAR data is used to compute scan-matching factors $\boldsymbol{f}_{scan,i}(\boldsymbol{\Theta}_i)$ between adjacent nodes in the graph. These factors are obtained using ICP by registering the current scan against a *submap* composed of the m previous scans, expressed in the frame of the previous pose. The *submap* size is controlled by m, the number of previous scans and d_{min}, the minimum travelled distance between consecutive scans.

Performing ICP against this *submap* helps dealing with sparsity in the scans, eg. when working with Velodyne data, and helps making the system more robust. The ICP registration step between the *submap* and the new scan results in the 6 DOF rigid transformation ${}_i\boldsymbol{T}_{ij}$ which is a transformation from pose $\boldsymbol{\theta}_i$ to $\boldsymbol{\theta}_j$ expressed in the frame of $\boldsymbol{\theta}_i$. This transformation is directly used to build a laser scan-matching factor $\boldsymbol{f}_{scan,i}(\boldsymbol{\Theta}_i)$ using the distance between the relative transformation prediction and measurement as an error function, i.e. $e = d({}_i\boldsymbol{T}_{ij}, {}_i\tilde{\boldsymbol{T}}_{ij})$.

B. Place recognition factors

Our multi-robot SLAM system has a flexible implementation which can receive place recognition candidates from different sources. In the present work, we propose one configuration where associations are generated using the *SegMatch* algorithm introduced in [11]. This algorithm takes as input (i) a source point cloud P_s representing local LiDAR measurements and (ii) a target point cloud P_t against which place recognition is performed. Segments are extracted and described from both P_s and P_t to respectively generate a source map M_s and a target map M_t which contain lists of low dimensional segments descriptors. In this work we consider two region growing algorithms for extracting segments from 3D point clouds: one based on Euclidean distance [11] and the other based on smoothness constraints [28]. Compact eigenvalue based features are used for describing the segments [11]. Associations are made between M_s and M_t and a geometrical verification step is used to identify clusters of matches which represent place recognitions as illustrated in Figure 3.



(a) First inter-vehicle association.

(b) Last intra-vehicle association.

Fig. 3: Two autonomous vehicles mapping KITTI sequence 05 using LiDAR information only. The trajectories are shown in red and green and the source map segments are shown in colors. The optimized associations are show in blue with the latest association shown by segment matches with vertical green lines. The target map M_t is shown in white below.

The remainder of this section will detail how segment matching is used within the full system and the reader is encouraged to consult our prior work for more information about the *SegMatch* algorithm [11].

1) Source map generation: As illustrated in Figure 2, each robot in the system is responsible of generating its own segment source map M_s^j where j denotes the robot's unique identifier. M_s^j is, along with the sequential factors $\boldsymbol{f}^j_{odom,i}(\boldsymbol{\Theta}_i)$ and $\boldsymbol{f}^j_{scan,i}(\boldsymbol{\Theta}_i)$, the only information communicated to the centralized master. Converting the raw point cloud P_s^j into M_s^j induces a high level of compression which is a key advantage for reducing the required communication bandwidth in multi-robot systems.

The intermediate source point cloud representation P_s^j is created by accumulating the 3D scans S_i once the corresponding nodes $\boldsymbol{\theta}_i$ are optimized by the back-end. Noisy data are filtered using a voxel grid of resolution res_{voxel} with n_{min}, a minimum number of points per voxel to consider them as occupied. An octomap [29] can also be used if one further wishes to filter dynamics. The growth of P_s^j is limited by extracting a cylindrical neighbourhood of radius R, centred around the current robot location. Applying this cylindrical filter inevitably results in cut objects, which then results in 'incomplete segments' in M_s^j that can interfere with 'complete views' in the target map M_t. These 'incomplete segments' are detected by filtering P_s^j with a smaller radius $r = R - b$, where b is the thickness of the outer zone. Segments containing points within that zone are discarded from M_s^j which is transferred to the master and used for both matching and building the target map M_t.

2) Incremental target map management: Given our centralized approach, the master agent is responsible of incorporating incoming source maps M_s^j into a single target map representation M_t. For each segment in M_s^j, we check for a duplicate in M_t, i.e. a segment resulting from the same object part, but extracted at different times. As single-robot odometry is locally accurate, these 'duplicate segments' can efficiently be detected by comparing the distances to the closest segments centroids in M_t with a minimum distance d_{seg}. As we prefer to keep the latest view of a segment, we choose to remove the oldest of these duplicates.

In event of place recognitions, M_t is updated with a similar approach. Given the updated robot trajectories, the positions of the target segments are first refreshed, knowing the origin of their segmentation relative to the trajectories. In case of successful place recognitions, segments of the target map will correctly align and can safely be filtered for removing duplicates as described above.

3) Place recognition factor generation: Given M_s and M_t, segment matching is performed through k-Nearest Neighbors (k-NN) retrieval. Afterwards, a geometric verification step based on RANSAC identifies clusters containing at least min_{RANSAC} segment matches that are geometrically consistent with a resolution res_{RANSAC}. In order to convert these segment matches in the form of place recognition factors $\boldsymbol{f}_{PR,i}(\boldsymbol{\Theta}_i)$, the nodes to be constrained by the factors are first identified. For both the source and the target, we select the trajectory node that is, on average, the closest to all corresponding segments and which lies within a time window defined by the segments' timestamp.

As the segments centroids are represented in world frame when doing geometrical verification, the resulting relative transformation will also be given in world frame, i.e. ${}_w\boldsymbol{T}_{ij}$. This transformation can be expressed in the frame of the first node θ_i using Eq. 6 and converted into a factor as described in Section IV-A.

$$ {}_i\boldsymbol{T}_{ij} = \boldsymbol{\theta}_i{}^{-1} \oplus_w \boldsymbol{T}_{ij} \oplus \boldsymbol{\theta}_j \tag{6} $$

Given these new factors, the pose-graph is incrementally optimized and M_t is updated as explained in IV-B.2. As shown in Figure 2, the resulting trajectory updates $\boldsymbol{\tau}^j$ are transmitted back to the individual robots which then update P_s^j to follow the transformation applied to the trajectory head. The performances of this *SegMatch* based place-recognition module are evaluated in the following section.

V. EXPERIMENTS

In this section we demonstrate the performance of the proposed system through two different multi-robot experiments. In the first experiment, we adapt sequence 05 of the KITTI odometry dataset [30] in order to generate a multi-robot scenario. The second experiment is based on data collected during a SaR mission performed by the "Long-Term Human-Robot Teaming for Robots Assisted Disaster Response" (TRADR) consortium[2] at the Gustav Knepper Power Station in Dortmund, Germany.

A. Implementation details

This section briefly presents implementation details which can be relevant when exploring the system. The system is built on multiple available libraries, as for instance, the incremental optimization back-end which is based on the iSAM2 implementation of the GTSAM library[3]. The factor's Jacobians are evaluated using the block automatic differentiation functionality of GTSAM. For creating scan-matching factors we use the ICP implementation of $libpointmatcher$[4]. The place recognition implementation is based on the *SegMatch* library[5] which itself uses PCL[6] for voxel grid filtering and geometric verification functionalities and $libnabo$[7] for segment matching with fast k-NN search in low dimensional space. The system has a full ROS interface with integration to the TF tree for publishing the estimated robot poses.

B. Experimental setup

The two following experiments are performed on a single computer equipped with an Intel i7-4900MQ CPU @ 2.80GHz and 32 GB of DDR3 RAM. In order to realize the multi-robot scenarios, the system is implemented with multiple threads. One thread per robot is used for computing the sequential factors and the local maps whereas the place recognition, pose-graph optimization and target map management functionalities are all running on a separate thread. For real missions, the latter thread would run on the master agent and the computational load would be distributed amongst the multiple computers.

C. System parametrization

The front-end parameters used for both experiments are resumed in Table I. For the place recognition module, segment retrieval is always performed using k-NN. Different segmentation algorithms and segment descriptors are used

[2] http://www.tradr-project.eu/
[3] https://research.cc.gatech.edu/borg/gtsam
[4] https://github.com/ethz-asl/libpointmatcher
[5] https://github.com/ethz-asl/segmatch
[6] http://pointclouds.org/
[7] https://github.com/ethz-asl/libnabo

TABLE I: Front-end parameters for the two experiments.

Parameter	KITTI	Power plant
Min. distance between poses (d_{min})	0.05 meter	0.1 meter
Number of scans per *submap* (m)	5	10
Voxel grid resolution (res_{voxel})	0.1 meter	0.1 meter
Min. point count per voxel (n_{min})	1	2
Source map radius (R)	60 meters	25 meters
Number of neighbors (k)	45	20
Min. RANSAC cluster size (min_{RANSAC})	5	5
RANSAC resolution (res_{RANSAC})	0.45 meter	0.55 meter
Min. segment distance (d_{seg})	2 meters	2 meters

which reflects the difference in sensor configuration, vehicle speed and environment between the two experiments. For the multi-robot KITTI experiment, Euclidean segmentation and eigenvalue based features are adopted, as described in [11]. For the power plant experiment, segments are obtained based on region growing with smoothness constraints [28]. Normals are computed with a radius of 0.5 meters and the 15 nearest neighbours of each seed are considered for growing segments. The threshold on the difference of normals is set to 8 degrees and we keep only segments having a minimum of 75 points. In addition to the eigenvalue based features, we use the fact that the power plant environment contains many vertical and horizontal planar surfaces and make a distinction between these by adding a single value feature describing its orientation. This can efficiently be computed by comparing the x, y, and z dimensions of the segments.

For the two platforms considered in the experiments, the noise models could easily be adjusted on different datasets in order to yield good localization results.

D. LiDAR-only multi-vehicular KITTI

In this first experiment, we split sequence 05 of the KITTI odometry dataset which lasts 2.2 km and 287 seconds into two sequences of equal duration. A multi-robot scenario is simulated by simultaneously playing back the two sequences.

With this experiment, we aim to show the capabilities of our system for performing online multi-robot SLAM by using LiDAR information only. In other words, the scan-matching factors $\boldsymbol{f}_{scan,i}(\boldsymbol{\Theta}_i)$ are the only sequential factors considered in this experiment. A constant velocity model is adopted for generating initial guesses to the ICP based registrations, acting as measurement functions for these factors. The rest of the parametrization is introduced in V-C.

On this multi-robot sequence and using LiDAR measurements only, our system could in real-time detect 18 valid intra and inter-robot global associations. The first and last associations are illustrated in Figure 3 along with the two estimated trajectories. The timings for each module are stated in Table II with an indication whether this module should be executed by the robot (R) or by the master agent (M). Cumulative computation times are given in Figure 4.

During this experiment, 154 source point clouds P_s^i were processed for place recognition and contained in average 96200 points (after downsampling, voxelization and cylindrical filtering). With each point defined by three doubles and assuming a typical size of 8 bytes per double, communicating

TABLE II: Timing of each module (in ms).

Module (Agent)	KITTI	Power plant
Scan-matching (R)	78.9	916.8
Trajectories estimation (M)	3.7	4.3
Trajectories estimation after PR (M)	43.1	24.9
Segmentation and description (R)	600.1	1176.0
Segment matching (M)	8.1	6.0
Geometric verification (M)	57.2	13.8
Duplicates removal (M)	91.8	21.2

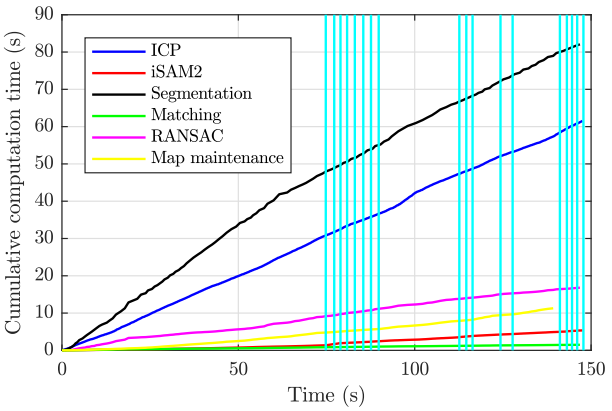


Fig. 4: Cumulative computation time for each module during the LiDAR-only multi-robot KITTI experiment. The 18 valid place recognitions are shown with vertical lines. During this experiment, no false place recognition was detected.

the source clouds to the master computer would require the transmission of 356 MB of data. In comparison, 5MB would be transmitted if one would select 100 keypoints per source cloud and describe these with Fast Point Feature Histograms (FPFH) [31] of dimension 1x33. With our segment approach, the source maps M_s^i contained in average 30.4 $\pm$ 9.9 descriptors for a total of 4682 descriptors transmitted to the master computer. With the compact representation of eigenvalue based features [1x7], this results in only 562 kB of data to be transmitted over 143 seconds of operation. Note that for these computations, six doubles and two unsigned int are additionally required to link each descriptor to the trajectories. We also only treat the *useful data* and do not consider the data transfer overhead.

E. Gustav Knepper Power Station

For this experiment, we use data collected during a SaR mission performed by the TRADR consortium at the decommissioned Gustav Knepper Power Station. The experiment took place in one large two-floors utility building measuring 100m long by 25m wide illustrated in Figure 1. During the experiment held in November 2016, three UGVs equipped with multiple encoders, an Xsens MTI-G IMU and a rotating 2D SICK LMS-151 LiDAR were teleoperated by firemen end-users in order to efficiently explore the scenario. With their skid-steering climbing capabilities, the TRADR UGVs can traverse and map challenging 3D environments.

The power plant mission lasted 950 seconds with the three robots starting at different locations and traversing different

paths with a cumulative distance of 694 meters. As can be seen in Figure 1, UGV_{green} began its mission outside, entered the building and performed a loop in clock-wise direction of the ground floor. Simultaneously, UGV_{red} first climbed challenging metal stairs at the right side of the ground floor in order to reach and explore the upper level whereas UGV_{yellow} started at the left side of the upper floor, went down and explored the ground floor.

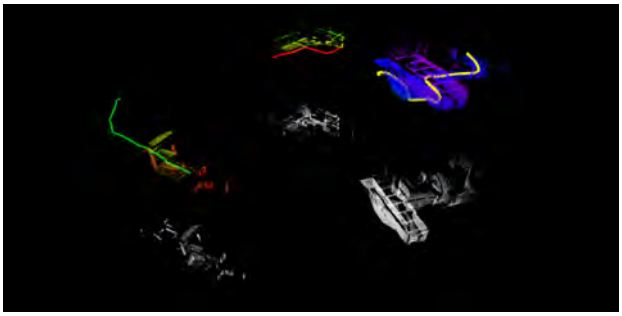
For this experiment, the odometry factors $\boldsymbol{f}_{odom,i}(\boldsymbol{\Theta}_i)$ are obtained by fusing encoders and IMU measurements with the technique presented by Kubelka et al. [27]. This odometry information is also used for accumulating measurements from the rotating 2D LiDAR sensor into dense 3D point clouds which are used to compute the scan-matching factors $\boldsymbol{f}_{scan,i}(\boldsymbol{\Theta}_i)$ as described in Section IV-A.2. The parametrization of the other modules is defined in Section V-C and Table I.

During the three-robot power plant experiment, our SLAM system with the presented configuration was capable of detecting 20 valid place recognitions, in real-time and on one single computer. Eight of these successful *SegMatch* detections were in challenging scenarios where the UGVs drove in opposite directions. The three trajectories, as estimated at different moments of the mission, are depicted in Figure 5. Figures 5b and 5c specifically illustrate the case where a global prior factor $\boldsymbol{f}_{prior}(\boldsymbol{\Theta}_0)$ is removed from the pose-graph due to a first inter-robot association, as described in Section III-B. Two of the eight place recognitions in areas visited in opposite directions are shown in Figure 5c and 5d. The final trajectories are illustrated in Figure 1 and a top-down view is given in Figure 6. As for the previous experiment, the timings are stated in Table II.

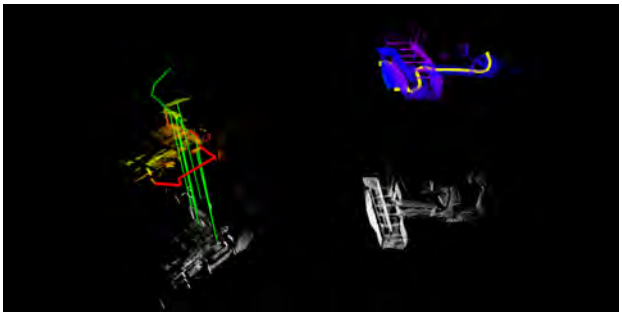
VI. CONCLUSION

This paper presented a SLAM system for multiple robots equipped with 3D LiDAR sensors. After considerable development efforts, our system successfully integrates different state of the art modules that are advantageous for multi-robot systems with regards to closing loops, computational complexity, and communication bandwidth requirements.

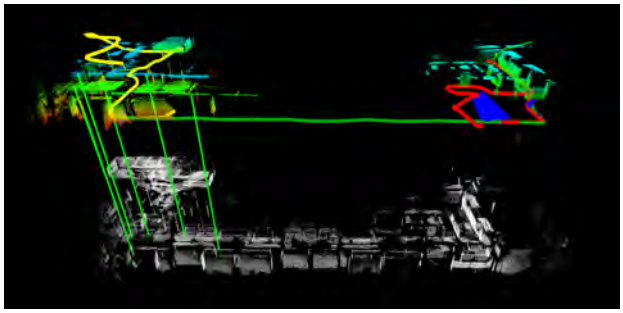
The system's back-end is based on a pose-graph optimization approach where updates and inference are performed incrementally using the Bayes tree. We showed how this architecture can easily be configured to handle multiple trajectories without any prior information on their relative position. The front-end is responsible of providing the graphical model with LiDAR odometry and place recognition constraints. The LiDAR odometry constraints relate successive nodes using ICP scan-matching between the latest scan and a *submap* of previous scans. Place recognition is performed by leveraging a segment extraction and matching algorithm. We demonstrated through two experiments in different scenarios that the presented system enabled multiple-robots to jointly map large areas in real-time and with a high rate of successful place recognitions. During these experiments, we found that identifying valid and accurate place recognitions is a crucial capability when dealing with multiple robots.



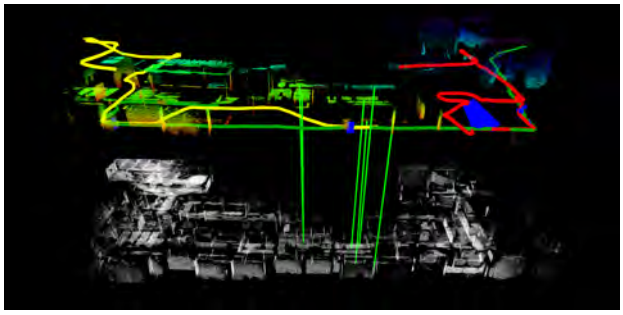
(a) Start of the mission with each robot in its own frame of reference.



(b) At $t = 180s$, first global association between UGV_{red} and UGV_{green}.



(c) At $t = 380s$, first encounter of UGV_{green} and UGV_{yellow} when exploring the lower floor in opposite directions.



(d) At $t = 476s$, another association is made between the same UGVs in a corridor traversed in opposite directions.

Fig. 5: A demonstration of our multi-robot SLAM system based on data collected in a two-floor building of the Gustav Knepper Power Station in Dortmund, Germany. The trajectories of UGV_{red}, UGV_{green} and UGV_{yellow} are estimated in real-time. The vertical green lines represent segment matches which resulted in one of the 20 inter-robot place recognitions. For visualization purposes, the source maps are coloured by height and the target map is illustrated in white below. The final trajectories are illustrated in Figure 1. The reader is encouraged to consult the video demonstration for a better visualization.

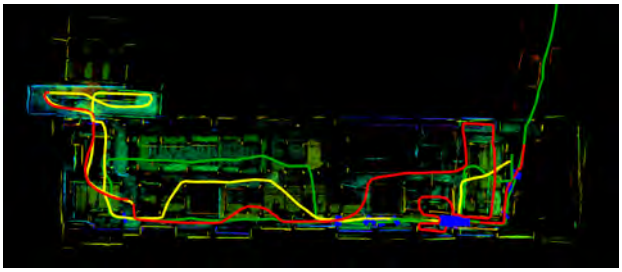


Fig. 6: A top down view of the multi-robot experiment in the Gustav Knepper Power Station. The target map M_t is coloured by height.

In the context of multi-robot systems acting in difficult scenarios and under limited communication bandwidth, one important challenge is to limit the data to be transmitted for finding robot associations [3]. One key advantage of the proposed system is to detect these associations on the basis of segments that offer a high level of descriptiveness and information compression.

Whereas this paper proposed a centralized system, future work could include a distributed system which would also directly benefit from these advantages. To this end, the implementation of the complete system is available online with easy to run demonstrations, along with the new SaR dataset introduced in this work.

References

[1] D. Tardioli, D. Sicignano *et al.*, "Robot teams for intervention in confined and structured environments," *Journal of Field Robotics*, 2015.

[2] I. Kruijff-Korbayová, F. Colas *et al.*, "Tradr project: Long-term human-robot teaming for robot assisted disaster response," *KI-Künstliche Intelligenz*, vol. 29, no. 2, pp. 193–201, 2015.

[3] S. Saeedi, M. Trentini *et al.*, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, vol. 33, no. 1, pp. 3–46, 2016.

[4] S. Lowry, N. Sunderhauf *et al.*, "Visual place recognition: A survey," *IEEE Trans. on Robotics*, 2016.

[5] K. Nagatani, Y. Okada *et al.*, "Multirobot exploration for search and rescue missions: A report on map building in robocuprescue 2009," *Journal of Field Robotics*, vol. 28, no. 3, pp. 373–387, 2011.

[6] N. Michael, S. Shen *et al.*, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[7] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3D lidar datasets," in *IEEE Int. Conf. on Robotics and Automation*, 2013.

[8] Y. Zhuang, N. Jiang *et al.*, "3-d-laser-based scene measurement and place recognition for mobile robots in dynamic indoor environments," *IEEE Transactions on Instrumentation and Measurement*, vol. 62, no. 2, pp. 438–450, 2013.

[9] T. Röhling, J. Mack, and D. Schulz, "A fast histogram-based similarity measure for detecting loop closures in 3-d lidar data," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[10] K. Granström, T. B. Schön *et al.*, "Learning to close loops from range data," *The Int. Journal of Robotics Research*, vol. 30, no. 14, pp. 1728–1754, 2011.

[11] R. Dubé, D. Dugas *et al.*, "Segmatch: Segment based place recognition in 3d point clouds," in *IEEE Int. Conf. on Robotics and Automation*, 2017.

[12] C. Cadena, L. Carlone *et al.*, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Trans. on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[13] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[14] M. Kaess, H. Johannsson *et al.*, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The Int. Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[15] K. Pathak, A. Birk *et al.*, "Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation," *Journal of Field Robotics*, vol. 27, no. 1, pp. 52–84, 2010.

[16] D. Droeschel, M. Schwarz, and S. Behnke, "Continuous mapping and localization for autonomous navigation in rough terrain using a 3d laser scanner," *Robotics and Autonomous Systems*, 2016.

[17] E. Mendes, P. Koch, and S. Lacroix, "Icp-based pose-graph slam," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 195–200.

[18] D. Borrmann, J. Elseberg *et al.*, "Globally consistent 3d mapping with scan matching," *Robotics and Autonomous Systems*, vol. 56, no. 2, pp. 130–142, 2008.

[19] A. Nüchter, K. Lingemann *et al.*, "6d slam—3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[20] R. Kurazume, S. Oshima *et al.*, "Automatic large-scale three dimensional modeling using cooperative multiple robots," *Computer Vision and Image Understanding*, 2016.

[21] B. Kim, M. Kaess *et al.*, "Multiple relative pose graphs for robust cooperative mapping," in *IEEE Int. Conf. on Robotics and Automation*, 2010.

[22] K. Ni, D. Steedly, and F. Dellaert, "Tectonic sam: Exact, out-of-core, submap-based slam," in *IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 1678–1685.

[23] J. McDonald, M. Kaess *et al.*, "Real-time 6-dof multi-session visual slam over large-scale environments," *Robotics and Autonomous Systems*, vol. 61, no. 10, pp. 1144–1158, 2013.

[24] K. Konolige and J. Bowman, "Towards lifelong visual maps," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1156–1163.

[25] G. Grisetti, R. Kümmerle *et al.*, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[26] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2013, pp. 556–563.

[27] V. Kubelka, L. Oswald *et al.*, "Robust data fusion of multimodal sensory information for mobile robots," *Journal of Field Robotics*, vol. 32, no. 4, pp. 447–473, 2015.

[28] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, pp. 248–253, 2006.

[29] A. Hornung, K. M. Wurm *et al.*, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[30] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[31] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE Int. Conf. on Robotics and Automation*, 2009, pp. 3212–3217.

Incremental Segment-Based Localization in 3D Point Clouds

Renaud Dubé*, Mattia G. Gollub*, Hannes Sommer, Igor Gilitschenski,
Roland Siegwart, Cesar Cadena, and Juan Nieto†

***Abstract*—Localization in 3D point clouds is a highly challenging task due to the complexity associated with extracting information from 3D data. This paper proposes an incremental approach addressing this problem efficiently. The presented method first accumulates the measurements in a dynamic voxel grid and selectively updates the point normals affected by the insertion. An incremental segmentation algorithm, based on region growing, tracks the evolution of single segments which enables an efficient recognition strategy using partitioning and caching of geometric consistencies. We show that the incremental method can perform global localization at 10Hz in a urban driving environment, a speedup of x7.1 over the compared batch solution. The efficiency of the method makes it suitable for applications where real–time localization is required and enables its usage on cheaper, low–energy systems. Our implementation is available open source along with instructions for running the system[1].**

***Index Terms*—Localization, Range Sensing, Recognition, SLAM**

I. INTRODUCTION

PERCEPTION capabilities are a key requirement for robots to perform high–level tasks like navigation and interaction. For this reason, mobile robots are often equipped with 3D time-of-flight sensors which can produce precise reconstructions of the environment. Processing these detailed data can however result in high computational costs. Amongst important capabilities which can strongly benefit from efficient solutions, we focus on the challenging task of localization in 3D point clouds. Making global associations in 3D data permits us to construct an unified representation without making an assumption of low drift, or known relative starting position, in case of multi-robot applications.

This work presents an incremental solution to localization in 3D point clouds based on the principle of segment extraction and matching from accumulated data [1]. As demonstrated in our previous work, accumulating data can help recognize places which are observed from different viewpoints [2].

Manuscript received: September 10th 2017; Revised December 6th 2017; Accepted January 3rd 2018.

This paper was recommended for publication by Editor Cyrill Stachniss upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the European Union's Seventh Framework Programme for research, technological development and demonstration under the TRADR project No. FP7-ICT-609763.)

*The authors contributed equally to this work. †Authors are with the Autonomous Systems Lab, ETH, Zurich. Corresponding authors: renaudube@gmail.com and gollubm@ethz.ch.

Digital Object Identifier (DOI): see top of this page.

[1]The implementation is available at https://github.com/ethz-asl/segmatch and a video demonstration is available at https://youtu.be/cHfs3HLzc2Y.

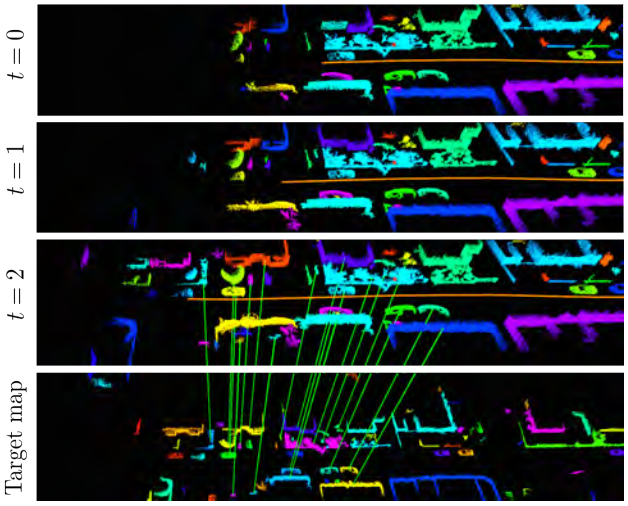


Figure 1: Example of localization made by our incremental approach in a urban driving scenario. The robot is driving from right to left and its trajectory is shown in orange. Segments are extracted, tracked and merged over successive observations. At $t = 2$ a localization is made against the target map with segment correspondences denoted by green lines.

However, fully processing such representation at each time-step leads to expensive redundant computations. Therefore, we propose to reuse the processed information by retaining data structures that are likely to save computations in later localization attempts.

The first module of the presented solution accumulates and filters a continuous stream of 3D point cloud data through a Dynamic Voxel Grid (DVG), providing information about newly occupied voxels. This key information is leveraged for estimating normals by re-computing only those affected by newly occupied voxels and by caching information to incrementally compute covariance matrices. Incremental region growing segmentation is then performed by using only the newly occupied voxels as seeds and by merging with previously clustered points. As illustrated in Fig. 1, this strategy enables to robustly track segments between successive observations which offers multiple benefits to the overall framework. Amongst others, it enables the final stage of our solution: an efficient recognition strategy based on partitioning and caching of geometric consistencies.

To the best of our knowledge, this is the first work to propose combining incremental solutions to normal estimation, segmentation, and recognition for finding global associations in 3D point clouds. The full solution is evaluated in real-world experiments demonstrating localization rates of up to 10Hz.

We also show that the efficiency of the presented method makes it suitable for real–time applications and enables its usage on cheaper, low–energy systems.

To summarize, this paper presents the following contributions:

- An incremental method for localization in 3D point clouds based on segment matching.
- A set of incremental algorithms for the normal estimation, segmentation, and recognition steps.
- An exhaustive comparison of the incremental approach with a batch solution through disaster response and urban driving experiments.

The remainder of the paper is structured as follows: Section II provides an overview of the related work, and Section III describes the proposed incremental approach. The incremental approach is compared to a batch solution in Section IV, and Section V finally concludes the work.

II. RELATED WORK

An overview of the related work in the field of localization in 3D point clouds was presented in our previous work [1]. In this section we review previously proposed methods related to the two core modules of our approach: efficient point cloud segmentation, and geometric verification.

a) Incremental point cloud segmentation: Closely related to our work, Whelan et al. [3] proposes an incremental region growing method for segmenting dense point cloud maps. Segmentation is done only once for each input cloud with a merging step afterwards. Only planar segments were considered whereas our generic region growing algorithm allows for different tuples of growing policies. Tateno et al. [4] merge RGB-D data into a *global segmentation map* that is maintained by matching and propagating segments extracted from the current depth map. Similarly, Finman et al. [5] propose to segment the depth maps using an incremental variation of the graph-based Felzenszwalb algorithm. A voting algorithm is proposed for recomputing parts of the segmented map given new data. None of the above works proposed a solution for retrieving models based on the generated segments. Contrastingly, we show through multiple experiments that our incremental region growing algorithm can effectively be leveraged for localization.

b) Efficient geometric verification: Strategies for reducing the number of correspondence pairs have been proposed for stereo images. Ayache and Faverjon [6] describe a partitioning scheme for efficiently finding neighbor segments in stereo images, while [7] performs RANSAC only on *spatially consistent* correspondences, i.e. correspondences that have a minimum fraction of matching neighbor features in both images. Both methods rely on assumptions about the disparity between images, thus their accuracy is influenced by the presence of high disparity and strong variation in viewing angles. In this work we present a method for performing localization efficiently through partitioning and caching, reducing the asymptotic complexity of the geometric consistency grouping method [8].

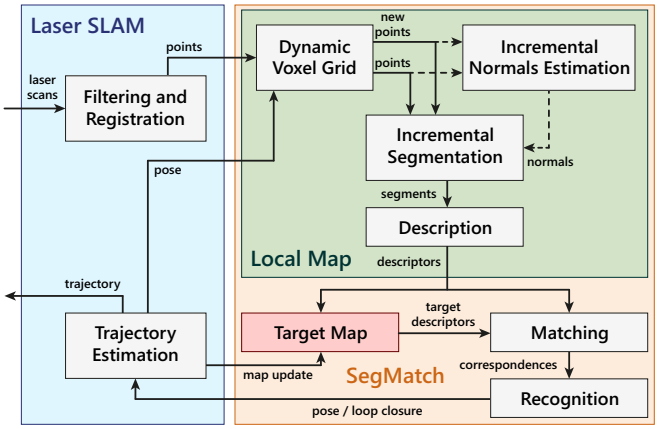


Figure 2: Architecture of the incremental segment matching pipeline and its interface with a laser SLAM framework[2].

III. METHOD

This section introduces our incremental solution to localization in 3D point clouds and the contributing modules. An overview of the approach is depicted in Fig. 2.

A. Dynamic Voxel Grid

The continuous input stream of 3D points is filtered and accumulated in a local cloud using a voxel grid approach. Instead of performing batch voxel filtering of the entire local cloud with each new measurement, we add and update only voxels that are affected by the new points. We implemented a DVG, an efficient data structure supporting dynamic insertion and removal of points. Occupied voxels are stored in a vector in increasing voxel index order. Each voxel maintain its index, centroid, and the number of points it contains. In order to reduce noise, a voxel is considered *active* if it contains at least a desired amount of points. For the successive stages of the pipeline, only active voxels are used.

1) Voxel Indexing: Virtually, the voxel grid is a regular grid of size $l \times w \times h$ voxels, where each voxel has a unique index in the interval $[0, l \cdot w \cdot h - 1]$. The grid has a fixed resolution r and a rigid transformation from the grid frame to the map frame T_{mg} which is initialized such that the robot starts at the center of the grid. Indexes are stored in b–bits unsigned integers, for computational efficiency we require the sizes of the grid to be powers of 2: $l = 2^{l_{bits}}$, $w = 2^{w_{bits}}$ and $h = 2^{h_{bits}}$, where $l_{bits} + w_{bits} + h_{bits} \leq b$. The voxel index $I(q)$ of a point q is computed as:

$$I(q) = \lfloor t.x \rfloor + \lfloor t.y \rfloor \ll l_{bits} + \lfloor t.z \rfloor \ll (l_{bits} + w_{bits}) \quad (1)$$

where $\ll$ is the bitwise left shift operator and t corresponds to the grid coordinates of the point q according to: $t = T_{mg}^{-1} \cdot q \cdot r^{-1}$.

[2]In this work we use a pose-graph SLAM system which performs registration between successive LiDAR scans using Iterative Closest Point (ICP) [2]. The implementation is available at https://github.com/ethz-asl/laser_slam.

2) Insertion and Removal: When new points are inserted, the DVG computes their voxel indices and sorts them in increasing voxel index order. Considering that sorting has an asymptotic complexity $O(n \log(n))$, this is an important optimization over sorting all the points as performed in batch voxelization. Once new points are sorted, they are added to the existing voxels in linear time with a merge operation: When m points q_i are inserted in a voxel with centroid p downsampled from n points, its properties are updated as:

$$p \leftarrow \left(n \cdot p + \sum_{i=1}^{m} q_i \right) \cdot \frac{1}{n+m}, \qquad n \leftarrow n+m \tag{2}$$

In addition, the indices of the voxels that turned active after the insertion are collected in a set U, so that the normal estimator and the segmenter can operate on the new voxels only. When the robot moves, the DVG is updated by removing voxels that fall outside the radius of the local map.

3) Rigid Transformation: When a loop closure is detected, the SLAM application re–evaluates the trajectory of the robot and provides a new estimation of its pose. Consequently, the local cloud must be updated with a rigid transform T which is applied to the centroid of each voxel. Moreover, in order for new points to be assigned to the correct voxel, the transformation of the DVG is updated as $T_{mg} \leftarrow T T_{mg}$.

B. Incremental normal and curvature estimation

The normal of a point p_i in a 3D point cloud is commonly estimated from the covariance matrix M of a neighborhood $\mathcal{N}(p_i)$ [9]. After finding the neighborhood by fixed–radius Nearest Neighbors (NN) search and arranging the neighbor points in a n_i-tuple $\{\nu_j\}_{j=1}^{n_i} := \mathcal{N}(p_i)$, M_i is computed as a sample covariance, using $\bar{\cdot}$ to denote the average operation, i.e. $\overline{\nu_j} := \frac{1}{|\nu|} \sum_{j=1}^{|\nu|} \nu_j$, omitting the index if unnecessary:

$$M_i := \overline{(\nu_j - \overline{\nu})(\nu_j - \overline{\nu})^\intercal}, \tag{3}$$

The estimate of the normal is equal to the normalized eigenvector of M_i corresponding to the smallest eigenvalue, while the curvature is computed as $\sigma = \lambda_0 (\lambda_0 + \lambda_1 + \lambda_2)^{-1}$ where $\lambda_0 < \lambda_1 < \lambda_2$ are the eigenvalues of M_i.

In this work, we apply two major optimizations to make the process incremental: The covariance matrix M_i is computed incrementally and only normals affected by new scanned points are updated. By expanding the factors in eq. (3), we obtain an incremental formulation:

$$M_i = \overline{\nu_j \nu_j^\intercal} - \overline{\nu}\,\overline{\nu}^\intercal = \frac{1}{n_i} \cdot A_i - \frac{1}{n_i^2} \cdot b_i b_i^\intercal \tag{4}$$

where A_i and b_i are respectively the accumulators for $\overline{\nu_j \nu_j^\intercal}$ and $\overline{\nu}\,\overline{\nu}^\intercal$. The advantage of this formulation over eq. (3) is that it can be computed incrementally, without the need to keep track of the neighborhood $\mathcal{N}(p_i)$ of each point. For reference, a similar formulation is introduced by Poppinga et al. [10] for performing efficient batch plane detection in 3D point cloud data.

1) Incremental Updates: The accumulators of each point are computed incrementally following a *contributions scattering and gathering* procedure. For each new point index $i \in U$ accumulators A_i, b_i and n_i are initialized with 0 and for each $p_j \in \mathcal{N}(p_i)$, including already the new points, contributions are scattered as:

$$A_j \leftarrow A_j + p_i p_i^\intercal, \quad b_j \leftarrow b_j + p_i, \quad n_j \leftarrow n_j + 1 \tag{5}$$

Similarly, contributions are gathered from old points only, i.e. if $j \notin U$:

$$A_i \leftarrow A_i + p_j p_j^\intercal, \quad b_i \leftarrow b_i + p_j, \quad n_i \leftarrow n_i + 1 \tag{6}$$

Finally, covariance matrices, normals, and curvatures are recomputed for points whose accumulators have been updated.

2) Rigid Transformation: In the event of loop closures, the trajectory of the robot is re–estimated and a rigid transformation is applied to the filtered point cloud C. When this happens, we transform the accumulators in order to avoid inconsistencies caused by the accumulation of points belonging to different reference frames.

The transformation, expressed as translation t after rotation R, is applied to all points p_i belonging to the local cloud as $p_i \leftarrow R p_i + t$. Let $\tilde{\nu} := (R\nu_j + t)_{j=1}^{n_i}$ be the transformed neighborhood tuple of the point p_i. Then the updated sample covariance (4) for $\tilde{\nu}$ can be computed using:

$$\overline{\tilde{\nu}_j \tilde{\nu}_j^\intercal} = R\,\overline{\nu_j \nu_j^\intercal} R^\intercal + R\,\overline{\nu} t^\intercal + t\,\overline{\nu}^\intercal R^\intercal + t t^\intercal \tag{7}$$

$$\overline{\tilde{\nu}} = \overline{R\nu_j + t} = R\,\overline{\nu} + t \tag{8}$$

Thus, the accumulators are updated as:

$$A_i \leftarrow R A_i R^\intercal + R b_i t^\intercal + t b_i^\intercal R^\intercal + n_i t t^\intercal, b_i \leftarrow R b_i + n_i t \tag{9}$$

The normals are updated as: $N_i \leftarrow R N_i$, while the point curvatures are not affected by the transformation.

C. Incremental region growing segmentation

This section presents our generic algorithm for incremental segmentation of 3D point clouds based on *region growing policies*. Given the low fraction of new points added with each measurement, we achieve an efficient segmentation by using only new points as seeds for growing regions.

Algorithm 1 shows the pseudocode for growing a region in a 3D point cloud starting from a seed point with index s. Growing is performed using the seeds contained in the ordered seeds list provided by the PREPARESEEDS policy. The growing strategy is controlled by the CANGROWTO and CANBESEED policies, which respectively determine if growing from a seed to a neighbor is allowed and if a point can be used as seed. The result is a *cluster* Γ with a unique *cluster ID* γ. New unclustered points initially have no cluster ID assigned. The NN function on line 5 finds the neighbors of a point by fixed–radius search. In order to reduce the number of k–d tree constructions, the same tree is shared with the normal estimator. Future work could gain efficiency over this

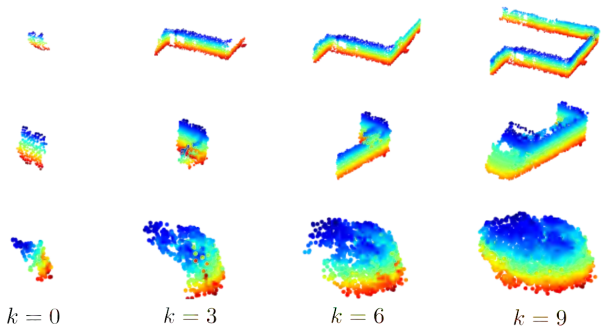


Figure 3: An illustration of three segments being incrementally grown over successive observations.

approach by exploiting the underlying structure of the DVG through organized region growing segmentation.

Once all points have been clustered, clusters that reached a minimum number of points are promoted to *segments* and obtain a unique *segment ID*. A mapping from cluster to segment IDs is maintained in order to enable *segment tracking* (section III-C3). An example of segments being grown over multiple observations is depicted in Fig. 3.

Algorithm 1 Incremental region growing given a starting seed s, a point cloud P and a new cluster ID γ.

```
1:  function GrowFromSeed(s, P, γ)
2:      Γ ← {s} , S ← {s} // Initialize cluster and set of seeds.
3:      while Seeds ≠ ∅ do
4:          s ← PopFront(S)
5:          for each n : NN(s, P) do
6:              if CanGrowTo(s, n) then
7:                  if HasClusterID(n) then
8:                      LinkClusters(s, n, γ)
9:                  else
10:                     SetClusterID(n, γ)
11:                     Γ ← Γ ∪ {n}
12:                     if CanBeSeed(n) then
13:                         S ← S ∪ {n}
14:     return Γ
15: end function
```

1) Clusters merging: While batch algorithms only need to cover the cluster growing case, the incremental version must also handle the cluster merging case (see Fig. 4). This is illustrated on line 8 where a point that has been previously assigned to another cluster is reached. In this case, the two clusters are merged and obtain the same cluster ID. If both clusters already have a valid segment ID (Fig. 4c), the minimum (i.e. the oldest) segment ID is used for the resulting set. In SLAM applications, this causes the segments to be merged in the target map as well.

2) Growing policies: In this work, we present two triples of policies for our incremental region growing algorithm. The *smoothness constraint policies* are derived from the work by Rabbani et al. [11]. During the preparation phase, PrepareSeeds collects the indices of the points that pass the CanBeSeed test and sorts them in increasing curvature order. This guarantees that regions are grown starting from the flattest points, reducing the number of segments created. CanGrowTo returns true if the normals of the seed and

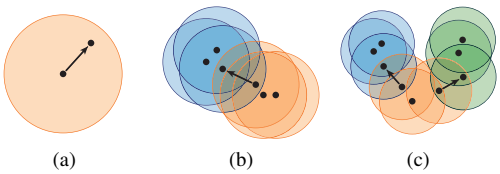


Figure 4: Region growing and merging examples using Euclidean distance policies in 2D. (a) An unclustered point is reached and added to the current cluster. (b) An existing cluster (blue) is reached and linked to the growing region (orange). (c) A current cluster (orange) reaches and is linked with two other clusters (blue and green).

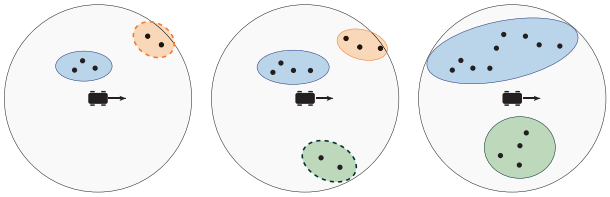


Figure 5: Evolution and tracking of segments as the robot moves. The black circle represents the boundary of the local map. In this example, the minimum size a cluster must have in order to be considered a segment is 3. (a) The robot observes a segment (blue) and a cluster (orange). (b) As the robot moves, more points are inserted in the local map. The orange cluster turns into a segment and another cluster appears. The blue segment grows but maintains the same ID. (c) From a different perspective, more points are observed, triggering the merge of the blue and orange segments.

neighbor points are close to parallel. Since the orientation of the normals is unknown, this is approximated by checking that the magnitude of the dot product between the two normals falls below a threshold. Another maximum threshold is applied on the point curvature in order for the CanBeSeed test to pass.

The *Euclidean distance policies* are straightforward as the incremental region growing algorithm already finds candidate neighbors based on their euclidean distance. Therefore, both CanGrowTo and CanBeSeed always return true and PrepareSeeds simply collects the indices of points that are not yet assigned to a cluster.

3) Segment Tracking: While cluster IDs are only temporary values used for identifying points belonging to the same clusters, segment IDs are lifetime–long identifiers of segments. The segmentation procedure presented in this section allows us to robustly track segments and their successive *views* in the local map which offers multiple benefits. In the previous work [1], multiple views could not be associated to the same segment and would obtain different IDs, causing the insertion of *segment duplicates* in the target map. Although heuristically identifiable by small distances between segment centroids, precise detection of such segments was not possible. Contrastingly, our method can robustly track segments and update them in the target map. Segment tracking also enables correspondences caching, which is needed by our incremental recognition approach (Section III-D). Moreover, having access to the complete history of the observations of each segment enabled the development of a method for learning segment descriptors that are more robust to changes in point of view [12]. Future work will explore different methods of leveraging the segment view history for improving segment matching performance.

D. Graph-based incremental recognition

Segments extracted from the local cloud are described with generic feature vectors (an eigenvalue-based descriptor [13] is used in the experiments of Section IV). Candidate *correspondences* between segments in the local and target maps are then found through NN searches in the feature space. A pair c_i, c_j of correspondences is called *geometrically consistent* if the difference of the Euclidean distance between the segment centroids in the local map and in the target map is less than a threshold ϵ, i.e. if

$$|d_l(c_i, c_j) - d_t(c_i, c_j)| \leq \epsilon, \tag{10}$$

where $d_l(c_i, c_j)$ and $d_t(c_i, c_j)$ are the distances between centroids in the local map and in the target map respectively. In our approach we formulate recognition as a graph problem with the goal of identifying a Maximum Pairwise Consistent Set (MPCS), which is a set of maximum size among all correspondence sets that are pairwise geometrically consistent.

Geometrical consistency relationships are encoded in a *consistency graph* $G = (V, E)$ where $V = \{c_i\}$ is the set of correspondences c_i and $E = \{e_{ij}\}$ is the set of undirected edges e_{ij} connecting all consistent pairs of correspondences (c_i, c_j). Identifying a maximum geometrically consistent set is then equivalent to finding a maximum clique of G.

We take advantage of the segment tracking feature described in section III-C3 which enables us to track correspondences as well. The number of consistency tests performed is reduced in an incremental fashion by reusing information computed in previous recognition steps. Since the insertion of new points in a segment changes its centroid, caching consistencies directly is not efficient. We rather propose to cache, for each correspondence c_i, a set of correspondences $\mathcal{S}(c_i) \subset V$ that are candidate to be consistent with c_i.

Based on eq. (10) we define the *consistency distance*, a measure for how far two correspondences c_i and c_j are from being consistent:

$$\Delta(c_i, c_j) = |d_l(c_i, c_j) - d_t(c_i, c_j)| \tag{11}$$

For each c_i, its *consistent candidates set* $\mathcal{S}(c_i)$ is then defined as the set of correspondences c_j whose consistency distance to c_i falls below a maximum threshold θ_Δ:

$$\mathcal{S}(c_i) = \{c_j \in V \mid j \leq i \wedge \Delta(c_i, c_j) \leq \theta_\Delta + \epsilon\} \tag{12}$$

where ϵ is the tolerance for consistency and the condition $j \leq i$ prevents duplicate entries of the same pair (caused by the symmetry of the consistency relation) from being stored.

1) Cache Maintenance: When a correspondence c_i is found for the first time, $\mathcal{S}(c_i)$ is computed and stored in the cache, together with the centroids of the local and target map segments. When a correspondence is not observed anymore, all references to it are removed from the cache. Furthermore, the consistent candidates set of a correspondence is invalidated if its two centroids move in total by more than $\frac{1}{2}\theta_\Delta$. The total movement is computed as the sum of the distances of each centroid to its cached position. This ensures that pairs of correspondences initially considered inconsistent are reconsidered if the combined movement of the segment centroids can cause them to be consistent. Correspondences, whose consistent candidate set has been invalidated, are reinserted in the cache as new correspondences. Fig. 6 shows an example of consistency cache maintenance.

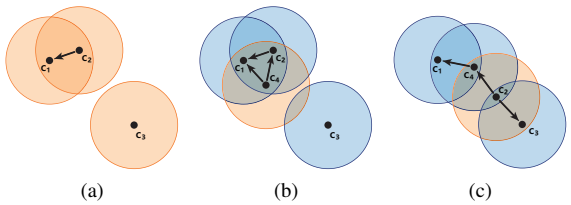


Figure 6: An example of consistency cache maintenance. Correspondences have arbitrary positions, while the distance between correspondences represents their consistency distance according to eq. (11). Arrows point from correspondences to their consistent candidates and circles represent the threshold for caching θ_Δ. (a) The correspondences c_1, c_2 and c_3 are inserted in the given order, and c_2 is found to be a candidate for consistency with c_1. (b) When c_4 is inserted, caching c_1 and c_2 as candidates for consistency. (c) The centroid of a segment of c_4 changes by a distance smaller than $\frac{1}{2}\theta_\Delta$, thus its cached information is still valid. c_2 changes by a distance greater than $\frac{1}{2}\theta_\Delta$, thus its consistent candidates are recomputed.

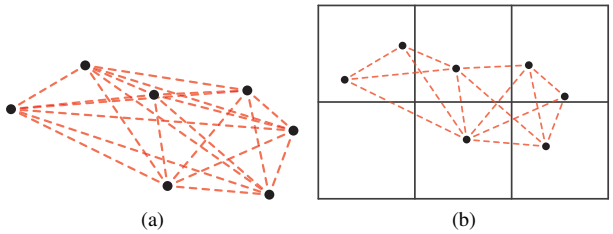


Figure 7: A consistency graph example where nodes and edges represent correspondences, plotted at their target map centroid's position, and tested pairs respectively. (a) Current approaches need to test all possible correspondence pairs for consistency. (b) Our partitioning approach allows to drastically reduce the number of consistency tests.

2) Consistent candidates set identification: In order to efficiently determine the consistent candidates set of a correspondence, we adopt the following partitioning approach. In order for two correspondences to be consistent, the distance between their target segments must be less or equal to the diameter of the local map. Thus we propose to prefilter candidates using a partition of the correspondences corresponding to the position of their target segment centroids in a regular grid with a cell edge length equal to the diameter of the local map. All the elements of $\mathcal{S}(c_i)$ can then be found among the correspondences whose target map centroid belongs to the same or to a neighbor partition as shown in Fig. 7. This process is described in detail in our workshop report [14].

3) Consistency Graph Construction: Consistencies involving new correspondences are identified while searching for their consistent candidates sets. For all other correspondences c_i, consistency tests only need to be performed for all the cached candidates, i.e. for $\{c_i\} \times \mathcal{S}(c_i)$.

4) MPCS Identification: We consider a recognition to be successful in case the size of the detected MPCS set is greater than or equal to a threshold parameter T. Thus we need to identify a maximum k–clique with $k \geq T$. Since the

consistency graph is sparse, we can rely on a particular class of algorithms [15] to find a maximum clique in linear time.

IV. EXPERIMENTS

In this section we evaluate the proposed incremental approach to localization in 3D point clouds. The performance of the system is first evaluated and compared to a baseline solution through three experiments. Finally, we present a challenging, large–scale, multi–robot, SLAM application which is made possible with the proposed approach.

A. Baseline

The baseline used for the comparison is the original *SegMatch* implementation [1] which is composed of standard Point Cloud Library (PCL) components. Specifically, batch voxel filtering is performed by the `pcl::VoxelGrid` while batch normals estimation is achieved with `pcl::NormalEstimation`. Batch segmentation is provided by `pcl::EuclideanClusterExtraction` and `pcl::RegionGrowing`. Recognition is finally performed using `pcl::GeometricConsistencyGrouping`.

B. Performance

The proposed solution and the baseline have been benchmarked in three different conditions:

- **KITTI Localization**: The vehicle drives in a known urban scenario, continuously localizing against a map generated from sequence 00 of the KITTI dataset [16].
- **KITTI SLAM**: The vehicle explores an unknown urban scenario, continuously updating a dynamic target map and trying to detect loop-closures (KITTI sequence 05).
- **Powerplant Localization**: A rescue robot drives in a known indoor scenario continuously localizing. The dataset has been recorded at the *Knepper* powerplant in Dortmund in the context of the TRADR project [2].

For both solutions, segments are extracted using Euclidean distance policies for the KITTI scenarios and with smoothness constraints policies the powerplant scenario. Segments are always described using eigenvalue-based features [13].The other parameters are detailed in Table I and have been found experimentally to yield good performance for both the baseline and proposed solutions. Note that coarser grid resolutions and higher minimum number of points results in fewer active voxels and faster segmentation rates. However, the resulting segments could be less descriptive of the actual scene objects.

1) Hardware: All experiments have been performed on a system equipped with 32GB of RAM and an Intel i7–6700K processor.The segment matching pipelines run in single–threaded mode and, for all experiments presented in this paper, the RAM usage of the whole system, including 3D mapping and trajectory estimation, never surpassed 1.6GB.

Parameter	KITTI	Powerplant
Local map radius (m)	50.0	25.0
DVG resolution r (m)	0.1	0.1
Min number of points for activating voxel	1	2
NN search radius for growing (m)	0.2	0.5
Max angle between normals for growing (degrees)	–	4.0
Max point curvature for using as seed (m^{-1})	–	0.05
Max distance for consistency ϵ (m)	0.4	0.4
Min MPCS set size T	5	6

Table I: Parametrization of the segment matching modules.

2) System performance: The timings and speedups resulting from these experiments are presented in Table II. Details are given separately for each module, to which an incremental solution is proposed, and for both the baseline and the proposed approaches. The category *others* includes segment description, matching, and, in the SLAM experiment, target map construction. Note that the batch method additionally requires 10ms for associating segments when updating the target map whereas this is obtained directly in the incremental solution. In event of loop closures, an average of 22.8ms more is required by both methods for updating the target map and the k–d tree used for matching segments.

In all experiments, the proposed localization approach can process the measurements faster than the update rate of the sensor. This is particularly interesting in the KITTI experiments where our approach allows to process in real-time the large data throughput of the Velodyne HDL-64E. The overall speedups achieved by the incremental approach over the batch solution are 8.9x, 7.1x, and 12.4x respectively for each scenario. The processing rates achievable by the incremental pipeline range between 13Hz and 25Hz depending on the experiment. However, in practice, these values are now limited by the sensor frequencies. Further detailed statistics about the experiments are summarized in Table III.

We observe that successful localizations are generally based on a redundant amount of correspondences. Since the recognition step automatically rejects inconsistent candidate correspondences, the algorithm is still able to localize even in moderately dynamic environments. As an example, the localization shown in Fig. 1 is based on 18 candidates. Since we require at least $T = 5$ correspondences, localization would still be successful even if the parked cars would move. Future work could further improve the robustness to dynamic objects by (1) leveraging semantic information that can be extracted from machine learning-based segment descriptors [12] and (2) simultaneously using multiple growing policies in order to generate more candidate segments.

3) Dead reckoning distances: The higher localization rates of the proposed method results in lower dead reckoning distances. This is illustrated in Fig. 8 which shows the probability of traveling a specific distance without successful localization in the map generated from KITTI sequence 00. See [1] for a definition of this metric. With the proposed incremental approach, localization happen within 1.5m more than 90% of the times, while the original approach can localize within the same distance less than 10% of the times, occasionally

Table II: Runtimes of the modules of the batch and incremental approaches (ms).

Module	KITTI Localization			KTTI SLAM			Powerplant Localization		
	Batch	Incremental	Speedup	Batch	Incremental	Speedup	Batch	Incremental	Speedup
Voxel filtering	56.9 ± 19.2	4.2 ± 2.8	x13.5	69.6 ± 35.1	4.11 ± 1.88	x16.9	32.7 ± 18.7	1.7 ± 0.8	x18.8
Normal estimation	-	-	-	-	-	-	166.4 ± 64.9	10.2 ± 1.6	x16.4
Segmentation	389.7 ± 147.8	40.7 ± 13.4	x9.6	395.1 ± 123.0	44.5 ± 15.2	x8.9	275.5 ± 112.5	22.4 ± 8.6	x12.3
Recognition	85.4 ± 39.7	6.0 ± 3.2	x14.2	41.3 ± 54.5	3.5 ± 4.15	x11.8	0.7 ± 0.5	0.1 ± 0.05	x7.4
Others	10.0 ± 3.6	10.2 ± 3.5	-	34.4 ± 19.7	23.9 ± 12.7	-	3.7 ± 1.7	4.4 ± 1.9	-
Total	542.1 ± 210.3	61.2 ± 23.0	x8.9	540.4 ± 232.3	76.0 ± 33.9	x7.1	479.1 ± 198.2	38.7 ± 13.0	x12.4

Quantity (per–step)	KITTI Localization	KITTI SLAM	Powerplant Localization
Sensor rate	10Hz	10Hz	0.33Hz
Created voxels	2.1k±0.6k	2.1k±0.7k	2.2k±0.3k
Local cloud size	156.5k±50k	159.5k±41k	76.5k±34k
Modified normals	-	-	10.6k±3.3k
Local map clusters	9.1k±3.9k	8.6k±2.1k	7.4k±2.4k
Local map segments	52.8 ± 13.7	60.1 ± 13.8	42.7 ± 20.9
Target map segments	1204	847 ± 423	83
Partitions	25	8.2 ± 3.9	1
Correspondences	3.1k±0.9k	1.6k±1.0	103 ± 55
Cached correspondences	2.8k±0.8k	1.5k±0.9k	95 ± 53
Cache invalidations	2.3 ± 8.4	1.8 ± 6.3	0.1 ± 0.3

Table III: Statistics (mean and standard deviation) characterizing the different experiments. Values refer to observations made in one localization step. It is interesting to note some strong differences between experiments (e.g. number cache invalidations) caused by the changes in scenario and configuration.

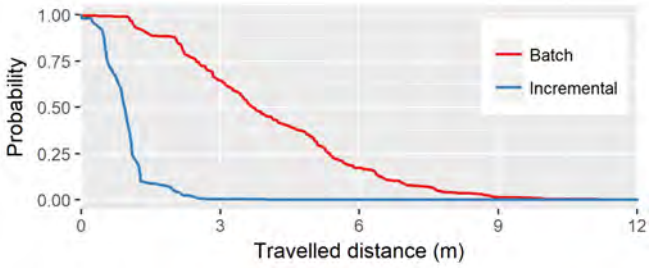


Figure 8: Probability of travelling a specific distance without a successful localization (data recorded during 5 runs of the KITTI localization example).

traveling more than 10m without localization.

4) Dynamic Voxel Grid: As described in Section III-A, voxel filtering requires a sorting step to group points belonging to the same voxel. This $O(n \log n)$ step has a significant impact when the local cloud contains a lot of points. With the incremental approach, this operation is reduced to sorting only the new points and then merging them with the stored sorted points in linear time. As shown in Table III, the new voxels represent only a small fraction of the entire local cloud, justifying the speedups observed in Table II. The timings stated for the voxel filtering include one pose update (removal of voxels outside the radius of the local cloud) and one insertion of the queued scans.

5) Incremental normals estimation: Normal estimation has been evaluated in the powerplant scenario only, as the Euclidean distance policies used in the KITTI examples do not require point normals. In this case, a speedup of 16.4x

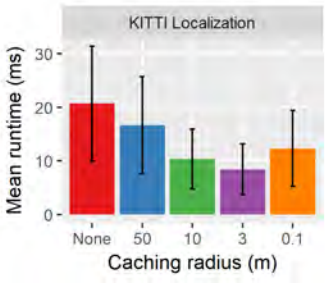


Figure 9: Mean runtime of the recognition stage with different caching radii. *None* indicates the pure partitioned approach without caching. In this dataset the ideal compromise between caching and invalidation is about 3 m.

is observed which is explained by the smaller number of NN searches required and by the caching of the covariance matrices. Furthermore, our implementation allows us to reuse the k–d tree built for segmentation which was not performed in the batch solution. In an equitable comparison where both estimators need to build a k–d tree of the local cloud, the incremental approach is in average 7.1 times faster.

6) Incremental region growing segmentation: Similarly, the most important improvement factor for the segmentation module is the reduction of the number of NN searches performed at every step. This is achieved by reusing stored information about the clusters in the cloud and only starting region growing from new unsegmented points. As stated in Table II, incremental segmentation achieves speedups over the batch methods of 9.6x, 8.9x and 12.3x.

7) Graph–based incremental recognition: Thanks to the partitioning scheme and the incremental caching, our recognition method reached speedups of 14.2x, 11.8x and 7.4x respectively. Moreover, the runtime of our method scales linearly with the number of correspondences, significantly improving over the cubic scaling of the batch algorithm (see Gollub et al. [14] for an asymptotic complexity analysis). Interestingly, cached correspondences represent $> 90\%$ of the total correspondences (Table. III), but are tested for consistency faster than new correspondences. In the KITTI localization experiment, the incremental recognizer performed on average 313k and 30k consistency tests on new and cached correspondences respectively. This is only 7.2% of the ~4.7 million tests performed by the batch solution at each recognition step. In the powerplant experiment, both recognizers can test for consistencies very quickly as the target map contains only a small number of segments.

The effect of different thresholds θ_Δ on the consistency distance for caching is compared Fig 9. Whereas high values of θ_Δ result in a lot of cached candidates, requiring a high

Figure 10: A top-down illustration of a common representation constructed in real-time on a single computer, by simulating five autonomous vehicles equipped with Velodyne HDL-64E sensors. The segments centroids are colored according to their associated vehicle trajectories.

number of consistencies tests, small values increase the number of invalidated cache entries. Therefore, the best setting for the radius is a trade-off between number of cached candidates and invalidation frequency.

C. Large-scale multi-robot experiment

This final experiment shows that the performance of the presented incremental approach enables us to address challenging SLAM scenarios. In order to simulate a multi-robot scenario, sequence 00 of the KITTI odometry dataset is split into five sequences which are simultaneously played back for a duration of 114 seconds. The data generated by five Velodyne HDL-64E sensors are processed in real-time on a single computer, in order to identify sufficient global associations to link the trajectories.

Although successful results were demonstrated in multi-robot scenarios with the batch approach [2], we found that it did not scale well to this higher number of vehicles. Specifically, its lower processing rate led to the extraction of too few segments, preventing the association of some trajectories. Contrastingly, our incremental approach successfully closed more than 100 loops which enabled to construct, in real-time, the common representation illustrated in Fig. 10. Similarly to the timings presented in Table II, 77.8ms$\pm$38.2ms were on average required to perform a localization step. This shows that the incremental approach effectively managed the higher number of voxels created at each step (5.2k vs 2.1k on average) which is caused by the delay when sequentially processing data from multiple sensors.

V. CONCLUSION

In this work, we presented a novel incremental approach for performing localization in 3D point clouds. We started by identifying the most computationally demanding operations in our previous pipeline. Then, efficient solutions were proposed for the individual sub-problems of the underlying segment extraction and matching technique. Unlike previous works, this approach maintains a segmented local map and performs geometry verification incrementally, reducing the computational burden and then allowing for more frequent localizations. The speed-up achieved allows for localizations at 10Hz, enabling real–time operation of 3D point cloud based SLAM systems. Our results indicate that the proposed approach could allow for seamlessly performing map-tracking, i.e. localization in a known map with a constrained search space based on the current position estimate. In the same direction it is worth to further investigate the application of our incremental recognition scheme to geometric verification for vision-based SLAM. Furthermore, whereas the present work considered eigen-based segment descriptors, it would be interesting to investigate incremental updates of learning-based descriptors that can potentially gain discriminative power and reliability over time.

REFERENCES

[1] R. Dubé, D. Dugas *et al.*, "Segmatch: Segment based place recognition in 3d point clouds," in *IEEE Int. Conf. on Robotics and Automation*, 2017.

[2] R. Dubé, A. Gawel *et al.*, "An online multi-robot slam system for 3d lidars," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2017.

[3] T. Whelan, L. Ma *et al.*, "Incremental and batch planar simplification of dense point cloud maps," *Robotics and Autonomous Systems*, vol. 69, 2015.

[4] K. Tateno, F. Tombari, and N. Navab, "Real-time and scalable incremental segmentation on dense slam," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2015.

[5] R. Finman, T. Whelan *et al.*, "Efficient incremental map segmentation in dense rgb-d maps," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[6] N. Ayache and B. Faverjon, "Efficient registration of stereo images by matching graph descriptions of edge segments," *International Journal of Computer Vision*, vol. 1, no. 2, 1987.

[7] T. Sattler, B. Leibe, and L. Kobbelt, "Scramsac: Improving ransac's efficiency with a spatial consistency filter," in *Computer vision, 2009 ieee 12th international conference on*, 2009.

[8] H. Chen and B. Bhanu, "3d free-form object recognition in range images using local surface patches," *Pattern Recognition Letters*, vol. 28, no. 10, 2007.

[9] H. Hoppe, T. DeRose *et al.*, *Surface reconstruction from unorganized points*. ACM, 1992, vol. 26, no. 2.

[10] J. Poppinga, N. Vaskevicius *et al.*, "Fast plane detection and polygonalization in noisy 3d range images," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2008.

[11] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 36, no. 5, 2006.

[12] A. Cramariuc, R. Dubé *et al.*, "Learning 3d segment descriptors for place recognition," in *LLM-IROS*, 2017.

[13] M. Weinmann, B. Jutzi, and C. Mallet, "Semantic 3d scene interpretation: a framework combining optimal neighborhood size selection with relevant features," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 2, no. 3, 2014.

[14] M. G. Gollub, R. Dubé *et al.*, "A partitioned approach for efficient graph–based place recognition," in *PPNIV-IROS*, 2017.

[15] D. Eppstein, M. Löffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," *Algorithms and computation*, 2010.

[16] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.