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

Real-Time Autonomous Ground Vehicle Navigation in Heterogeneous Environments Using a 3D LiDAR

Andreas Pfrunder[1,2], Paulo V K Borges[1], Adrian R Romero[1], Gavin Catt[1], Alberto Elfes[1]

***Abstract*— The ability to drive autonomously in heterogeneous environments without GPS, pattern identification (e.g. road following), or artificial landmarks is key to field robotics. To address this challenge, we present a complete waypoint navigation framework for unmanned ground vehicles. A Velodyne PUCK VLP-16 LiDAR and an IMU are mounted on an autonomous, full size utility vehicle and used for localization within a previously created base map. We redesign a six degrees of freedom LiDAR SLAM algorithm to achieve 3D localization on the base map, as well as real-time vehicle navigation. We fuse the low-frequency, high precision SLAM updates with high-frequency, odometric local state estimates from the vehicle. The navigation costmap consists of a 2D occupancy grid which is computed from the 3D base map. Relying on this setup, the vehicle is capable of navigating through a complex site completely autonomously. The test site has densely and sparsely built areas, bushland, industrial activities, pedestrians, and other manned or unmanned vehicles. Extensive testing was done using both current and outdated base maps for comparisons, and a high precision RTK-GPS was used for ground truth. So far, more than 60 km of completely autonomous driving has been performed without a single system or navigation failure.**

Fig. 1. Automated John Deere TE Gator, an electric, full size utility vehicle used for all experiments in this paper. The Velodyne PUCK is marked with a red circle, the 2D Hokuyo LiDARs with blue circles and one of the rear wheel encoders with a purple dot.

I. Introduction

One of key challenges in robotics is unmanned ground vehicle (UGV) navigation in outdoor environments without dedicated infrastructure. Although much effort is currently being put into autonomous cars [1], [2], UGVs also find application in industrial and warehouse automation, mining and agriculture. In this paper, we present a novel navigation pipeline for UGVs discussing solutions for the localization and navigation modules. Full 3D localization is done by adapting an existing 3D SLAM algorithm [3], which was developed at CSIRO[1]. For clarity, we refer to this algorithm as CSIRO SLAM (C-SLAM) throughout this paper. Similarly, the localization framework that we present is referred to as C-LOC. Based on 3D LiDAR and IMU, C-LOC is very reliable and works in real-time. In C-LOC, we propose a pyramidal multi-level surface element (surfel) map which is used for accurate localization with update rates of approximately 1 Hz in our implementation. Since this update rate is not sufficient for vehicle control, we combine the C-LOC output with odometric information for control in-between C-LOC updates. Moreover, the odometric information is used to overcome a small delay originating from the C-SLAM algorithm. Relying on the real-time 3D pose of the vehicle and a 2D occupancy grid for path planning, we perform autonomous waypoint navigation. With a simple user interface the vehicle can be commanded to drive anywhere in QCAT, a CSIRO site in Brisbane, Australia. For additional safety, 2D LiDARs are placed on each corner of the vehicle to assist in avoiding collisions in any direction.

The algorithms are implemented on a John Deere TE Gator (Fig. 1), a full size utility vehicle which was automated by our team at CSIRO. The Gator is able to perform completely autonomous waypoint navigation while avoiding obstacles in the heterogeneous environment of QCAT. Our experiments consist of dozens of runs of more than 1 km each, totalling approximately 60 km of autonomous driving under normal site operations, with other vehicles, forklifts, pedestrians, courier delivery trucks, etc. The heterogeneous test area consists of a constantly changing parking lot, narrow and wide passages between high buildings, open areas and bushland regions. Fig. 7 shows the area, with waypoints and one of the driving paths overlaid in red.

The C-SLAM algorithm was first presented by Bosse and Zlot [3], followed by further improvements and application domains [4], [5], [6]. In this paper, we adapt a more recent version using a nodding 3D Velodyne PUCK LiDAR instead of the 2D Hokuyo LiDAR version [4]. The mapping output of C-SLAM can be further improved with place recognition [7], [8], which identifies similar physical regions of the environment in laser scans collected at different times. Hence, place recognition is used to identify loop closure in single datasets or merge multiple datasets together.

The six degrees of freedom (6 DOF) LiDAR C-SLAM

[1]Robotics and Autonomous Systems Group, Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia.
[2]ETH Zurich, Switzerland.

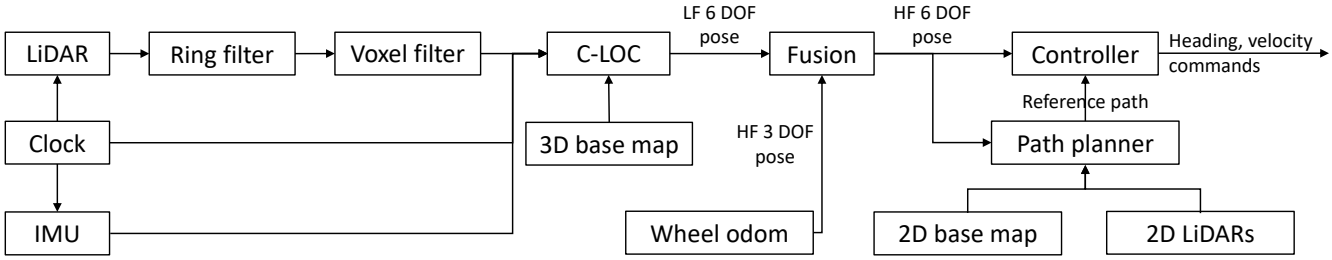


Fig. 2. Overview block diagram of all the relevant parts of this project to perform autonomous waypoint navigation. (Key. LF: low frequency; HF: high frequency)

algorithm is able to create accurate 3D point cloud maps and a commercial version for mapping is available[1]. Alternatives include LOAM (Lidar Odometry and Mapping in Real-time) [9] and an enhanced version with visual odometry [10]. In contrast to C-SLAM, LOAM considers corrections in the trajectory rather than to the trajectory in the state. Moreover, LOAM does not perform loop closure.

Several works show accurate 3D SLAM capabilities using aerial vehicles [11] or ground vehicles [12], [13], although they also do not discuss autonomous navigation. Another approach [14] uses a UGV to navigate its environment based on a real-time available 3D point cloud which is used to create an occupancy grid for the path planner. From a ground station this vehicle is tele-operated using the real-time available point cloud.

Regarding operation, a similar UGV to the one presented in this paper is SMART, a driverless golf cart [15] which however relies on curb lines for navigation.

One of the key aspects of the system presented here is that it does not rely on GPS, dedicated infrastructure, specific environment models (e.g. curbs), lane following or object recognition. It only requires a single scan of the environment (for map creation) and is resilient to significant changes in the base map, as discussed later.

The remainder of this paper is structured as follows. In Section II we first give an overview of the entire navigation framework (II-A), followed by a description of the underlying C-SLAM algorithm (II-B). Subsequently, in Sections II-C and II-D we describe the technical contributions that form C-LOC. A list of the main hardware components (III-A) and additional software implementations required to achieve full autonomous waypoint navigation (III-B) are presented in Section III. In Section IV we describe experiments, with results from current and outdated maps which are evaluated against ground truth from an RTK GPS. Final conclusions are drawn in Section V.

II. Methodology

Our goal is to perform completely autonomous waypoint navigation in a heterogeneous industrial/natural environment setting without GPS and using only inertial and LiDAR information. In this section, we present a summary of the navigation pipeline followed by a detailed description of each module.

A. System Overview

We initially build a 3D base map for localization purposes and from this map we create a 2D occupancy grid for navigation. The 3D base map is accurate (centimeter precision) as it is built offline with drift removed using global registration. In a navigation scenario, though, the map used for localization does not necessarily have to be extremely accurate. Experiments conducted over multiple months have proven that good localization results are obtained using C-LOC even in changing environments such as a parking lot with moving cars or a bushland area which is subject to vegetation change. The localization algorithm C-LOC outputs a 6 DOF pose at a low frequency around 1 Hz with a delay behind real-time due to data buffering [3]. This makes the algorithm unsuitable for real-time vehicle control, which requires a faster update rate.

To overcome this limitation, we fuse this delayed, low frequency 6 DOF pose with any kind of high frequency odometry information, obtaining a real-time pose at a high frequency, which is suitable for vehicle control. The controller then compares the robot pose with a reference path, taking into account a 2D occupancy grid of the environment and the obstacles detected by the 2D LiDARs. Using the output of the controller, a heading and a linear velocity command, the UGV can perform precise, fully autonomous waypoint navigation on all roads of QCAT. An overview of the full navigation pipeline is shown in Fig. 2.

B. C-SLAM Overview

The 3D C-SLAM algorithm uses an iterative non-linear least squares approach referred to as sweep-matching. Consecutive sweeps over the environment are compared to each other by first (i) identifying corresponding surfels from the LiDAR point cloud and secondly (ii) updating the trajectory such that the errors between matching surfels and the deviation from the measured IMU accelerations and rotational velocities are minimized. These two steps (the correspondence and the optimization step) are repeated until convergence is achieved. Surfels are calculated by statistically analyzing points that are both spatially and temporally close together. They contain information about the average, the variance and the surface normal of the ellipsoid of best fit.

[1] https://geoslam.com

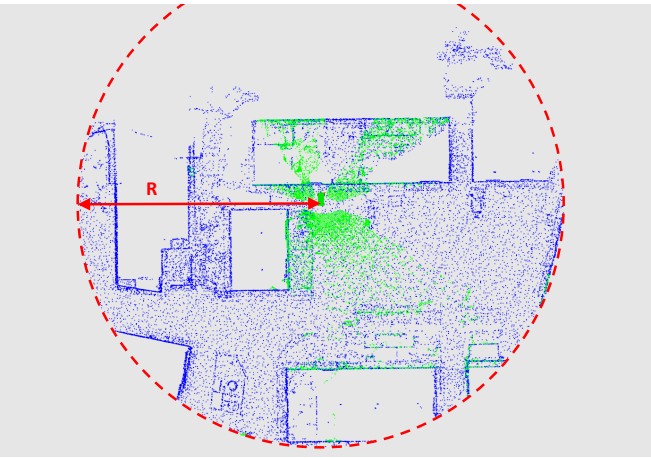


Fig. 3. For localization, only surfels within a 50 m radius (R) are used. Fixed surfels can be seen in blue, local surfels in green and the localizing UGV is shown with a green box.

In the correspondence step, surfel matches are obtained by approximate k-nearest neighbor search of a kd-tree in the 6D space of surfel positions and normals. For every matched pair a match error is computed which is then minimized in the optimization step. During the optimization step the surface correspondence error, the IMU measurement deviations and the initial condition constraints are minimized by adjusting the current trajectory estimate. The non-linear set of equations are linearized about the current best estimate by taking a first-order Taylor expansion and then solved by iterative re-weighted least squares.

Apart from open-loop SLAM, the C-SLAM algorithm [4] can also be used to produce a closed-loop trajectory by applying global point cloud registration, given an initial guess for the trajectory. This initial trajectory guess can be the open-loop solution explained previously. Generally, drifts larger than 10 m in the open-loop trajectory cannot be corrected by global registration. In this case, we use a place recognition algorithm [8] as an identification step before running global registration.

Both in C-SLAM (to create the base map) and the on-line C-LOC (used for real-time vehicle localization), new incoming data are processed in a sliding window fashion: A time-windowed segment of the trajectory (currently 3 s) is processed, then the window is advanced by a fraction of its length (currently 1 s). Each trajectory segment is processed by solving a linearized system of equations to obtain the trajectory which best explains corresponding surfels in the associated 3D point cloud while taking into account the boundary conditions which ensure continuity with the previous segment.

It is important to mention that the C-SLAM algorithm was never used for fast localization but only for accurate mapping, hence one of the contributions of this paper is adapting it for real-time localization.

C. C-LOC Overview

The C-LOC algorithm is, as previously mentioned, an adaptation of C-SLAM which allows for real-time full 3D localization within a prior built base map. C-SLAM keeps a number of surfels from recent past views, so called fixed-views. In order to limit the computational complexity from generating and processing those additional fixed surfel matches, a rolling window approach is used where only recent past views are considered. Those fixed surfels are additionally used to minimize the match error and help reduce the accumulated drift.

However, our goal in this work is to perform autonomous waypoint navigation based on a previously created map. Therefore, we eliminate the process of keeping surfels from recent past views as done in C-SLAM. Instead, for C-LOC, we consider a subset of all the fixed surfels of the previously created map. This C-LOC map is created in the following steps: first (i) from the point cloud of the environment and the associated trajectory an initial map is created that contains surfels based on not only on position but also viewpoint. This is important when localizing against small objects which can be seen from opposite sides such as lamp posts or branches. In case surfels were created only based on position, localization would be possible only from the side where the normals point in the same direction, since localization relies on the surfel normals to agree. In a second step, (ii) we voxelize this initial surfel map using the Point Cloud Library (PCL) [16] octree filter where we randomly select one surfel per voxel to ensure having surfels from different viewpoints. We create the surfels in a pyramidal fashion with voxel resolution of (0.4 m, 0.8 m, 1.6 m, 3.2 m) which is key for reliable localization. Hence, the result is a multi-level surfel map which is well suited for real-time vehicle localization. When running the online C-LOC algorithm, we use only the fixed surfels within a radius of 50 m around the position of the LiDAR (see Fig. 3) as this is sufficient for efficient localization and limits the computational complexity.

D. Odometry Fusion

In C-SLAM and C-LOC, the optimization problem is solved in a sliding window fashion which buffers data and causes the 6 DOF pose to be delayed [3]. Every time the optimization is finished, a series of poses is output. The minimum delay a pose can have, consists of the time required to solve the optimization problem, roughly 0.25 s in the current setup. However, the oldest pose in this series will have a delay of 1.25 s since we use a window shift of 1 s. In addition, the C-LOC pose is not just delayed but also output at a low rate of 1 Hz. For control purposes, this is both too slow and too far behind real-time. We therefore address those two problems by integrating the locally accurate wheel odometry between two consecutive C-LOC poses. As wheel odometry is received, it is integrated from the time of the last received C-LOC pose. Since the wheel odometry is published at a relatively high frame rate of 10 Hz, this produces a trajectory estimate that is sufficient for controlling a UGV at a low speed.

III. Implementation-Specific Considerations

In this section, we discuss the specific hardware configuration, followed by implementation-specific software solutions.

A. Hardware

The main hardware modules are the following:

- Gator: All experiments are performed using a John Deere TE Gator, an all-electric utility vehicle. The base vehicle was automated by our Team at CSIRO. Hence, the vehicle can be driven both manually or autonomously. Four 2D Hokuyo UTM-30LX LiDARs are used for local obstacle avoidance and coverage of the back of the vehicle. The localization algorithm runs on a nodding Velodyne VLP-16 PUCK LiDAR jointly with a Microstrain 3DM-GX3-25 IMU.
- Dell workstation: Customized Dell Precision M4800 with a 2.9 GHz Intel Core i7 4910MQ CPU, 32 GB of RAM and an Intel Haswell Mobile graphics card. The workstation runs on Ubuntu 14.04 and is used with the Robot Operating System (ROS) Indigo.
- Microcontroller: A smooth nodding motion is generated by sending commands to the serial port of the Dynamixel servo at 40 Hz. Using a separate microcontroller, the Atmel AVR XMEGA, the load on the workstation, which runs all the high-level algorithms, is reduced.
- Clock: A u-blox evaluation kit EVK-M8MEVA is used to create a precise Pulse Per Second (PPS) signal to synchronize the 3D LiDAR and the IMU.
- 3D LiDAR: We use a Velodyne VLP-16 PUCK LiDAR with a Horizontal Field of View (HFOV) of 360 ° and a Vertical Field of View (VFOV) of 30 °. The PUCK runs on firmware 3.0.32.0 and reports up to 289'351 points per second using its 16 different laser channels with the rotation rate set to 20 Hz.
- IMU: A Microstrain 3DM-GX3-25 is rigidly attached to the Velodyne PUCK. Both sensors are tightly coupled and used for full 3D localization. The orientation of the IMU is used to transform the Velodyne points into a world fixed reference frame and the accelerations are used as a motion prior.
- Servo: A Dynamixel MX-106R servo motor, which is a contactless absolute encoder is used to create a nodding motion with the Velodyne PUCK. Currently the PUCK is nodded $\pm$ 30 ° at a frequency of 70 °/s. The nodding setup is illustrated in Fig. 4.

B. Software

1) Point Cloud Filters: We subsample the Velodyne point cloud before feeding it into the localization algorithm C-LOC, for computational speedup. First, we create a ring filter to reduce the 16 beams to only 4: the top, the bottom and two equally spaced in the middle. In addition, we further reduce the amount of points by voxelizing the entire point cloud. This is done using the PCL octree filter with a minimum voxel resolution of 10 cm. Using this nodding setup together with a reduced number of beams leads to an increased field of view with spatial diversity while keeping the computational effort small.

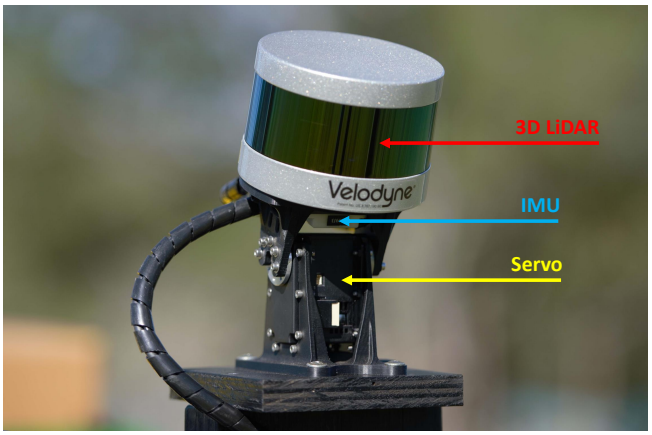


Fig. 4. Nodding setup consisting of the Velodyne PUCK 3D LiDAR, the Microstrain 3DM-GX3-25 IMU and the Dynamixel MX-106R servo.

2) Map Creation Using Offline C-SLAM: We create the 3D base map used for localization by manually driving around the QCAT site and recording the synchronized LiDAR and IMU data. For map creation, the LiDAR data are again subsampled, but in this case only using the octree voxel filter (the ring filter discussed above is used only during online localization C-LOC). The recorded information is then processed offline, generating: (i) a 3D point cloud of the environment; (ii) the 3D LiDAR trajectory driven during data recording (see Fig. 5).

3) 2D Occupancy Grid Creation for Navigation: Unlike the full 3D localization C-LOC, the entire navigation pipeline runs in 2D and we therefore need to create a 2D map from the 3D map. We use a volumetric 3D environment model, based on the Octomap algorithm [17]. Octomap is a 3D occupancy grid framework relying on octrees which uses a probabilistic occupancy estimation. This framework represents not only occupied but also free and unknown cells. Hence, the entire environment is represented by small 3D grid cells, so called voxels. We use a grid resolution of 20 cm which represent the environment accurately enough for autonomous navigation.

Our 2D occupancy grid representation is obtained by down-projecting the 3D voxel grid. Before this is done, we filter the grid by setting voxels from the ground and from collision-free overhanging structures as free. Otherwise those voxels would appear as obstacles in the 2D map, which is not desired since we require a map where only obstacles relevant for the ground vehicle are shown. In addition, a box filter marks as free all the voxels which lie in a box around the LiDAR trajectory and therefore guarantees a free path.

The final 2D occupancy grid map which is used for navigation can be seen in Fig. 6. Obstacles are marked black, free space where the Gator can drive is white and unknown space is grey. It is worth mentioning that we did not include the hill observed in Fig. 5 into the occupancy grid in Fig. 6, due to the fact the experiments in this paper are limited to paved roads, which do not exist on the hill.

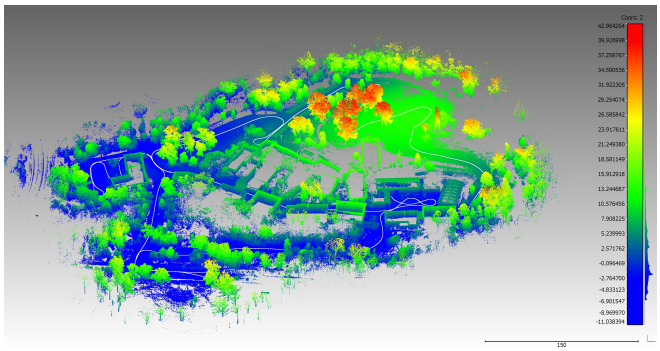


Fig. 5. Height coded 3D point cloud map of the entire QCAT site including the 3D LiDAR trajectory in white. The entire point cloud consists of about 315 million points and covers an area of 160'000 m^2. Units are in meters.

4) Controller and Path Planner: We use the ROS navigation stack[2] for computing heading and velocity commands to navigate in autonomous mode. More specifically, we use `move_base`[3] to combine both a local and a global path planner for navigation. In order to do this, `move_base` uses the 6 DOF pose, the 2D occupancy grid map and the four Hokuyo LiDARS as an input.

We use costmaps from the navigation stack to define where the robot is allowed drive. They contain information from both static maps and live sensors such as LiDAR data. By using costmaps we can customize the generated trajectory of the path planner, without modifying the actual planner. In our implementation, we used the `navfn` global planner available in the ROS navigation stack. The planner works on Dijkstra's shortest path algorithm and computes paths for omnidirectional robots which are able to move in any directions instantaneously [18]. To use this path planner with an Ackermann-steering vehicle we restricted the allowable paths at the starting and goal point since the Gator can neither start nor reach the final waypoint by moving in any direction.

To solve this issue, we implemented a costmap layer which raises the cost in a U shape both around the starting point, where the U is open in the forward direction, and also around the goal point where the U acts like a bay. The raised cost is chosen such that the path planner considers the U shape as a normal obstacle and will never traverse those barriers. Hence, the vehicle can only leave its starting position in the forward direction and reach its goal point in the direction in which the U shape is open. This goal direction is specified by the user selected waypoint which consists of an x, y position and an angle ψ.

Another costmap for navigation helps the path planner create desirable paths. The idea is to input one or multiple manually driven paths into the navigation pipeline and slightly lower the cost along those routes. Since Dijkstra's shortest path algorithm searches for the path with the lowest cost, it automatically favors those previously driven routes. However, this does not restrict the drivable area in any way and the vehicle can still drive everywhere in the map even if there is no manually driven trajectory input into the navigation stack. During all the experiments presented in this paper one single manually driven trajectory was input into the navigation stack.

[2]`http://wiki.ros.org/navigation`
[3]`http://wiki.ros.org/move_base`

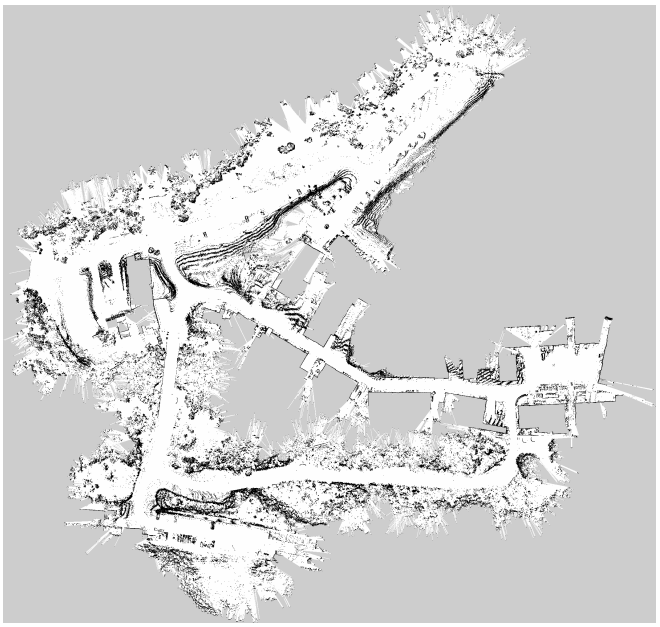


Fig. 6. Occupancy grid map (approximate scale: 420 m x 400 m) from the QCAT, Pullenvale site, created with a grid resolution of 20 cm.

IV. Experiments

To illustrate the applicability of the system, we completed various autonomous driving experiments at QCAT. First, ten runs using a "recent map" are presented. Second, five loops are performed using an older, outdated map. In addition, we use a high precision RTK GPS to compare the trajectory generated by the C-LOC algorithm to the real world. Finally, dozens of runs are made on a longer and more complex route.

During all the autonomous runs, the Gator drove between 1 m/s and 1.5 m/s and was supervised by an operator.

A. Overview of the Test Area

QCAT is a heterogeneous site and consists of areas with very different characteristics as illustrated in Fig. 7. Each region presents its own challenges. The blue region contains a large parking lot with constantly moving cars. This makes both localization and navigation more challenging. In the red region, mostly buildings are present. Using the Velodyne PUCK, localization is relatively easy in this area since buildings are static and hence the base map is very similar to the live LiDAR observation. Finally, the green region consists of a road delimited by trees and bushes. This region is particularly challenging for localization, especially in areas where only bushes are present because they grow, get frequently cut, drop their leaves and easily move with the wind.

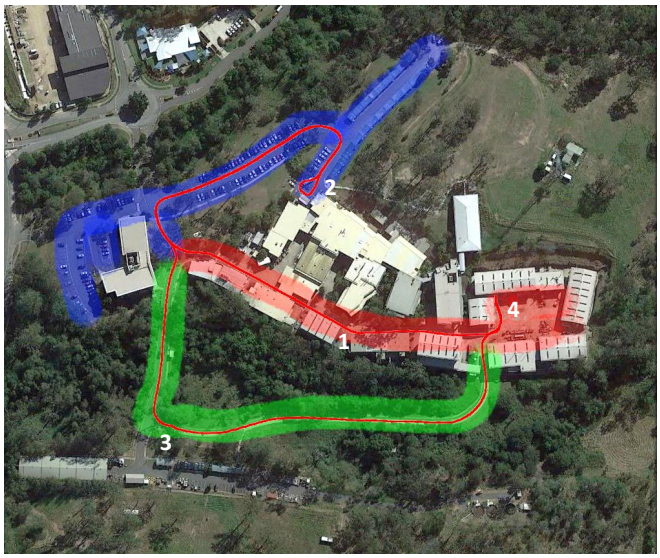


Fig. 7. Google Earth image of the QCAT site, with different regions highlighted. The blue region is a parking lot with continuously moving cars. In the red region, mostly buildings are present and in the green region, bushes and trees are dominant. Additionally, the autonomously driven trajectory (red line) with the four specified waypoints (white) is shown.

B. Sample Based Covariance

In order to evaluate the accuracy of the C-LOC pose we employ the sample based covariance as a performance metric. The sample based covariance is calculated by analyzing the pose change, from the initial guess to the final value during the optimization process. Since the C-LOC algorithm works on a sliding window approach, a pose is optimized multiple times during its "life" in the sliding window. Based on experimental evidence, we implicitly rely on the assumption that the covariance does not change significantly over a short period of time since we use past pose changes to calculate the covariance of the current pose.

More specifically, we compute the covariance by storing initial non-optimized poses. Every time the sliding window is shifted, we compare all the poses in the sliding window to their non-optimized counterpart. Currently, we use a sliding window of 3 s and shift the window by 1 s every iteration. This means that every pose is optimized three times before it is considered fixed. Hence, we calculate the sample based covariance matrix $\mathbf{Q}$ of the current pose based on the comparison of the last three poses to their non-optimized counterpart:

$$\mathbf{Q} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^T \tag{1}$$

where n is the number of pose comparisons, x_i is the difference vector of the i-th non-optimized pose with the i-th optimized pose and $\bar{x}$ is the mean of all the difference vectors. This covariance calculation is used in Fig. 8c to visualize the varying localization confidence in different parts of the QCAT site.

C. Current Map Experiments

In June 2016, we mapped the entire QCAT site and created both a 3D surfel map for localization and a 2D occupancy grid for navigation. Using four manually set 2D waypoints consisting of an x, y position and an orientation ψ, a 1161 m route with a height difference of about 14 m was driven completely autonomously. The path is represented by the red line in Fig. 7. Data from ten consecutive runs were recorded to test the reliability of the system. We set the waypoints once manually and re-used them for the remaining nine runs to allow for meaningful path comparison. In order to emphasize the robustness of the system, the map was created by driving in the opposite direction to the one later used for localization.

Results: All the ten, consecutive runs were performed completely autonomously and are shown with their exact starting date, time and unique color in Fig. 8a. Human intervention during the ten runs consisting of more than 11 km of driving took only place twice where the operator stopped the vehicle during 5-10 s to let heavy traffic pass. It can be seen, that the x, y positions of all the ten trajectories are very similar. This mainly shows that C-LOC worked reliably, neither was the algorithm lost at any time nor was there visible drift in any of the trajectories.

Analyzing the first of the ten runs, the sample based covariance (IV-B) can be color coded onto the trajectory, see Fig. 8c. The color coding at a certain time in the trajectory $c(t)$ is calculated as:

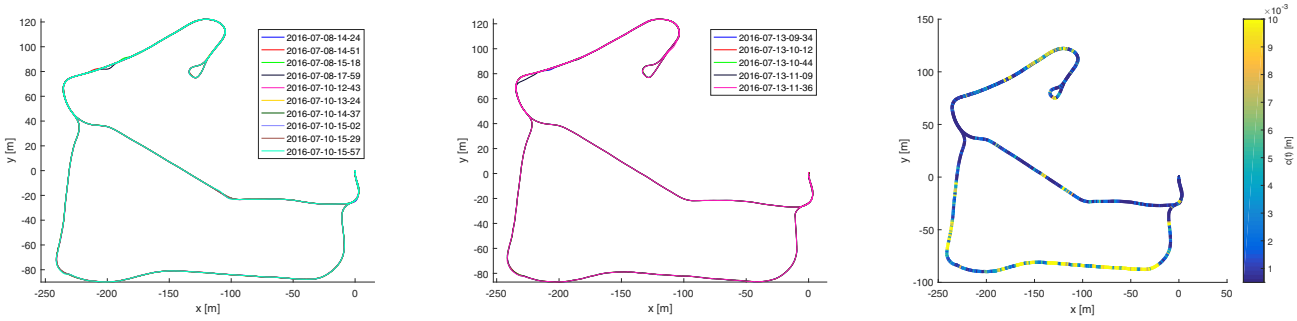
$$c(t) = \sqrt{q_x(t)^2 + q_y(t)^2} \quad t \in [0, T] \tag{2}$$

where q_x is the variance in x direction from the sample based covariance matrix $\mathbf{Q}$ and q_y respectively. The end time of the trajectory is denoted as T. In order to enhance the visibility in the plot, values are limited to a maximum of 0.01 m. However, for the statistical analysis presented in Table I all the ten runs were used without this limit. The C-LOC algorithm is less certain about the x, y position in yellow and vice versa in blue trajectory segments. When comparing this trajectory to the Google Earth image, Fig. 7 it becomes clear that C-LOC is more certain in areas with buildings and less in open areas or regions where trees and bushes are dominant. This is what we expected since localization in changing bushland regions is more challenging than in areas with static buildings.

D. Outdated Map Experiments

We show that an outdated base map can be significantly different to the current map and still allow localization to work successfully. We perform five consecutive, completely autonomous loops using a map which was created about 1.5 months before the current map. During this time, trees and bushes have changed their appearance, cars and barrels have been moved around.

Results: Using the outdated map, five consecutive, completely autonomous runs were performed and plotted in Fig. 8b. During all the runs not a single human intervention took place. Similar to the ten consecutive runs using the



(a) Current map. (b) Outdated map. (c) Variance visualization.

Fig. 8. Autonomous driving results using a new and an outdated map. The trajectories are colorized and labelled with their respective date and time. In addition, the C-LOC variance of the first current map run, is shown using the color coding c(t) with blue, low variance and yellow, high variance.

TABLE I

FOR ALL 15 C-LOC PATHS (TEN WITH THE CURRENT, FIVE WITH THE OUTDATED MAP), $c(t)$ WAS CALCULATED SEPARATELY AND SUBSEQUENTLY USED FOR TWO SEPARATE STATISTICAL ANALYSES.

	Current map	Outdated map
Mean [m]	0.005	0.007
Std. dev. [m]	0.007	0.010
Min. [m]	1.153e-05	1.441e-05
Max. [m]	0.053	0.083

current map, the trajectories of the five runs are nearly identical, therefore no visible drift occurred and the Gator never got lost.

In Table I we present the same statistical analysis as in the current map results, using all the five available runs performed with the outdated map. When comparing the statistical analysis of the variances obtained from the outdated map to the one from the current map, it can be seen that all the values from the outdated map are higher. This makes sense, since the outdated map is older, and therefore has changed more over time than the newer, current map. As a consequence, localization becomes more challenging and the variances increase.

E. RTK GPS Comparison Experiments

For comparisons, we use a high precision Novatel OEMV 3G Real-Time Kinematic (RTK) GPS with a GPS-702-GG Dual-Frequency antenna for ground truth. Using WiFi to connect to the CSIRO wireless LAN network, the correction factor from a base station, a Septentrio PolaRx3eG Pro, is constantly transferred to the Novatel GPS.

Results: We navigated autonomously while recording both the C-LOC and a high precision RTK GPS trajectory. Since the C-LOC and the RTK GPS trajectories were in different frames, we used an affine transformation to align them (see Fig. 9a). As expected the RTK GPS performs badly at the beginning of the trajectory where large buildings are located. Everywhere else, the GPS seems to perform well and can be used as reliable ground truth. Figure 9a indicates that the trajectory reported by C-LOC aligns very well with the one reported by the RTK GPS.

To quantify the difference of the C-LOC path to the RTK GPS path we used the Euclidean norm: $e(t) = \sqrt{e_x(t)^2 + e_y(t)^2}$ with $e_x(t) = x(t)_{RTK} - x(t)_{CLOC}$. In Fig. 9b the Euclidean norm $e(t)$ is color coded onto the C-LOC trajectory. The trajectories were only compared in areas where the RTK GPS performed well and hence a part of the trajectory is missing. The minimum of the Euclidean norm is 0.096 m, the maximum 2.004 m and the mean 0.995 m. It is worth mentioning that those values can be considered an upper bound of the actual difference between the C-LOC and the RTK GPS path. This is due to the fact that during comparison small errors were introduced. First, the two paths were aligned using an affine transformation. Second, both the RTK GPS and the C-LOC path were linearly interpolated for comparison.

In addition, the RTK GPS path was plotted on a Google Earth image (see Fig. 9c) using the online GPS Visualizer[4]. The RTK GPS trajectory seems to align well with the streets of the QCAT site. Taking into account all the performed RTK GPS comparison experiments, we can conclude that the trajectory reported by the localization pipeline C-LOC accurately represents the real world trajectory.

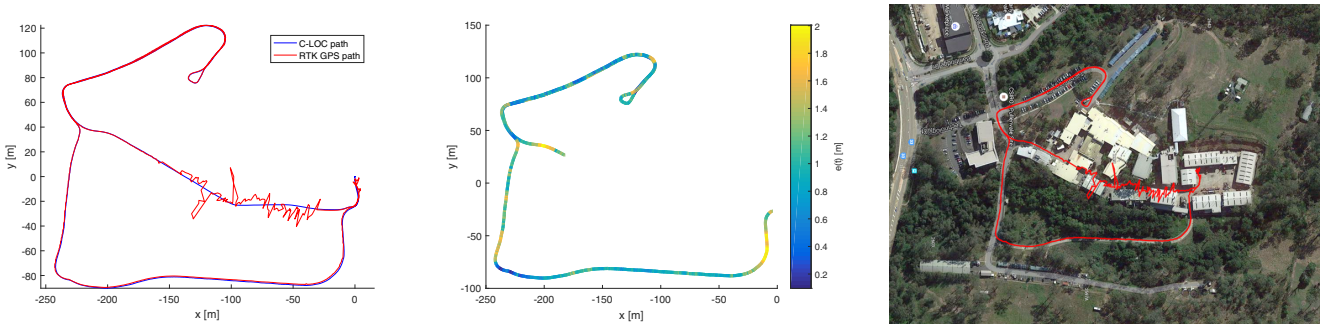
F. Additional Runs

In addition to the well-defined and repeatable experiments described above, a longer route of 1310 m was driven completely autonomously dozens of times for testing purposes and to demonstrate the working system to both CSIRO internal and external people. This longer loop was also used to create a video[5]. It includes navigation through an additional loop and intersection within the site. Once the base system was set, no navigation or localization failures were observed in any of the runs.

V. CONCLUSIONS

So far, more than 60 km of autonomous driving around the QCAT site has been performed using the novel navigation stack presented in this paper, without experiencing a single system failure. This provides evidence that the system works

[4]http://www.gpsvisualizer.com
[5]https://youtu.be/nFGYMIXhqF4



(a) C-LOC, RTK GPS paths. (b) C-LOC error plot. (c) Google Earth overlay.

Fig. 9. Comparison of the C-LOC path to the RTK-GPS path.

reliably in a dynamic and arguably challenging environment. By presenting experiments with both a new and an outdated base map, it was shown that even though the base map is highly accurate, it is not crucial for successful localization and navigation. Changes in the map such as moving cars, barrels or also vegetation change can be tolerated without causing the system to fail. Finally, the performed RTK GPS experiments have successfully shown that the path reported by the localization system C-LOC corresponds to the actually driven one in the real world.

ACKNOWLEDGMENTS

We would like to thank Kazys Stepanas for helping with the creation of the 2D occupancy grid, Dave Haddon for his support regarding the Gator, Ross Dungavell for his assistance with the RTK-GPS, Brett Wood for designing the timing board and Robert Zlot for his corrections in the paper.

REFERENCES

[1] A. Broggi, P. Medici, P. Zani, A. Coati, and M. Panciroli, "Autonomous vehicles control in the vislab intercontinental autonomous challenge," *Annual Reviews in Control*, vol. 36, no. 1, pp. 161–171, 2012.

[2] A. M. Kessler, "Elon musk says self-driving tesla cars will be in the us by summer," *The New York Times*, p. B1, 2015.

[3] 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*, pp. 4312–4319, May 2009.

[4] 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, pp. 1104–1119, Oct 2012.

[5] R. Zlot, M. Bosse, K. Greenop, Z. Jarzab, E. Juckes, and J. Roberts, "Efficiently capturing large, complex cultural heritage sites with a handheld mobile 3d laser mapping system," *Journal of Cultural Heritage*, vol. 15, no. 6, pp. 670–678, 2014.

[6] R. Zlot and M. Bosse, "Efficient large-scale three-dimensional mobile mapping for underground mines," *Journal of Field Robotics*, vol. 31, no. 5, pp. 758–779, 2014.

[7] M. Bosse and R. Zlot, "Place recognition using regional point descriptors for 3d mapping," in *Field and Service Robotics*, pp. 195–204, Springer, 2010.

[8] M. Bosse and R. Zlot, "Place recognition using keypoint voting in large 3d lidar datasets," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 2677–2684, IEEE, 2013.

[9] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems Conference (RSS)*, pp. 109–111, 2014.

[10] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2174–2181, IEEE, 2015.

[11] M. Beul, N. Krombach, Y. Zhong, D. Droeschel, M. Nieuwenhuisen, and S. Behnke, "A high-performance mav for autonomous navigation in complex 3d environments," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*, pp. 1241–1250, IEEE, 2015.

[12] D. M. Cole and P. M. Newman, "Using laser range data for 3d slam in outdoor environments," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 1556–1563, IEEE, 2006.

[13] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 699–722, 2007.

[14] M. Whitty, S. Cossell, K. S. Dang, J. Guivant, and J. Katupitiya, "Autonomous navigation using a real-time 3d point cloud," in *2010 Australasian Conference on Robotics and Automation*, pp. 1–3, 2010.

[15] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, B. Rebsamen, P. Dai, E. Rankin, and M. H. Ang Jr, "Autonomy for mobility on demand," in *Intelligent Autonomous Systems 12*, pp. 671–682, Springer, 2013.

[16] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

[17] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[18] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Autonomous mobile robots," *Massachusetts Institute of Technology*, 2004.