Long-term 3D map maintenance in dynamic environments

François Pomerleau, Philipp Krüsi, Francis Colas, Paul Furgale and Roland Siegwart

***Abstract*— New applications of mobile robotics in dynamic urban areas require more than the single-session geometric maps that have dominated simultaneous localization and mapping (SLAM) research to date; maps must be updated as the environment changes and include a semantic layer (such as road network information) to aid motion planning in dynamic environments. We present an algorithm for long-term localization and mapping in real time using a three-dimensional (3D) laser scanner. The system infers the static or dynamic state of each 3D point in the environment based on repeated observations. The velocity of each dynamic point is estimated without requiring object models or explicit clustering of the points. At any time, the system is able to produce a most-likely representation of underlying static scene geometry. By storing the time history of velocities, we can infer the dominant motion patterns within the map. The result is an online mapping and localization system specifically designed to enable long-term autonomy within highly dynamic environments. We validate the approach using data collected around the campus of ETH Zurich over seven months and several kilometers of navigation. To the best of our knowledge, this is the first work to unify long-term map update with tracking of dynamic objects.**

***Index Terms*— Long-term mapping, dynamic obstacles, ICP, kd-tree, registration, scan matching, robot, SLAM.**

I. Introduction and Related Work

The success of SLAM has been a major enabler of robot autonomy. Until recently, the majority of research focused on increasing the accuracy and robustness of single-session SLAM. Now that robotic hardware and software are becoming more widespread, applications such as navigation in dense crowds [1], or autonomous driving in cities [2] are demanding new algorithms that can maintain maps over time. A SLAM system should not only be able to build a map of the geometry of an environment, but it should also be capable of updating this map over time and of encoding useful semantic information for planning. For example, a map that includes lane and average speed information may be used to calculate the probable trajectories of cars tracked by a robot, so planning a collision-free path may be focused on likely trajectories and thus, can be computed faster. Prior knowledge of motion patterns within a map may also be used to detect anomalous behavior of other agents so that extra attention may be paid to these agents. Furthermore, it is important that dynamic objects are not included in static environment maps as they may degrade localization accuracy and cause motion planning to fail. Motivated by these applications, we therefore seek to develop a mapping and localization system that is able to build and maintain these hybrid geometric/semantic maps.

The authors are with the Autonomous Systems Lab, ETH Zurich, 8092 Zurich, Switzerland. f.pomerleau@gmail.com

Fig. 1. ARTOR, a search and rescue robot specialized for outdoor applications navigating in a highly dynamic urban environment. Typical mobile elements include pedestrians, bikers, cars, trucks and trams.

This paper presents an algorithm that performs SLAM, map updating, classification of map points as dynamic or static, and estimation of the velocity of dynamic points, all in real time and using only data from a 3D laser scanner. The approach is bottom up; it does not rely on prior information, such as object models. The only prior used is a weak smoothness assumption on the velocity of dynamic scene points. By saving the time history of point-wise velocity estimates, we are able to infer the dominant motion patterns in the map. The algorithm is validated using data collected over seven months with the field robot shown in Fig. 1 in the highly dynamic urban area around the main campus of ETH Zurich.

The work presented here is related to two research areas: map updating for lifelong navigation and segmentation of dynamic obstacles during localization and mapping. As the techniques used in laser processing and vision processing have not yet converged on common solutions for either of these problems, we will restrict our survey of prior work to papers that use lidar data.

The work of Biber et al. [3] describes a long-term map updating scheme that shares the same goals as our framework. They describe a long-term two-dimensional (2D) SLAM system based on scan matching and odometry. The mapping system maintains a set of local maps (2D scans at fixed positions) with multiple hypotheses for the range values at different timescales. Specifically, they highlight four requirements for any long-term mapping strategy: (1) map adaptation should not depend on the wall-clock time, (2) mapping should be resilient to outliers, (3) multiple

hypotheses should be maintained until there is enough data for inference, and (4) the map should only contain measured values (not interpolated or smoothed quantities). Our method fulfills the same requirements, but we extend the capability to 3D space, remove the dependence on viewpoint and, rather than maintaining multiple hypotheses, we probabilistically segment samples into static and dynamic. Our segmentation strategy is similar to the dynamic mapping technique of Burgard et al. [4] who use an expectation maximization scheme to differentiate between dynamic and static cells in a 2D grid map. They show that removing dynamic points from the data during mapping increases localization accuracy. However, they make no attempt to cluster or track dynamic objects throughout a scene. Theoretically, it should be possible to extend their method to process 3D laser data by adopting an efficient 3D grid representation such as Octomap [5]. However, the processing of data with long sensor beams in large outdoor areas can become prohibitively slow due to the requirement to ray-trace through the grid. In contrast, our method works in real time on raw 3D laser data in expansive environments. The trade-off is that we do not explicitly model free space throughout the entire volume of the mapped area—only in places where our laser has returned a previous reading. While the explicit modeling of free space may be required for mobile manipulation or flying robots navigating in tight spaces, it is not strictly necessary for a large class of mobile robots with local sensing that is accurate enough to infer the drivable area directly around the robot.

There have been a number of other algorithms developed specifically for updating map over several passes through an environment. Aijazi et al. [6] use highly accurate localization based on a differential global positioning system to resolve points from a single pass into a fixed grid. Grids from multiple passes can then be directly compared to infer the static parts of the scene. The algorithm produces excellent segmentation results, but the reliance on highly accurate localization makes it unsuitable for the general case of a moving robot. Ryde and Hillier [7] also use a grid-based representation to detect changes in 3D laser maps. They sidestep the need for accurate localization by matching the latest point cloud with an existing voxel grid. A change is detected after alignment by finding points that do not lie within an occupied voxel.

Another class of work has ignored map update to focus on the segmentation and tracking of dynamic objects from laser data. Several studies have produced notable results using a static 3D laser [8], [9], [10]. However, it is not immediately clear how to extend them to a moving sensor. Wang et al. describe an impressive system for 2D laser-based SLAM with moving object tracking [11]. The system segments the incoming laser scan, matches segments with the predicted locations of tracked objects, and accumulates point-based models of the dynamic objects and the static scene. Moosman and Fraichard have developed a very similar system for 3D laser data [12]. However, both approaches rely on the initial range-image segmentation step, and neither one addresses the difficult problem of splitting tracks that can appear when

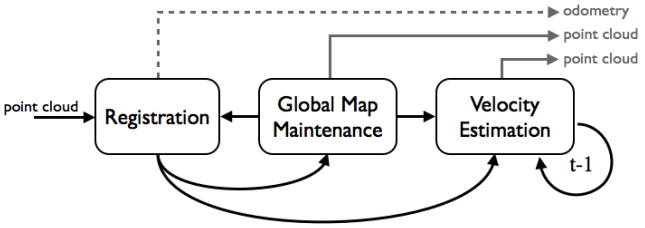

Fig. 2. Block diagram of the processing pipeline. Boxes represent separate processes running at different frequencies. Solid arrows represent point clouds being communicated to each modules and dashed line, odometry. The gray arrows show the output of each module.

the initial segmentation step clusters objects with similar positions and speeds. In contrast, our approach is entirely point-based with a weak smoothness prior on velocities. The lack of reliance on segmentation allows us to seamlessly handle clumps of agents that merge and diverge without relying on specialized detection and handling of these cases.

To the best of our knowledge, this is the first work to unify map update with tracking of dynamic obstacles, and we believe it is a significant step toward the unified geometric and semantic mapping needed for robot navigation in highly dynamic environments.

II. System Overview

A general overview of the proposed system is presented in Fig. 2. Point clouds from the sensor are provided to the *Registration* module, which corrects the odometry of the robot and provides the point clouds in global coordinates. The *Global Map Maintenance* module evaluates if past information were dynamic or not, and concatenate the new registered point cloud in order to keep the global map at a constant density. The *Velocity Estimation* module takes the newly registered point cloud and segments it based on the information of the global map. What is currently considered as dynamic at time t is than evaluated against the last dynamic elements at time $t-1$. The module outputs a point cloud with all the points having a speed larger than zero. The 3D registration module is based on the already published `libpointmatcher` [16], therefore this article will focus on the two other modules.

III. Dynamic Element Identification

We infer dynamic parts of the scene based on visibility assumptions; if we observe a laser point behind a point that was previously observed, that previous point might be dynamic. For such process, the standard approach is ray-tracing. While widely used in 2D, scaling up to 3D is expensive in memory as it requires a dense representation of both the occupied and free space. To ensure online computation, we propose to directly use the same representation as for the localization: sparse point clouds. First, we transform a local subset of the point cloud map Q into the reference frame of the current point cloud P. Then, using spherical coordinates, we associate the points q of the map to each single reading point p in the same small conical aperture

of size θ_{max}. This can be done quickly using an efficient kd-tree implementation, `libnabo` [13]. All the points of the map that are further than the point of the current sensor reading in each cone are left untouched. However, the points of the map that are closer than the reading need to be updated as they should have intercepted the ray. Based on our field observations, this update should fulfill the following criteria:

1) the greater the angle between the beam producing p and q, the less we change the knowledge on q,
2) the greater the angle between the beam producing q and its surface normal n, the less we change the knowledge on q,
3) if p and q are spatially close, q is more probably static; otherwise more probably dynamic,
4) a point has more chances to become dynamic knowing that it is static than the inverse,
5) most of the new points observed are static.

Most of those criteria are easy to motivate, except maybe criterion 2), which require more explainations. Indeed, the sensor produces readings of environmental elements that can be located at up to 80 m from the sensor itself. At that distance, many points from the ground can be in the cone of a reading point due to the big incidence angle, but they should not be considered as dynamic. This problem was also observed by Wurm et al. [5] but not explicitly addressed in their updated equations.

In order to give a formal expression to those criteria, we observe that the problem is to update some knowledge state based on uncertain information. Therefore, we use a Bayesian approach to update the knowledge on each map point q. We consider the following notations for the variables, while the parameters are defined in Table I:

- p: point from a newly acquired reading expressed from the center of the sensor,
- q: point of the global map expressed from the center of the sensor and associated to p,
- Dyn: binary variable indicating whether q is now dynamic or not,
- $Odyn$: binary variable indicating whether q was dynamic or not,
- U: binary variable indicating whether we need to update the point or not,
- θ: angle between p and q defined as $\operatorname{acos}\left(\frac{q \cdot p}{\|q\| \cdot \|p\|}\right)$,
- ϕ: incidence angle on q based on its surface normal n defined as $\operatorname{acos}\left|n \cdot \frac{q}{\|q\|}\right|$,
- δ: distance between q and p defined as $\|p - q\|$.

We can write:

$$P(Dyn|\theta,\phi,\delta) \propto \sum_{U,Odyn} \begin{vmatrix} P(Odyn)P(U|\theta,\phi) \\ \times P(Dyn|Odyn)P(\delta|Dyn) \end{vmatrix} \quad (1)$$

Where:

- $P(Odyn)$: is either a prior (80% chance to be static) or the result of a previous inference,
- $P(U|\theta,\phi) = \begin{cases} \frac{\theta}{\theta_{max}}\left(1 - \frac{2\phi}{\pi}\right) & \text{if } \|p\| \geq \|q\| \\ 0 & \text{otherwise} \end{cases}$ is the probability to update based on separation and incidence angles,
- $P(Dyn|Odyn) = \begin{bmatrix} \alpha & 1-\beta \\ 1-\alpha & \beta \end{bmatrix}$ is a decay matrix to allow points to change,
- $P(\delta|Dyn) \propto \begin{cases} \max(0, \min(1, \epsilon_d + \|p\|\epsilon_a - \delta)) & \text{if } Dyn \text{ is false} \\ 1 - P(\delta|Dyn = false) & \text{if } Dyn \text{ is true} \end{cases}$ is the observation probability distribution.

With this model, we are able to compute whether a point is dynamic or static based on multiple observations. In the results section, we will demonstrate that this works well for highly dynamic objects. However, it can have issues with dynamic objects that are periodic (i.e., object that comes back at the same location often). This applies particularly to trams that are constrained to their tracks and to cars parked in well-defined parking spaces. The definition of a static object becomes therefore ambiguous, and higher-level models of objects are needed. Here, we retain a bottom-up approach by deciding that if an object is sufficiently dynamic—meaning that it was seen and disappeared—then it cannot go back to being static if $P(Dyn = true) \geq \gamma_{max}$.

TABLE I

DEFINITION OF THE PARAMETERS AND THEIR VALUES USED FOR OUR TARGETED SCENARIOS.

	Values	*Descriptions*
θ_{max}	1°	Angle around which all points in Q are associated to a point p.
α	0.99	Probability of staying static given that the point was static.
β	0.90	Probability of staying dynamic given that the point was dynamic.
ϵ_d	0.1 m	Fixed noise on depth measurement of a point p.
ϵ_a	0.2	Ratio of noise based on depth measurement of a point p.
γ_{max}	0.9	Probability at which a point is considered permanently dynamic.

IV. VELOCITY ESTIMATION

Building on top of the dynamic object classification, one can estimate the velocity of moving objects. To be useful for dynamic obstacle avoidance algorithm, like the one proposed by Rufli et al. [14], velocity must be extracted at high rate. Most approaches rely on the clustering of the points into objects for which the velocity is then estimated looking, for example, at the change in position of the center of mass. In this section, we briefly introduce our fast and generic approach as a complement to the dynamic element classification.

From a newly acquired point cloud P_t at time t, we associate all of its points to the global map. A subset of mobile points M_t is generated from P_t, fulfilling the requirement of being a dynamic obstacle. This can be based on the dynamic element identification (as described in the previous section) of the global map and on the definition of obstacles for a given platform. Those dynamic obstacles M_t

can then be compared to the last subset M_{t-1} to extract velocity vectors. We based our approach on point-cloud registration using iterative closest point (ICP), where M_t is the *reading* point cloud and M_{t-1} is the *reference*. Having different transformation parameters for each point is known as non-rigid ICP [15]. We reuse the underlying principles but extracted only translation components instead of the full 6 degrees of freedom (DOF) transformation. In essence, we propose to do dual non-rigid ICP—both from reading to reference and from reference to reading—and, given that we have a timestamp per point, divid the alignment error with the difference of acquisition time to estimate the velocity vectors. We use neighboring constraints to harmonize the velocities across close points.

As shown in Fig. 3, the measurements received are sparser than typical full point clouds used with ICP. Moreover, a high ratio of noise is possible, especially during exploration of new areas where not enough information has been acquired to accurately classify points. To cope with those challenges, we vary the number of nearest neighbors inversely proportional to the number of iterations. More precisely, for every p_{read} in M_t, we assign k nearest neighbors p_{ref} from the reference point cloud M_{t-1}. Then, we compute the average velocity vector produced by the k matched points and assign it to p_{read}. This augments the robustness of the association phase against noisy matches when using a large k, while keeping the accuracy of a single match at the last iteration. To ensure locally coherent velocities, we apply a windowed-mean filter in the Euclidean space for all p_{read}. The iterative process reuses velocity estimates of the latest iteration to project points before association with the speed of new points initialized to zero. In the current instantiation of the algorithm, five iterations were sufficient for convergence.

Often, objects change rapidly in term of shape and density between two scans, which lead to an asymmetry between the velocity estimation from the reading to the reference, and vice-versa. To cope with this situation, we compute both directions of matching in parallel. Fig. 3 shows an example of this process on a pedestrian at different viewing distances with points projected in both directions. As it can be seen from the projected points (in green and blue), our process is able to correctly estimate the velocity of the points by re-projecting them properly to the other point cloud taken at a different time. It also shows the importance of the iterative process; with point clouds 1 m apart, points from one point cloud would all initially match a few points of the other, like an extended foot or arm.

Having several iterations that average over matches and neighbors fosters local consistency in the velocity estimate while allowing deformations. Although multiple nearest neighbor searches are used, only one kd-tree generation is required for every new scan, which gives a very fast computation. As opposed to cluster-based approaches, all dynamic objects can be treated in parallel as each point has its own velocity. Timing will be discussed in the results section.

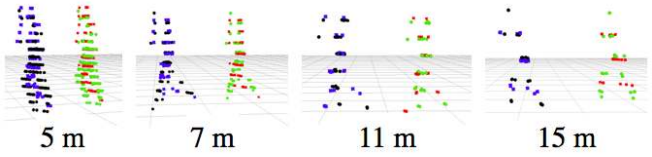

Fig. 3. Registration example of a pedestrian over different distances from the robot. Points in black are the current readings and the points in red are the references, taken roughly 0.5 s before. The points in green are the projections of the readings to the reference, while the points in blue are the projections of the reference to the readings.

V. Experimental Results

Although we could evaluated our approach in a simulated environment, we decided to go for real environments leading to a richer set of events at the expense of a direct access to precise ground truth information. We validated our approach with two scenarios in controlled environments and two large scale (i.e., kilometer range) experiments with daily challenges caused by dynamic elements. The robot assigned to the task was ARTOR (see Fig. 1), a platform based on the LandShark system by Black-I Robotics (USA), with custom modifications realized by RUAG Land Systems (Thun, Switzerland). The robot has a maximum speed of 3.5 m/s but is typically driven at 1 m/s in crowded environments. Equipped with a large sensor suite, the final platform is suitable for applications ranging from Search & Rescue to surveillance and reconnaissance. The main sensor used for the experiments is the Velodyne HDL-32E. It produces roughly 70'000 points per 360° scan at a rate of 11 Hz and with a maximum range of 80 m. The map representation is a set of sparse points with information about their surface normals, timestamps, probability of being dynamic, etc. A new point is only added to the map if the distance to its nearest neighbor in the global map is larger than 0.3 m. This keeps the point density of the global map constant, keeping the computation time close to real time. All the following results were obtained by running the input data at the same rate at which they were recorded.

A. Dynamic Element Segmentation

The goal of this experiment is to evaluate the capability of the system to identify dynamic elements. We selected the visitor parking lot of a hospital, which means that cars do not typically stay overnight. The section of the parking lot we surveyed has 50 dedicated parking places. Moreover, the middle of the parking lot is also a busy bike path during the day. Thus, this environment presents two kinds of dynamic obstacles: cars that come and go in between experimental runs, and bikes and pedestrians that move during the runs.

The robot was driven around to survey the area at different times during three consecutive days. The first survey was considered as the exploration phase, while the following missions are built upon the prior map. Points are kept in the map independently of their categorization and only split at the end for evaluation with the threshold $P(Dyn = true) < 0.5$. The environment and the path the robot did is depicted in

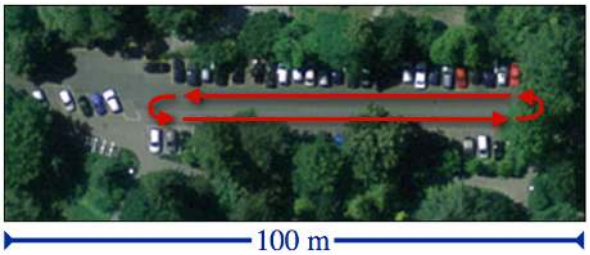

Fig. 4. Aerial view of the parking lot used for the segmentation experiment. In red, the survey path realized by the robot. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000.

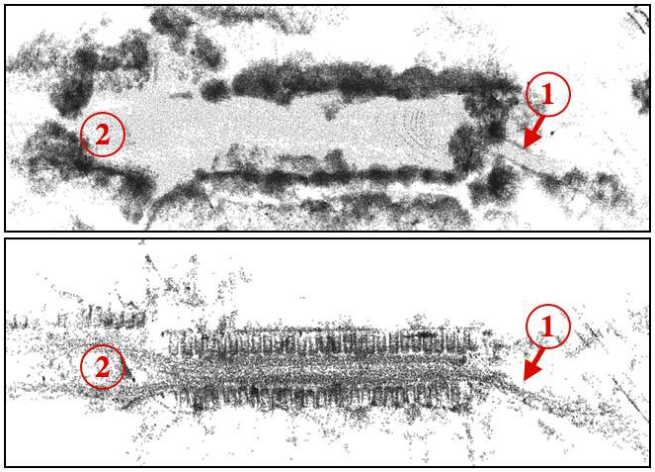

Fig. 5. Result of the segmentation after 9 surveys over the course of 3 days. *Top*: Reconstruction with $P(Dyn = true) < 0.5$. *Bottom*: Reconstruction with $P(Dyn = true) > 0.5$. The flow of pedestrians and bikes can be seen using a path (1) and splitting to avoid another row of parked cars (2).

Fig. 4. During the experiment, 61 pedestrians, 27 bikes and 10 cars were in motion through the surveyed area, without considering the multiple punctual changes in parked cars that appeared of disappeared through the days.

In addition to those surveys, we produced a ground truth map of the environment with a density that was three times higher than our regular maps. We recorded this map at night in order to have neither cars parked nor bikes or pedestrians. This map constitutes our ground truth for static elements and is not part of the evaluation set. In the following experiment, a point in the survey map is considered static if there is a point in the ground truth map within a radius of 0.15 m; otherwise, it is considered dynamic.

Fig. 5 shows the resulting segmentation between static and dynamic points. In the static map (top panel), the trees and the ground are clearly visible, whereas in the dynamic map (bottom panel) the cars are well highlighted as well as many points in the middle belonging to bikes or pedestrians. Moreover, the comparison of both maps in position (1) shows a path in the static map and trails from pedestrians and bikes in the dynamic map. The same comparison in position (2) shows the flow of pedestrians and bikes splitting around parked cars.

Using our ground truth map, we can evaluate the error

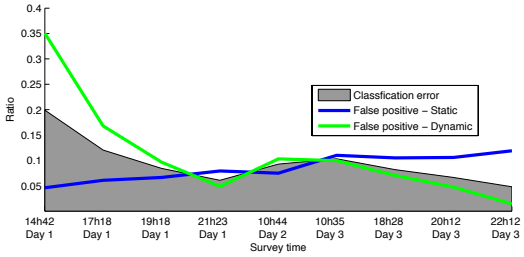

Fig. 6. Evolution of the segmentation results for each surveyed time. Blue line: static points wrongly identified with $P(Dyn = false) < 0.5$. Green line: dynamic points wrongly identified with $P(Dyn = true) < 0.5$. Shaded area: total classification error expressed as the ratio of wrong classification over the total number of points.

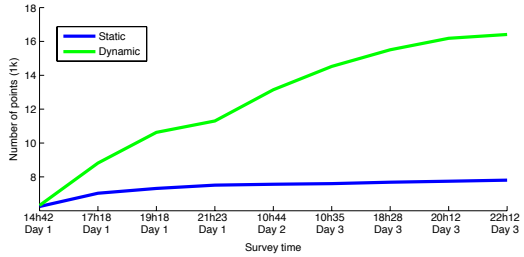

Fig. 7. Evolution of the static (in blue) and dynamic (in green) points added to the map for each surveyed time. The classification is done by comparison with our ground truth map.

rate of our classification. Fig. 6 presents both the global classification error, as well as the error for each class. If we distinguish static points from dynamic points, we can see two different evolutions. On the one hand, the error on the dynamic points decreases over time. This is expected because we need to observe that the point is missing to provide a classification. On the other hand, we see a steady increase in the rate of static points. Those points are mainly points that were close to the ground and were classified as dynamic; often, they are the lower part of cars. Finally, we observe an overall decrease of the error, from 20% to around 5%. This shows that there are more points that are dynamic than static.

Fig. 7 illustrates the evolution of the number of static and dynamic points in the map. This graph confirms that the number of static points is smaller than of dynamic points and that the difference increases with time. As expected, the number of new points added into the map at each visit decreases with each run as the environment gets to be better known. Moreover, the static information gets added into the map faster than dynamic elements and at the end of the first day, most static points are there. The plateaus in the dynamic points at the end of both day 1 and day 3 show the decrease in activity in the evening for this parking lot. Finally, with each successive run, dynamic points are added into an almost uniform 2 m-thick layer above the ground. These points represent all the pedestrians and bikes that have crossed the area during all the runs. The main issue about those dynamic points, greatly outnumbering static points, is that they can prevent ICP localization to properly align the

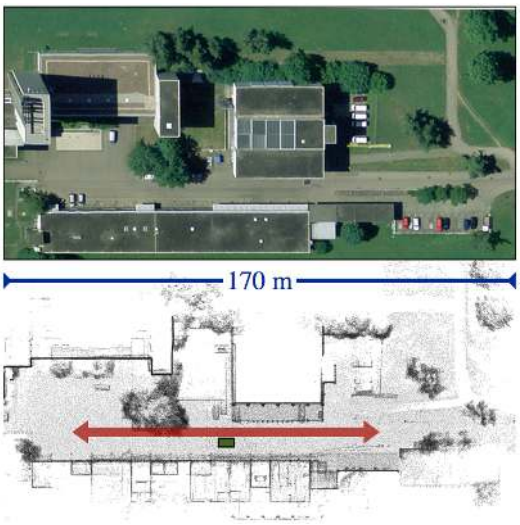

Fig. 8. Experimental setup for the velocity estimation. *Top*: Aerial view of the street used for the tests. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Bottom*: 3D reconstruction of the street with the zone reserved for our controlled dynamic elements in red and the position of the robot in green.

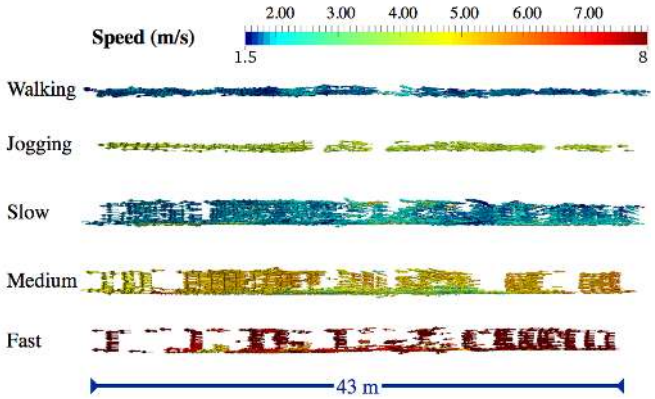

Fig. 9. Top view of the 3D dynamic trails. The 2 first lines represent the trails of a pedestrian, and the 3 last lines represent the ones of a minibus. The color indicates the estimated velocity in m/s.

current sensor reading with the map, causing a slow drift in the localization of the robot. With our approach, we can use $P(Dyn)$ as a weight in the ICP algorithm in order to not trust dynamic points for the computation of the transformation between the point clouds. This way, we were able to solve the issue of localization drifting due to the dynamic points, by providing both a more precise and cleaner map of the environment. This is consistent to what Burgard et al. [4] have already demonstrated in their 2D experiments.

B. Velocity Estimation

In this experiment, we aim at assessing the capability of the system to estimate different velocities while minimizing the noise induced by the dynamic element segmentation and the localization. The experiment was conducted in a controlled environment consisting of a remote street without traffic. This allowed us to add only one moving obstacle at a time. First, we drove the robot along the street in absence of any dynamic elements, yielding a 170 m wide 3D map of the static part of the environment. Fig. 8 shows an aerial view of the test area, the 3D reconstruction of the street, and the zone where we moved objects. We parked the robot at the indicated position (in green), and then let different dynamic objects pass by in linear motion. We tested two kinds of dynamic objects: a pedestrian and a minibus. The pedestrian was asked to cross the scene once by *Walking* and once by *Jogging*. The driver of the minibus was asked to drive at three different speeds: the lowest speed the vehicle could go (approximately 2 m/s, *Slow*), at 5.5 m/s (*Medium*) and at 11 m/s (*Fast*).

Fig. 9 shows the resulting velocity estimation of our system. The median values of the pedestrian's estimated speed were 1.7 m/s and 4.1 m/s for *Walking* and *Jogging*, respectively. In the experiments with the minibus, the speed of the vehicle could be roughly controlled using the car's speedometer. The velocities listed above represent the *maximum* values reached in each of the three runs. However, the acceleration phase was included in the test track, and is therefore in the estimation. This explains why the estimated median values—1.9 m/s (*Slow*), 4.6 m/s (*Medium*), and 8.1 m/s (*Fast*)—are considerably lower than the targeted speeds. One can also observe that velocities can be estimated at up to 22 m from the sensor.

C. Applications

Finally, we conducted two experiments in a dynamic urban environment. The aim was to demonstrate the performance of our system in real world scenarios, and to give an idea of the range of possible applications. In the first experiment, we drove the robot three times along a 1.3 km long route in the city of Zurich, spread over seven months (March 12, May 23 and September 9, 2013). The first pass was used as the exploration phase, with all subsequent passes building upon the prior map. Fig. 10 shows the results of the survey; the bottom graph mapped the count of dynamic elements that were removed to produce the static map. This graph can be used to identify zones of interest. Two construction sites, that partially occupied the streets, are marked with (1) in red. Marked with (2), is a very large tree (i.e., 1 m diameter trunk) that has been chopped between March and May. Finally, the zone marked with (3) is a busy intersection with cars, trucks, bike paths, trams and many pedestrians (see Fig. 1). The experiment shows that our system can be employed to extract zones that are potentially dangerous to navigate (i.e., places that contain plenty of dynamic objects) or that exhibit large seasonal changes.

The second experiment was conducted in front of the main building of ETH Zurich. It took place during the *information day*, which meant that many young students gathered in the streets, with sometimes as many as 15 persons in the vicinity of the robot. The main street consists of two large sidewalks, two lines for cars, and two lines for trams. The robot surveyed the area twice within 20 minutes, each time

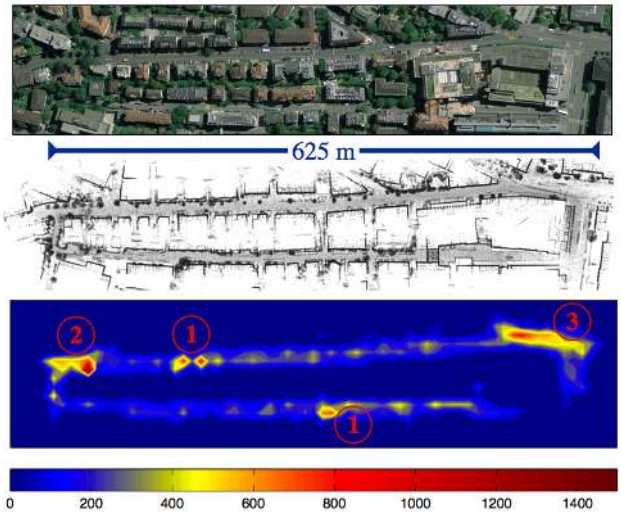

Fig. 10. Long range survey over a 1.3 km long path. The environment was monitored over a period of seven months. *Top*: Aerial view of surveyed area. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Middle*: 3D reconstruction after dynamic elements removal. *Bottom*: Occurrence of dynamic elements. The graph highlights the position of (1) construction sites, (2) a large tree and (3) a busy street intersection. Color represent to number of dynamic points over a cell size of 10 m.

driving on the sidewalks on both sides of the street. Fig. 11 presents the results of the experiment. The two lower graphs show the extracted dynamic objects over the course of the survey, with their estimated speed and direction of motion. In the speed graph (left), blue corresponds to the range of typical walking speeds of pedestrians. The sidewalks and the pedestrian crossings (the latter marked with red arrows) can be clearly identified by looking at the blue objects. Furthermore, there are two lines of faster objects (yellow to red), which designate the car lanes. Note that the velocities are lower in the vicinity of the pedestrian crossings, which comes from the fact that drivers stop to let people cross the street. In the orientation graph (right), the two main directions of the cars are clearly visible. On the sidewalks the situation is naturally more chaotic, as pedestrians do not walk on distinct lanes. Trams were less detected because their speed can only be detected if the robot sees the front or the rear of the wagons. Otherwise, they look like large walls appearing and disappearing from the laser perspective.

The result of this experiment is a first, yet significant, step towards automatic road graph extraction: our system can correctly identify regions of low speed (sidewalks and pedestrian crossings) and road lanes, including the direction of traffic. Enough data to identify crowd and traffic behaviors were collected only by surveying the environment twice and in continuous motion. The velocity extraction is robust enough to estimate a significant number of measurements, even when exploring the environment for the first time.

D. Computation Time

All computations were realized on a single laptop with a four-core Intel Core i7 and 4 GB of RAM. As explained earlier with Fig. 2, all modules run at different speeds. The registration module runs between 6 and 11 Hz by down-sampling the input points and using the wheel odometry as prior alignment. The map maintenance, including the concatenation of the new information and identification of dynamic elements, runs in average at 2 Hz, even with maps as large as 600'000 for the 1.3 km long survey. The computation time for the velocity estimation depends on the number of dynamic elements in the scan. In average, it is 0.03 s (≈30 Hz) in the single dynamic object experiment. The laser sensor produces scans at 11 Hz, which means that we can follow the sensor rate, with some margin. However, a typical pedestrian would only move by 15 cm between two scans. This poses a problem for the speed estimation, as the velocity vectors become noisy at very small distances. On the other hand, if the object is too fast, the assumption of the closest point will fail. As a compromise, we slowed down the data rate to 8 Hz, which we found to be a good compromise to handle velocities from 1.5 to 10 m/s.

VI. Conclusion and Future Work

Our paper presents an online approach for computing both the probability of a 3D point to be dynamic or static and the velocities of dynamic points. Based on 3D point clouds as a sparse representation, we use a Bayesian model for assessing whether a point is dynamic and static, which leads to a cleaner map and a better localization. We also use dual non-rigid ICP to simultaneously compute the velocity of all dynamic points. This approach is cluster- and model-free, with only a weak smoothness assumption, and is able to successfully evaluate the velocity of dynamic objects.

We have shown that identifying dynamic objects produces an accurate map of the static scene geometry. This is especially important for difficult path-planning tasks in highly cluttered 3D environments. In both cases, it is detrimental for dynamic objects to be wrongly classified as obstacles, as it could invalidate the only feasible path to a goal.

We have also shown that the integration of the velocity information in the map reveals the main characteristics of traffic or pedestrian flow. This could be used to extract higher-level semantic information, which is necessary for more advanced path-planning techniques in dynamic environments. For example, a lot of advanced collision avoidance techniques require the ability to predict the trajectory of all other dynamic objects. This is usually not feasible, unless those dynamic objects are other robots tracked by an external localization system [17], restraining the use of such techniques. Our approach provides velocity estimates for dynamic obstacles as well as the aggregated knowledge of past observations, which may be used to predict future behavior, like the possibility that cars slow down at the pedestrian crossing. In future work, we would like to use the time history of dynamic objects to build probabilistic models of motions within the map, as the 2D camera model in [18], and then use these models to perform safe real-time motion planning and navigation in crowded urban settings. We will also evaluate a larger range of parameters to define their impact on the robustness of the system and give more

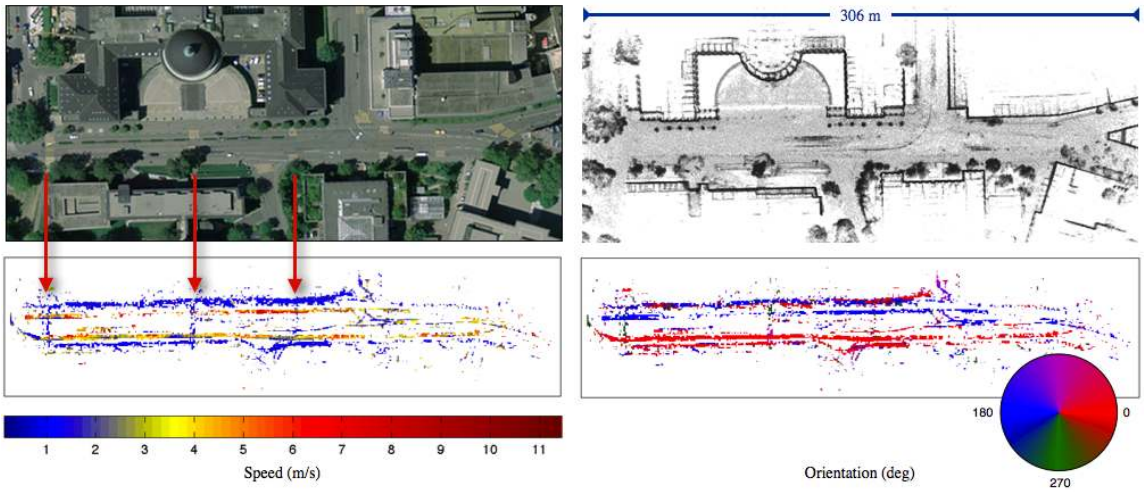

Fig. 11. Extraction of velocity information at a global scale. *Top left*: Aerial view of the street in front of ETH Zurich. *Source*: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000. *Top right*: 3D reconstruction after dynamic element removal. *Bottom left*: Average speed of the moving objects. *Bottom right*: Average orientation of the moving objects. The red arrows highlight the pedestrian crossings.

insides on how others could tune them given their specific applications. We believe that with our approach, we have brought advanced path-planning techniques closer to field robotics.

VII. Acknowledgments

This work was supported by the EU FP7 IP projects Natural Human-Robot Cooperation in Dynamic Environments (ICT-247870), TRADR (ICT-609763), the armasuisse S+T UGV research program, and the EU FP7 2007-2013 Program, Challenge 2, Cognitive Systems, Interaction, Robotics, under grant agreement No 269916, V-Charge. The authors are grateful to M. Rufli, U. Schwesinger and M.-È. Garneau for useful comments.

References

[1] R. Kümmerle, M. Ruhnke, B. Steder, C. Stachniss, and W. Burgard, "A navigation system for robots operating in crowded urban environments," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[2] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. T. S. Rottmann, B. Li, B. Schmidt, T. N. Nguyen, E. Cardarelli, S. Cattani, S. Brüning, S. Horstmann, M. Stellmacher, H. Mielenz, K. Köser, M. Beermann, C. Häne, L. Heng, G. H. Lee, F. Fraundorfer, R. Iser, R. Triebel, I. Posner, P. Newman, L. Wolf, M. Pollefeys, S. Brosig, J. Effertz, C. Pradalier, and R. Siegwart, "Toward Automated Driving in Cities using Close-to-Market Sensors, an Overview of the V-Charge Project," in *IEEE Intelligent Vehicles Symposium (IV)*, Gold Coast, Australia, 23–26 June 2013, pp. 809–816.

[3] 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.

[4] W. Burgard, C. Stachniss, and D. Hhnel, "Mobile robot map learning from range data in dynamic environments," in *Autonomous Navigation in Dynamic Environments*, ser. Springer Tracts in Advanced Robotics, C. Laugier and R. Chatila, Eds. Springer Berlin Heidelberg, 2007, vol. 35, pp. 3–28.

[5] 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 *Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, 2010 (ICRA)*, 2010.

[6] A. Aijazi, P. Checchin, and L. Trassoudaine, "Automatic Removal of Imperfections and Change Detection for Accurate 3D Urban Cartography by Classification and Incremental Updating," *Remote Sensing*, vol. 5, no. 8, pp. 3701–3728, Aug. 2013.

[7] J. Ryde and N. Hillier, "Alignment and 3D scene change detection for segmentation in autonomous earth moving," in *Robotics and Automation, 2011. Proceedings of the IEEE International Conference on*, 2011, pp. 1484–1490.

[8] R. Kaestner, J. Maye, Y. Pilat, and R. Siegwart, "Generative object detection and tracking in 3D range data," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 3075–3081.

[9] J. Shackleton, B. VanVoorst, and J. Hesch, "Tracking People with a 360-Degree Lidar," in *Advanced Video and Signal Based Surveillance (AVSS), 2010 Seventh IEEE International Conference on*, 2010, pp. 420–426.

[10] P. Anderson-Sprecher, R. Simmons, and D. Huber, "Background subtraction and accessibility analysis in evidence grids," in *Robotics and Automation, 2011. Proceedings of the IEEE International Conference on*, 2011, pp. 3104–3110.

[11] C. C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, "Simultaneous Localization, Mapping and Moving Object Tracking," *The International Journal of Robotics Research*, vol. 26, no. 9, pp. 889–916, Sept. 2007.

[12] F. Moosmann and T. Fraichard, "Motion estimation from range images in dynamic outdoor scenes," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 142–147.

[13] J. Elseberg, S. Magnenat, R. Siegwart, and A. Nüchter, "Comparison of nearest-neighbor-search strategies and implementations for efficient shape registration," *Journal of Software Engineering for Robotics*, vol. 3, no. 1, pp. 2–12, Mar. 2012.

[14] M. Rufli, J. Alonso-Mora, and R. Siegwart, "Reciprocal Collision Avoidance With Motion Continuity Constraints," *Robotics, IEEE Transactions on*, vol. 29, no. 4, pp. 899–912, 2013.

[15] J. Feldmar and N. Ayache, "Locally affine registration of free-form surfaces," in *Computer Vision and Pattern Recognition, 1994. Proceedings of the IEEE Computer Society Conference on*, 1994, pp. 496–501.

[16] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing ICP Variants on Real-World Data Sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, Feb. 2013.

[17] J. Alonso-Mora, A. Breitenmoser, M. Rufli, R. Siegwart, and P. Beardsley, "Image and animation display with multiple mobile robots," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 753–773, May 2012.

[18] K. Kim, D. Lee, and I. Essa, "Gaussian process regression flow for analysis of motion trajectories," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*. IEEE Computer Society, November 2011.