PoseMap: Lifelong, Multi-Environment 3D LiDAR Localization

Philipp Egger[1,2], Paulo V K Borges[1], Gavin Catt[1], Andreas Pfrunder[2], Roland Siegwart[2], Renaud Dubé[2]

Abstract— **Reliable long-term localization is key for robotic systems in dynamic environments. In this paper, we propose a novel approach for long-term localization using 3D LiDARs, coined *PoseMap*. In essence, we extract distinctive features from range measurements and bundle these into local views along with observation poses. The sensor's trajectory is then estimated in a sliding window fashion by matching current and old features and minimizing the distances in-between. The map representation facilitates finding a suitable set of old features, by selecting the closest local map(s) for matching. Similarly to a visibility analysis, this procedure provides a suitable set of features for localization but at a fraction of the computational cost. PoseMap also allows for updates and extensions of the map at any time by replacing and adding local maps when necessary. We evaluate our approach using two platforms both equipped with a 3D LiDAR and an IMU, demonstrating localization at 8 Hz and robustness to changes in the environment such as moving vehicles and changing vegetation. PoseMap was implemented on an autonomous vehicle allowing it to drive autonomously over a period of 18 months through a mix of industrial and unstructured off-road environments, covering more than 100 kms without a single localization failure.**

I. Introduction

Accurate and reliable localization in changing environments is key for autonomous navigation of robots and self-driving cars. Although Simultaneous Localization and Mapping (SLAM) is extremely useful in a number of scenarios, in many applications the goal is to perform map-based waypoint navigation, where a vehicle has to travel along predefined routes within a certain known area. Examples include autonomous vehicle navigation in cities, mining sites, ports, warehouses, agricultural lands, among others.

A number of sensors (and related algorithms) can provide an estimate of the current position of a vehicle in a map, each with their advantages and drawbacks. Methods using a Global Positioning System (GPS) and marker-based approaches have shown good results, but rely on external infrastructure. GPS, for example, only works in open outdoor environments and has limited precision to be used for vehicle control, while beacon-based approaches require prior installation. Therefore, systems based on local environment sensing are preferable in many scenarios. Cameras are generally affordable and provide great contextual information, leading to good localization results in structured environments (e.g. roads) [1]. However, cameras are sensitive to changes in lighting [1], [2] and often unsuitable for night-time operations.

[1] Robotics and Autonomous Systems Group, Data61, CSIRO, Australia.
[2] Autonomous Systems Lab, ETH Zurich.
Emails: philipp_egger@outlook.com, paulo.borges@csiro.au, gavin.catt@csiro.au, andrepfr@ethz.ch, rsiegwart@ethz.ch, rdube@ethz.ch

Among the many sensor options, LiDARs are one of the most popular due to their ability to provide precise range information and to be invariant to illumination changes. For these reasons, the localization solution proposed in this paper, coined *PoseMap*, focuses on LiDAR sensing.

As aforementioned, for many robotic applications it is desirable to localize and navigate repeatedly within a known area. In order to do so, the area first needs to be mapped. Then, for localization, we must find a set of map features that is visible at the robots location. These map features are used to estimate the current position of the robot in the map. Over time, however, the environment, tasks, and the area of operation may change or be expanded. Therefore, for robust long-term localization, the amount of changes between current and stored map data needs to be considered, and the map representation needs to be flexible and expandable.

To address the challenges above, instead of storing all detected map features in one global map, PoseMap consists of map feature bundles, each observed at unique locations. Every set of features is associated with its pose of observation. When performing localization, the closest map *nodes* are obtained. The features they hold, reflect what is visible at this location. Hence, the selection of features is of similar quality to the result of a visibility analysis with only the cost of searching the nearest map nodes. This procedure works due to the relatively large field of view (FOV) that LiDAR units generally have.

In order to make the localization more robust to changes we not only localize against the map but also build optimization constraints against the local history. This procedure allows for some differences between the map and the current real world without causing the localization to fail. When substantial change occurs, affected map nodes in the area are replaced automatically. Furthermore, the proposed algorithm continues localizing when the map is exited by performing local scan matching without additional map constraints. When the map is entered again, the loop is closed at run-time and the newly discovered area is instantly added to the map. These capbilities make PoseMap an extremely flexible solution for map-based localization, with experiments illustrating the applicability of the method on an autonomous unmanned ground vehicle (UGV). The vehicle has navigated without a single localization failure for more than 100 kms in very hybrid environments (ranging from built-up areas to off-road bushland) with significant slopes, using maps that are more than one year old. The navigation results considered changes in the environment (parking lots, buildings, vegetation) and online updating of the map when entering unexplored areas.

The remainder of this paper is structured as follows.

In Section II we discuss related work, contextualizing our approach within existing solutions. In Section III we describe the underlying SLAM algorithm used in PoseMap and how it can be used for localization in a map. Section IV presents PoseMap, the core contribution of this paper. In Section V we present experiments, with result from current and outdated maps. Relevant conclusions are drawn in Section VI.

II. Related Work

In this section, we provide an overview of LiDAR SLAM methods and contextualize PoseMap within SLAM, contrasting PoseMap with other localization methods.

In the past, LiDAR SLAM was commonly addressed in 2D with either an occupancy grid in combination with a Rao-Blackwellized particle filter [3] [4] or a pose-graph where all measurements are stored relative to their origin [5]. LiDAR SLAM in 3D has only in the last decade become of more interest due to its high computational cost. Bosse and Zlot presented a system that actuates a 2D LiDAR unit actively with a motor [6] or passively with a spring [7]. Their work was developed at CSIRO and hence we refer to it as C-SLAM in this paper. The trajectory is estimated continuously in a batch optimization process. A similar solution with additional high-frequency odometry has been proposed [8], in addition to other open source 2D/3D SLAM solutions such as Cartographer [9] and a multi-robot SLAM system for 3D LiDARs [10].

For long-term localization the fact that environments gradually (or suddenly) change over time needs to be considered, evaluating how much it impacts performance. In 2D SLAM this problem is often addressed with an *occupancy grid* [11], where the space is divided into cells, with each cell holding a probabilistic value for being occupied/free. Various adaptations of the original principle have been proposed such as modelling the occupancy with a two-state Markov process [12][13] or creating a 3D occupancy grid with octrees [14].

Sample-based approaches, in contrast, store and replace exact measurements instead of averaging them. Biber and Duckett [15] proposed a system for 2D SLAM that replaces random map measurements at a fixed rate. Walcott-Bryant et al. [16] proposed a system that keeps a history of past scans and activates and deactivates individual parts of the scan depending on whether new observations match previous views. More recently, Maddern et al. [17] showed that for successful localization it is not necessary to keep a fully accurate, up-to-date map. Instead they show successful 3D localization in cities using a set of past experiences simultaneously.

The method proposed in this paper, PoseMap, relies on a SLAM solution which needs to be able to provide a coherent (all loops closed) initial map that is accurate enough for waypoint localization. From the 3D methods mentioned above, we found C-SLAM [7] to be able to meet these requirements based on our experiments, hence we use C-SLAM as our base SLAM algorithm for PoseMap. Note, however, that the PoseMap concept can equally be applied to other 3D LiDAR-SLAM solutions as well.

The work we present is closest to the *sample-based* approaches. We do not keep actual measurements but instead store sets of features as they are much more compact than the raw point data. Similarly to dynamic pose graph SLAM [16], we keep a number of old views. However, we do not need multiple views of the same area, nor do we split the views up into sections. Furthermore, we do not try to keep track of highly-dynamic changes. Similarly to the work by Maddern et al. that leverages prior experience in cities [17], our map does not attempt to estimate the latest state of the environment. We argue that with a combination of constraints within our local history and against a map, we can handle a large amount of changes and localize over extended periods of time. We only try to update the map when substantial static changes occur. The PoseMap is designed for localization and not mapping. We do not aim to reflect the current surroundings in our map nor is the map “coherent”, meaning that map nodes next to each other can be updated at different times and reflect different states of the environment. These aspects are discussed thoroughly in Sections IV and V.

III. Background

This section provides background SLAM information to assist the reader in understanding the contributions of this paper.

A. C-SLAM

C-SLAM [6][7] works with a sliding window optimization approach. At the core of the algorithm lie surface elements termed *surfels*. These are planar features extracted from the point cloud. For every new iteration, all newly added laser data are projected into the map frame according to a motion prior generated from IMU and/or odometry data. Then the point cloud is split up into cubic bins, so called voxels. The voxels are not only built with spatial constraints but also split up according to time such that a new surfel is generated for each new 3D scan of the same area. The points in every bin are analyzed for their planarity p as

$$p = 2\frac{\lambda_2 - \lambda_1}{\lambda_1 + \lambda_2 + \lambda_3}, \qquad (1)$$

where $\lambda_1 \leq \lambda_2 \leq \lambda_3$ are the eigenvalues of the voxel's second order moment matrix. Surfels with similar position and orientation but different time of observation are matched with a k-Nearest Neighbors (k-NN) search in a *correspondence step*. Once the matching is finished the correspondences are used to build constraints for the *optimization step*. The optimization procedure calculates corrections of the trajectory estimation in order to minimize the distance of two similar surfels along their mean normal. This mean normal represents the eigenvector corresponding to the minimum eigenvalue of the sum of the matched surfels moment matrices. Deviations from the IMU measurements are penalized to ensure smoothness of the trajectory [6] [7].

Finally, continuity with the previous trajectory outside of the window is enforced through initial condition constraints. The optimization problem can be written as:

$$\begin{bmatrix} A_{match} \\ ----- \\ A_{smooth} \\ ----- \\ A_{initial} \end{bmatrix} \begin{bmatrix} \delta_r(\tau_1) \\ \delta_t(\tau_1) \\ \cdot \\ \cdot \\ \cdot \\ \delta_r(\tau_n) \\ \delta_t(\tau_n) \end{bmatrix} = \begin{bmatrix} b_{match} \\ ----- \\ b_{smooth} \\ ----- \\ b_{initial} \end{bmatrix} \tag{2}$$

where $\delta_r(\tau_n)$ and $\delta_t(\tau_n)$ are the frame corrections at time step τ_n. The A-b pairs are the linearized constraints. Match constraints are formed in continuous time and interpolated for the discrete optimization problem. The optimization problem is solved with iterative re-weighted least squares in an M-estimator framework. In order to minimize drift, the algorithm stores a set of m past surfel views, i.e. fixed views. A fixed view contains the surfels that were observed during a defined time-span which is no longer in the window. New fixed views are stored whenever significant translational or rotational movement occured since the last fixed view. For offline loop closure an optimization problem as in Eq. 2 is formed for the entire trajectory with a specific set of parameters.

B. Adjustments for Localization

Previous work [18] showed how replacing the fixed surfels with features from the map allows using C-SLAM for repeated localization. Hence, the match constraints in the optimization problem are derived from a combination of matches between local features, as well as matches between local and map features.

In this work, we also experimented to only match local features against map features. Due to the smaller optimization problem, a largely increased (more than 20 Hz) update rate was achieved. However, when combined with local constraints from within the sliding window, localization proved to be much more robust to changes in the environment. In experiments, the purely map based localization failed after around two months due to changing vegetation. Therefore we believe the latter, which was also used in [18], to be better applicable for most real world scenarios.

A further adaption of C-SLAM for localization was the introduction of a dynamic window shift size, as opposed to a static one used in [7] and [18]. Whenever an iteration has ended, a new one is started immediately with all new data available. This allows for localization at the maximum possible frequency of currently 4-10 Hz using both local and map features.

IV. PoseMap

The main contribution of our paper is a novel map representation which we call the *PoseMap*. It represents the environment as a set of map nodes that contain their own sets of 3D surfels. When localizing in the bounds of the map, the surfels of the closest map node(s) are used in the matching and optimization step of C-SLAM. When the map is left for a brief moment, the features in the newly discovered area are optimized to blend into the map and then added. In addition, map nodes are continuously evaluated for major changes. When an area has experienced a substantial and static change, the nearby nodes are automatically replaced with new data.

The approach builds on three main assumptions:

1) **The sensor setup has a very large field of view (FOV).** Otherwise, a more complicated method needs to be used to find a good feature overlap (potentially similar to visual slam solutions). Fortunately, a large FOV is characteristic to most of the LiDAR units currently available, assuming adequate positioning.
2) **The initial point cloud is of high accuracy and does not require further optimization.** This assumption is based on maps that are realistic representations of the world that are accurate enough to serve as a base map for localization [7].
3) **For localization it is not necessary to maintain a globally consistent map.** Instead, we propose a distributed map composed of submaps where each node can be assessed and updated separately. Two neighbouring positions in the map may consist of data taken at entirely different points in time.

A. Overlap Criteria

In various parts of the PoseMap algorithm, such as change detection and map filtering, we use a simple metric which we call the *overlap*. The metric is calculated from two sets of surfels: the test set S_T and the reference set S_R.

First, we propagate all surfels of S_R by position through an octree. The surfels of S_T are then added, and each surfel that falls into a bin which contains no surfel of S_R is marked as new. As a result, we have a new set of surfels S_N which is a subset of S_T, i.e. $S_N \in S_T$. This is illustrated with an example of change detection in an industrial environment in Fig. 1, where the metric is used to estimate the amount of change between the map and the local view.

Now let $c(S)$ be a function returning the number of surfels in a set of surfels S. Then we calculate

$$overlap = \frac{c(S_T) - c(S_N)}{c(S_T)} = \frac{c(S_T \setminus S_N)}{c(S_T)} \tag{3}$$

as the relative amount of "matched" surfels in S_T. Correspondingly, the amount of change can be reflected as

$$change = 1 - overlap = \frac{c(S_N)}{c(S_T)}. \tag{4}$$

It is important that a sensor setup with a similar FOV is used when the overlap metric serves for change detection. Otherwise, the additional area perceived with a new setup with increased FOV is marked as change too.

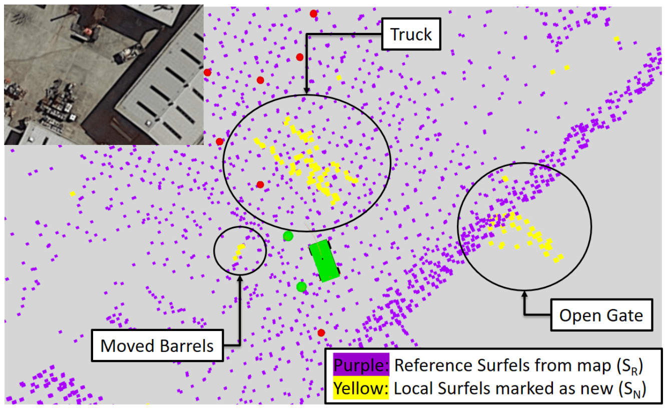

Truck
Open Gate
Moved Barrels
Purple: Reference Surfels from map (S_R)
Yellow: Local Surfels marked as new (S_N)

Fig. 1: Overlap/change metric illustrated on the basis of change detection in an industrial environment. Top left picture shows the area on Google Maps. Purple and yellow squares mark the centroids of surfels of S_R and S_N respectively. Hence the yellow surfels mark new surfels that had no surfel of S_R nearby. While most surfels of S_N have a obvious reason to be marked "new", the yellow surfels in the top right are simply due to insufficient map density in that area. Such noise artifacts influence the overlap/change by a total error of less than 5%.

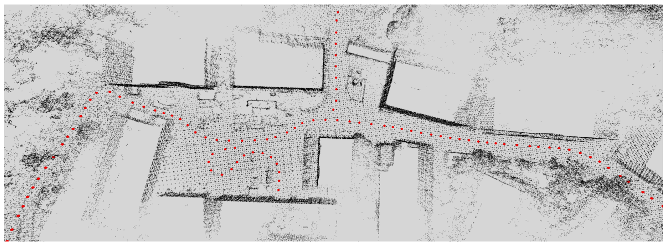
B. Initial Map Creation and Sparsification

Initially, we create a map of the area we want to localize using C-SLAM to provide the registered point cloud and trajectory. The data are then divided by time into small bins such that the full FOV is perceived (we used 1.5 seconds with our sensor setup.) Each bin is associated with its trajectory's mean position. Multiple measurements of the same area are filtered out such that only one surfel exists in each spot. This provides a set of submaps, each containing its mean position and a set of features visible at this place. Neighbouring submaps, however, still contain mostly overlapping data. Hence, we filter out poses that do not contain sufficiently unique surfels according to the overlap measure. We filter out all submaps that have an overlap with their closest neighbours that is higher than a threshold $T_{overlap}$. The filtering was implemented such that in each iteration the submap with the highest overlap, i.e. with the least unique data, is deleted and then the scores of affected neighbours are updated. This yields a very sparse, yet sufficient map where submaps in key positions, such as corners, are kept. This is clearly visible in the two map sectors in Fig. 2.

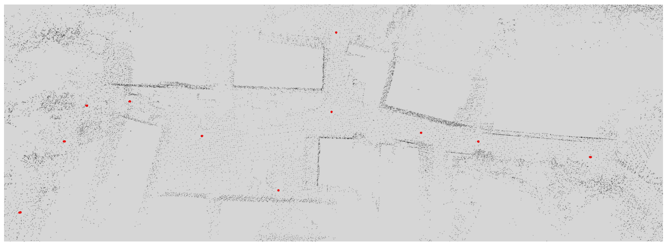
C. Localization

A major advantage of the proposed map representation is the simplicity and speed of map queries. When localizing, we simply use the surfels of the map's nearest neighbours to our current position. We generally use the closest two submaps which are in most cases on opposite sides of our ground vehicle. However, when they both lay on the same hemisphere, we have

$$\frac{\overrightarrow{PN_1} * \overrightarrow{PN_2}}{||\overrightarrow{PN_1}||\,||\overrightarrow{PN_2}||} > 0 \tag{5}$$



(a)



(b)

Fig. 2: Red points represent the PoseMap nodes. Black squares are surfel centroids. The top image shows map nodes filtered to a minimum distance of 3m. The bottom image shows filtering with $T_{overlap} = 0.8$. Note that the entire open area is solely represented by 2 nodes. Along the way nodes are kept at intersections and other points of interest.

with P being the current position and N_1,N_2 the two closest map nodes. In this case, we propose adding an additional map bundle into the correspondence search. This procedure provides the advantages of a computation-intensive visibility analysis at no cost. In our baseline approach [18], a k-NN search around the estimated position was proposed which is a more efficient alternative to a visibility analysis. However Fig. 3 nicely illustrates the advantages of the new method. The PoseMap offers a stronger assumption on what is visible, i.e. locally perceived features nicely overlap with the local map. Although the radius search in the baseline approach was limited to 50 m, it still provided 90% more features for matching. The PoseMap implementation not only reduces computational costs of the matching problem, but it also allows taking advantage of the full 100 m range of the LiDAR unit used. Our experiments (Section V-B) show that the increased range for localization improves robustness in open areas with little structure nearby.

D. Online Map Extension

To increase robustness and versatility of our system, we added online map extension capabilities. This allows the robot to go into a new area which has not been mapped yet. When the known map is left, motion estimation is continued through local matching. Upon re-entry, the loop is closed through a global optimization step where the PoseMap is used as fixed constraints and the extension is optimized to blend into the map. Map exit and entry can be identified either by the proximity to map nodes or by the overlap measure. The optimization is run in a separate thread in order not to interrupt the ongoing localization processing. The optimized data are then grouped, filtered and added to

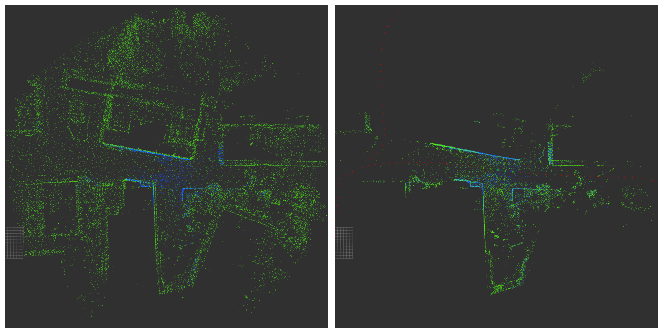


Fig. 3: Comparison of the surfels handed over by radius search (left) in a global map and by a PoseMap submap query (right). Green are the surfels from the map and blue are surfels in the current window. The PoseMap consists of 90% less surfels.

the PoseMap. This procedure is designed for small extensions for a new task around the known map. In order to make bigger extensions, the presented technique would need to be augmented with a global place recognition strategy [19], [20], [21].

E. Lifelong Localization

With our proposed approach, we do not aim to incorporate every change of the environment into the map. Our experiments indicate that, for localization purposes, the system is very robust to a significant amount of change without the need for map updates. We attribute the high robustness to three main factors:

1) Our sensing systems have an extremely large field of view. Hence they can perceive all static objects, such as buildings and trees, that are in the environment. We believe the concept would, for instance, not work in 2D LiDAR SLAM as there are cases where only dynamic elements are perceived, i.e. in a car park.
2) The combination of local and fixed constraints in the optimization increases reliability (see Section III-B). Map features of areas that changed are generally not matched at all in the correspondence search. However, these new features are still used for matching against the local past, earlier in the sliding window.
3) Surfels are rather generic features, averaging the underlying environment. We believe that this is superior over other features such as lines for long-term localization, particularly in natural environments.

Accordingly, updates are not necessary in daily scenarios with people and cars moving or a tree being removed. As presented in the experiments of Section V-C, we propose to only incorporate *substantial* static changes, such as new buildings or large adjustments of the terrain.

Whenever a local observation next to a map node has *overlap* smaller than a threshold T_{change} the environment is considered to have changed substantially. The local observation is then bundled to a map node replacement candidate and stored. When revisiting the same place again and the local surfels match now sufficiently well, this indicates that change was not of static nature and the replacement candidate is discarded. If the map, however, still does not reflect the present state, but the candidate does, the map node is replaced. This procedure can be tuned with the threshold T_{change}, a minimum time interval between evaluations and a the number of reconfirmations before node replacement. For practical operation, in our experiments, we used a minimum time interval of 12 hours and a single confirmation.

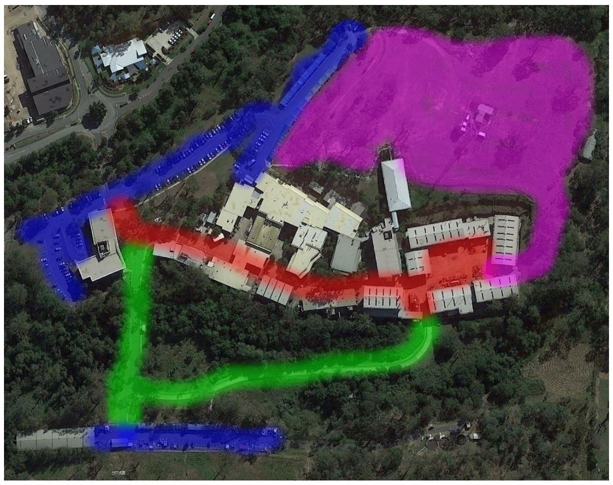


Fig. 4: QCAT, the test site, with coloured areas. red: buildings; blue: parking lot, green: dense trees and bushes, purple: off-road hilly area with open grassland and sparse trees.

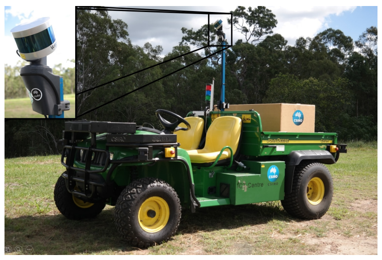
V. Experimental Results

To illustrate the applicability of the proposed method, we run experiments at the Queensland Centre of Advanced Technologies (QCAT) in Brisbane, Australia. The site features a rich mixture of static and dynamic elements, as well as bounded and open spaces. An aerial view of site is shown in Fig. 4, where the different areas are color coded. In the red area buildings and paved roads are predominant. Marked in blue is the parking area, which is constantly changing due to car movements. Green and purple mark non-manufactured, natural areas. While there is a paved road enclosed by trees and bushes in the green zone, the high hill (purple) is covered with open grassland and scattered trees.

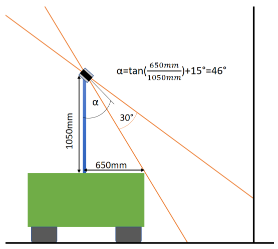
We mapped the 15 hectare test site with C-SLAM and built a PoseMap from the trajectory and point cloud. The PoseMap was filtered as explained in Sec. IV-B with $T_{overlap} = 0.6$, which we found to be a good level of filtering. At lower levels of $T_{overlap}$, gaps begin to appear between map nodes. The resulting map had a size of only 9.4 MB, which is arguably very small compared to the 5 GB point cloud from where the map was generated.

A. Hardware

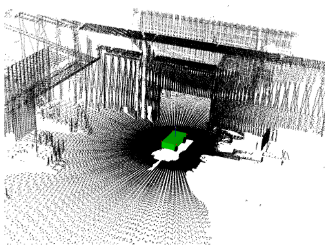
We implemented the localization algorithm on an electrical utility vehicle, i.e. a John Deere Gator TE, that was automated at CSIRO (Fig. 5a). The vehicle is equipped with a Microstrain GX3 IMU and a Velodyne PUCK VLP-16 LiDAR. Although the VLP-16 already features 16 beams, the generated data are rather sparse. We therefore designed a setup to continuously rotate the LiDAR on an electric motor



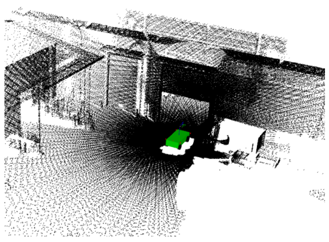
(a)



(b)



(c)



(d)

Fig. 5: (a) Picture of the fully automated John Deere Gator equipped with a Velodyne VLP-16 LiDAR. (b) Design considerations for the 45° sensor configuration. (c) Point cloud captured with a upright sensor mount. (d) A tilted sensor at 45° yields improved sampling due to alternating directions of the beams.

approximately 1 m above the vehicle in order to maximize the FOV. We initially mounted the sensor at an upright position but later redesigned the rotating platform for the LiDAR to be mounted at an angle of 45°. This way we maximize the amount of data along the horizontal plane and obtain fewer measurements from the sky or from the vehicle itself. The mounting is illustrated in Fig. 5b.

This angled configuration helps to increase robustness in open areas where there may be insufficient data on objects far away (the sensor has a range of 100 m). Furthermore, the beams sample the environment in different directions and therefore yield an improved sampling of the environment. This is visualized on the basis of point clouds in Fig. 5c and Fig. 5d, where the difference between an upright configuration and the 45° setup is clearly visible. We limit the amount of data coming from the sensor for online operation. We employ the same two filters proposed by Andreas et al. [18]. Firstly, a "ring" filter filters out all the data of some of the sensor's 16 beams. Figure 8 illustrates the effect of this filter on localization speed. In a second step, the octree filter then propagates the remaining data through an octree and deletes all but one random measurement in each voxel.

B. Autonomous Navigation

In order to evaluate the suitability of the proposed method, we performed a large number of autonomous test runs, using a waypoint navigation pipeline [18]. Fig. 6 shows an example trajectory that was driven fully autonomously with six predefined waypoints. We observe that the estimated trajectory is globally consistent as it precisely aligns with

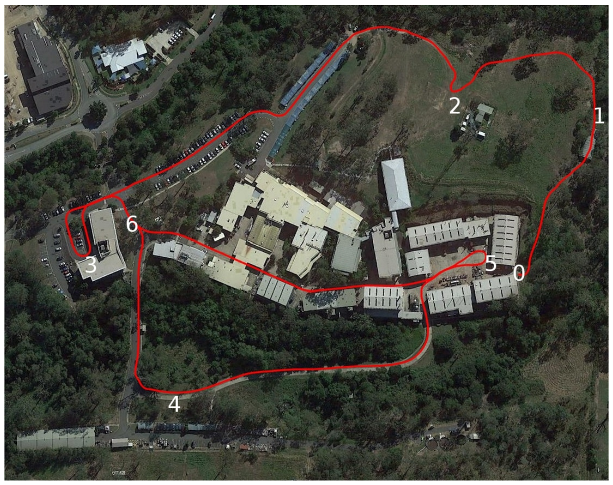


Fig. 6: Trajectory of an autonomous test run at QCAT overlaid on an image from google maps. The route was defined via six waypoints and spans a total length of 1.6 km.

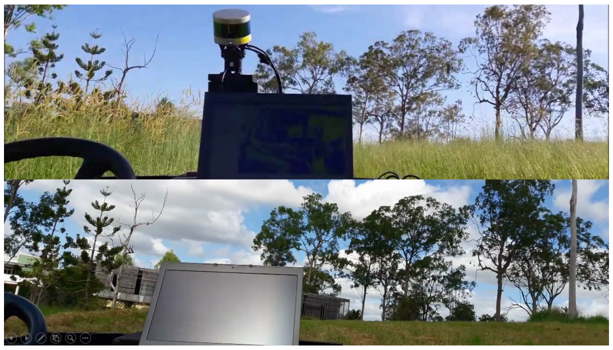


Fig. 7: Pictures taken approximately four months apart along the same track. The grass in the top pictures is approximately 1.5 m high and was freshly cut in the second photo. Localization and autonomous navigation were still working after the changes.

aerial images. A comparison against RTK GPS was already made by Pfrunder et al. [18] and is therefore not shown again in this paper. However, we managed to repeatedly navigate throughout all areas of the test site. The offroad area is especially demanding due to the vibrations and little structure. In this environment, localization was not yet possible with the baseline approach [18]. This is due to the fact that with the PoseMap we make use of the full 100 m range and have a more suitable set of features from the map. Furthermore, the 8x higher update rate reduces errors in the motion prior from IMU integration. A video of an autonomous test run is available online: https://youtu.be/B-WxDRWdIpY

C. Lifelong Localization

To this date, the localization technique presented has been tested for a period of over 18 months. Throughout this time, localization was always possible despite constant changes in the environment. Fig. 7 illustrates how much the environment can change without affecting the localization. Throughout the period of testing no level of change occurred

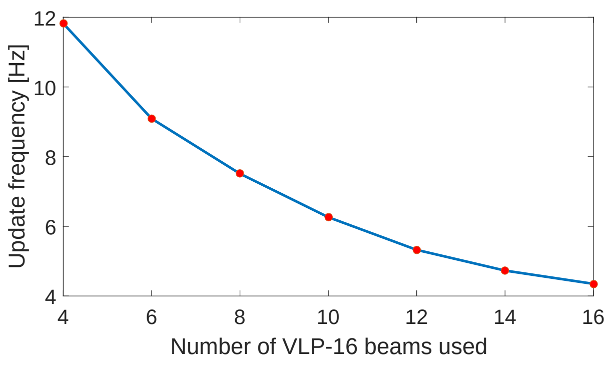


Fig. 8: Average update rate calculated on the same dataset for varying numbers of beams. Numbers were computed on a customized Dell Precision M4800 with a Intel Core i7-4910MQ processor.

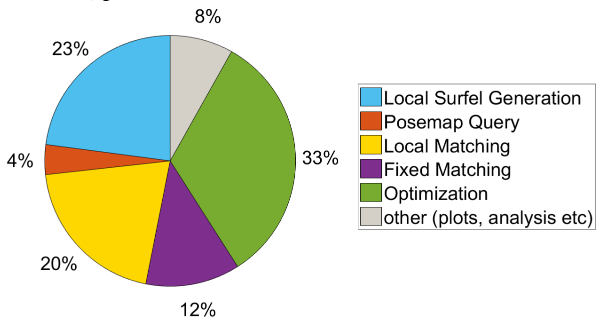


Fig. 9: Average computation time of different software components used during an autonomous run at QCAT.

that would compromise the localization and therefore the map update procedures were not necessary. Nevertheless the update procedures are still important to maintain successful operation in the case of any major change. We implemented and tested map updates with an overly sensitive threshold in the car park area. While the map was built during the day when the car park was full, we ran various test runs at night with no cars and a threshold $T_{overlap} = 0.85$ which was high enough to replace some nodes that were fully surrounded by cars. This was only for testing, as we generally do not want to adapt to dynamic elements as these. For long-term operation we now use $T_{overlap} = 0.6$ which was not yet reached in all our testing.

D. *Performance*

With the dynamic window shift size (Sec. III-B) the update rate of the algorithm is not fixed but mostly depends on the amount of data to be processed. The correlation to raw range measurements is clearly visibly in Fig. 8, which depicts the update frequency depending on the number of LiDAR beams being considered. Empirically, we found that 6-8 beams yields the best compromise between robustness and performance at an average localization rate of around 8 Hz.

However, performance also largely depends on the number of surfels. The amount of local surfels depends on the environment. In open spaces we have more surfels than in confined areas, hence the localization is slower. With 8 beams the update rate varied between 4 to 10 Hz. The amount of map surfels depends on how the set of features is selected in the map. Compared with our baseline system [18], which ran at a fixed update rate of 1 Hz, we were able to significantly increase performance due to the PoseMap's improved selection of map surfels compared to the radius search (Sec. IV-C).

Fig. 8 illustrates how, on average, only around 4% of the computation time is used by the PoseMap. This includes all PoseMap functionality except loop closure. With only matching against fixed surfels we were able to increase the update rate by a factor of approximately four. In that case fewer local surfels were generated, which reduces the computation time in all sectors but the PoseMap query. However, as described in Sec. III-B, the system became significantly more sensitive to changes.

E. *Urban Driving*

In addition to the experiments using the UGV Gator platform, we also tested the algorithm's suitability for urban driving. For this purpose we built a new fixed sensor setup with no moving parts outside the LiDAR unit (Fig. 10c), using only the vehicle's movement in order to scan the environment. We mapped a suburban neighbourhood (Indooroopilly) in Brisbane, Australia. In an initial mapping run, a total distance of 6.1 km was driven (Fig. 10b) and a

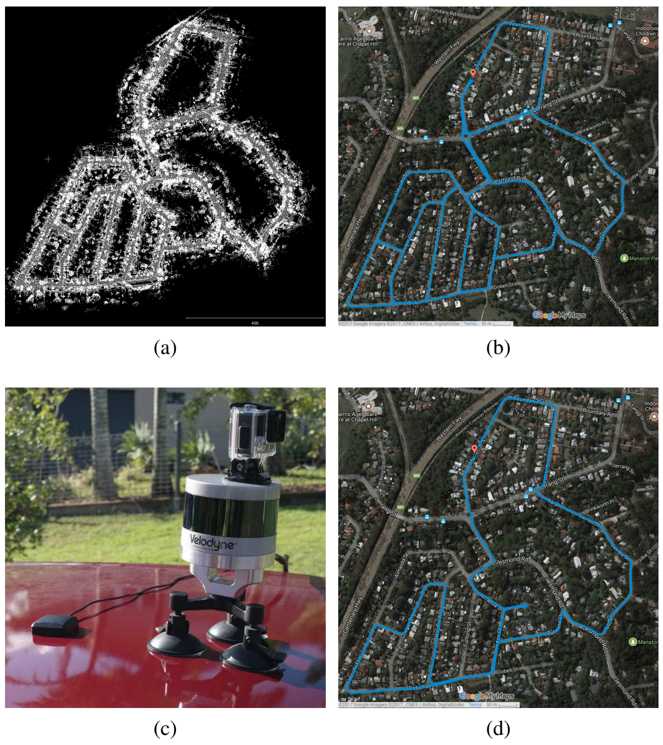


Fig. 10: (a) point cloud of initial map (b) georeferenced trajectory of initial map overlaid in Google MyMaps (6.1 km) (c) fixed sensor setup with GPS antenna and GoPro for documentation. IMU unit is embedded in the aluminium disk below the LiDAR unit (d) trajectory of a localization run (3.7 km).

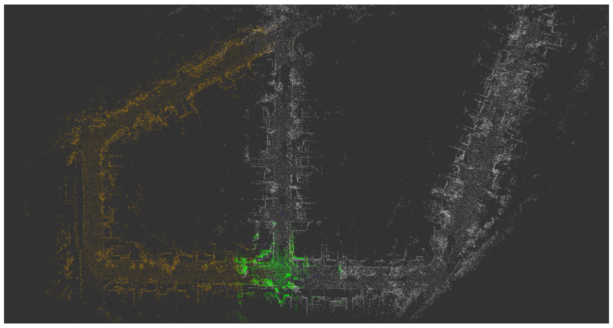


Fig. 11: Map extension in the street. The extension spans over a distance of 480 m. White points are used for surfel centroids of the map and orange for the extension. Green points are the local surfels of the localization.

point cloud (Fig. 10a) was generated using additional data from a GPS module. This was necessary due to the smaller FOV of the sensor configuration. From the point cloud the PoseMap was extracted with only 10.3 MB in size. The map was then used to localize in this area on multiple occasions at regular speeds of 40-60 kmh. All localization runs were successful with no GPS used. Tests were made at different times of the day including nighttime and times with increased traffic. Fig. 10d illustrates the estimated trajectory of such a localization run. The corresponding video is available online: https://youtu.be/KSxuxDnfiko

F. Map Extension

Map extensions were successfully tested both at QCAT and in the urban driving experiment. In Fig. 11 an extension in Indooroopilly is shown. The proposed technique is suitable for extensions up to about 1 km in length. For larger extensions an additional place recognition algorithm would be necessary. It is important to point out, however, that the online extensions were not designed to map large new areas, but instead to increase flexibility within and around the mapped area. In case localization is required in a new area, it should first be mapped in a designated mapping run.

VI. Conclusions

This paper presented a technique for long-term 3D localization in a mapped environment enabling autonomous agents to navigate in order to fulfill their tasks. The PoseMap consists of local views of features which are used for localization and can be updated or extended at run-time. We showed successful localization in a variety of environments spanning from grasslands to industrial buildings. During a test period of 18 months the system functioned without a single localization failure despite various changes in the test area. In comparison to the considered baseline, our approach led to 8 times higher localization rate and allowed localization in even more challenging terrain. Our approach additionally features map update and extension capabilities at runtime and is currently in use on some of CSIRO's drones and hexapods, illustrating the flexibility of the method. In future work, we would like to further increase the robustness of our method by detecting and filtering out dynamic objects, keeping only static elements. This can potentially allow for localization only against map surfels at increased speed (Section III-B).

References

[1] C. McManus, B. Upcroft, and P. Newman, “Learning place-dependant features for long-term vision-based localisation,” *Autonomous Robots*, vol. 39, no. 3, pp. 363–387, 2015.

[2] P. Mühlfellner, M. Bürki, M. Bosse, W. Derendarz, R. Philippsen, and P. Furgale, “Summary maps for lifelong visual localization,” *Journal of Field Robotics*, vol. 33, no. 5, pp. 561–590, 2016.

[3] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[4] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable slam system with full 3d motion estimation,” in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 155–160.

[5] M. Bosse and R. Zlot, “Map matching and data association for large-scale two-dimensional laser scan-based slam,” *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 667–691, 2008.

[6] M. Bosse and R. Zlot, “Continuous 3d scan-matching with a spinning 2d laser,” in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4312–4319.

[7] M. Bosse, R. Zlot, and P. Flick, “Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping,” *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[8] J. Zhang and S. Singh, “Loam: Lidar odometry and mapping in real-time.” in *Robotics: Science and Systems*, vol. 2, 2014.

[9] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-time loop closure in 2d lidar slam,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1271–1278.

[10] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, “An online multi-robot slam system for 3d lidars,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[11] A. Elfes, “Using occupancy grids for mobile robot perception and navigation,” *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[12] D. Meyer-Delius, M. Beinhofer, and W. Burgard, “Occupancy grid models for robot mapping in changing environments.” in *AAAI*, 2012.

[13] J. Saarinen, H. Andreasson, and A. J. Lilienthal, “Independent markov chain occupancy grid maps for representation of dynamic environment,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 3489–3495.

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

[15] P. Biber and T. Duckett, “Experimental analysis of sample-based maps for long-term slam,” *The International Journal of Robotics Research*, vol. 28, no. 1, pp. 20–33, 2009.

[16] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph slam: Long-term mapping in low dynamic environments,” in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1871–1878.

[17] W. Maddern, G. Pascoe, and P. Newman, “Leveraging experience for large-scale lidar localisation in changing cities,” in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 1684–1691.

[18] A. Pfrunder, P. V. K. Borges, A. R. Romero, G. Catt, and A. Elfes, “Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2601–2608.

[19] M. Bosse and R. Zlot, “Place recognition using keypoint voting in large 3D lidar datasets,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

[20] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, “SegMatch: Segment based place recognition in 3D point clouds,” in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5266–5272.

[21] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, “SegMap: 3d segment mapping using data-driven descriptors,” in *Robotics: Science and Systems (RSS)*, 2018.