Plane-based Scan Registration with Moving Vehicles Exclusion

Chongyang Wei[a], Ruili Wang[*b], Tao Wu[a],
Tongtong Chen[a],Yuqiang Fang[a], Hao Fu[a]
[a]*College of Mechatronic Engineering and Automation,*
[a]*National University of Defense Technology, Hunan, P.R. China.*
[a]{*cyzq3566, wtt09.cs, chentongtong5208, yqfang.cs, fuhao927*}*@gmail.com*
[b]*School of Engineering and Advanced Technology,*
[b]*Massey University, Auckland, New Zealand.*
[b]*r.wang@massey.ac.nz.*

Abstract

Moving vehicles have a considerable negative effect on the accuracy of scan registration and lidar odometry. To remove the negative effect, we propose an extended 2D virtual scan to obtain all moving objects in the sensing range of lidar by a scan differencing operation between two consecutive scans. The dynamic objects' poses are estimated with our proposed likelihood-field-based vehicle measurement model and the motion evidence is utilized to classify the objects as moving vehicles or not. In this way, the moving/dynamic vehicles are detected and the points hitting them are removed. The remaining points are then taken as an input into the alignment.

In the registration, we adjust the raw distorted points by modelling the lidar motion as the constant angular and linear velocities within a scan interval, and then exploit the probabilistic framework to model the local plane structure of the matched feature points instead of the original point-to-point mode. The transform is achieved by the combination of coarse motion estimation and fine batch adjustment. The algorithm has been validated by a large set of qualitative tests on our collected point clouds and quantitative comparisons with the excellent methods on the public Karlsruhe Institute of Technology and Toyota Technological Institute (KITTI) odometry datasets.

[*]Corresponding author

Keywords: Scan registration, Moving vehicle detection,
Likelihood-field-based model, Plane-based criterion, Autonomous vehicles.

1. Introduction

Data registration is a hot research topic in several key technologies of autonomous driving, e.g., visuar/lidar odometry, scene understanding and common simultaneous localization and mapping (SLAM). The algorithms of alignment rely on the static point clouds in overlapping regions to optimize the rigid transform between two consecutive frames or between the frame and accumulated map. Owing to lidar's insensitivity of lighting changes, and high frequency depth measurements where errors are relatively constant irrespective of the distance travelled, different kinds of lidars [1, 2, 3], especially, Velodyne HDL-64E scanner [4, 5], have been widely applied in the field of self-driving.

Moving/dynamic objects have great effects on the accuracy of scan registration, and the accumulated incremental error over time is bound to drift in the following lidar odometry. One natural idea to reduce the drift is to remove moving objects in the alignment. This refers to the classical problem of detection and tracking of moving objects (DATMO). There are some differences between DATMO in the aspects of the safe autonomous navigation (marked as General DATMO) and the alignment or odometry (marked as Aligned DATMO).

1): Processing range. The critical threat for safe autonomous navigation comes from the closest obstacles which possibly induce potential collision, whereas the further away obstacles have little or even no effects on driving safely. Thus many approaches in the general DATMO only record and process the closest obstacles [6, 7], which are reasonable and efficient. However, the registration is expected to use the total point cloud [8, 9, 10] or the features extracted in the whole scene [11] to match, and thus those points hitting the more distant obstacles must be taken into consideration. This demands that the aligned DATMO methods extend the processing range to the total scene.

2): Processing scan number. To achieve a robust and reliable output, the general DATMO methods [7, 12] are first to detect the moving objects in the current scan and then to gradually validate by tracking. The process will take several scan periods or even more to accomplish [13]. This strategy is not suitable for our application. Moving objects generate negative effects on

scan registration before they are distinguished through sequential tracking and unfortunately, the error cannot be removed once it appears. Thus for registration or odometry, what it really needs is to find moving objects and remove them within just two scans.

3): Processing objects. To avoid collision, the general DATMO approaches [6, 14, 15] need to process common moving objects such as vehicles and pedestrians, whereas the aligned methods [16] are usually only interested in the moving vehicles, either because the velocity of the pedestrian is relatively low (about $1 \sim 2$ m/s), which may be covered by the noise from the measurement and model, or because the number of points hitting the pedestrian is relatively small due to their small volumes. In this paper, we only consider the removal of moving vehicles at present.

4): Tolerant recall. The general DATMO algorithm has a high demand for recall because any false negative can possibly induce danger for the autonomous ground vehicle. By contrast, misjudge stationary objects as moving ones on account of not tracking and not using those points falling on them will have little or no effects on the matching performance. This is because the misjudged points can include only a very small portion in the total abundance of points from Velodyne HDL-64E.

Based on the four differences discussed above, in this paper, we propose an extended 2D virtual scan to handle all moving objects in the scene, which overcomes the drawback of the original virtual scan [7] only recording the closest obstacles. For each moving object, its pose is estimated with our novel likelihood-field-based vehicle measurement model, and the motion evidence proposed in [7] is utilized to validate the moving object to be the vehicle or not, in just two consecutive scans. Because the vehicles are restricted to moving on the 2D ground surface, some vehicles, visible in the 3D space, could be occluded in the 2D space. Our proposed algorithm can deal with the pose estimation of all the moving vehicles in the whole scene, while [7] and its variant [17] cannot. The detected moving vehicles are removed and the remaining point cloud is taken as an input for our novel scan registration algorithm.

The readings collected by the Velodyne HDL-64E are distorted when the lidar itself is moving. In this paper, we model the lidar motion as the constant angular and linear velocities during a scan interval, and compensate the raw distorted point cloud by linearly interpolating the initial pose transform (provided by the wheel encodes). The lidar collects over 130,000 3D points per scan. To reduce the computational complexity, we use the same

approach as in reference [11] to extract feature points located on sharp edges and planar surfaces, and match them to the points projected in edge line and planar surface patches, respectively. However, the point-to-point metric criterion does not consider the geometric attribution of the local neighborhood of the selected point. Based on this, we exploit a probabilistic framework [9] to model local plane structure which ensures a high confidence in the normal direction. The estimation of pose transform is achieved by the excellent coarse-to-fine strategy [11]. Integrating with the removal of all moving vehicles, our aligned method can obtain higher performance in comparision with the excellent methods [9, 11, 18].

There are two main contributions in our paper: one is the likelihood-field-based vehicle measurement model, which can deal with the pose estimation of the vehicle; the other is the plane-based metric criterion in the alignment process, which can match well the lidars feature points. The first contribution is the more significant one.

The rest of the paper is organized as follows. In the next Section, a survey of related works is summarized. After that the moving vehicle detection algorithm in two scans is presented in detail in Section 3. The novel scan registration algorithm is described in Section 4. The experimental results and analysis are shown in Section 5. Finally, Section 6 gives the conclusions.

2. Related works

Moving vehicles have a great negative effect on the accuracy of scan registration and lidar odometry. To remove this effect, the detection of moving vehicles is needed. Current methods generally can be divided into two categories: feature-based methods [19, 20, 21, 22, 23] and model-based methods [7, 13, 17].

In [19], the radar data was accumulated in an occupancy grid where the objects were detected. The candidate object was validated as the parked vehicle using four features with two random forest classifiers. Reference [20] proposed three geometric features for the finely segmented 3D object to classify the vehicles with the kernel support vector machine (SVM). Reference [21] exploited the distance criterion to cluster the obstacle points and fitted the cluster in a cube bounding box. The length, width and height were utilized to classify the cube box into a vehicle or not. In [22], the contour information, i.e. the ratios of length to height and width to height of the object bounding box, was employed to detect the vehicles. The RANSAC approach

was implemented in [23] to fit a straight-edge feature with "L-shape" to detect the vehicle. The common flowchart of feature-based methods consists of segmentation, cluster similarity, feature detection and model fitting. The main drawback lies in that, with noisy and cluttered data, there are many ambiguities in fitting a vehicle.

The model-based methods generally exploit the cuboid or rectangle to fit a vehicle. Reference [17] proposed a view-dependent adaptive matched filter algorithm to obtain a vehicle pose. It took self-occlusion into account and used four rectangles to represent two visible sides, the interior and the exterior regions of a vehicle, respectively. A gradient-descent-based optimization was used to maximize the integrals over the four rectangles. In [7], the pose of a moving vehicle was estimated with a Bayesian filter and every particle was weighted with the rectangular measurement model, then the motion evidence was used to quickly prune false positives caused by noise. Reference [13] extended the work in [7] to overcome the unavailability of road network information and introduced geometric and temporal cues to reduce the sampling range for improvements to efficiency. The geometric model in [7] and its variant [13] can only detect the closest vehicles. This model is unsuitable to our scan registration that requires all moving vehicles to be removed. We focus on this problem in this paper.

In scan registration, selecting a small quantity of basic elements from the richness of points and using a reliable criterion to match is an effective way to reduce computational complexity and improve robustness. One popular idea is to extract reliable features to align. Various features have been applied, i.e., the key point [24], the spin image [25], FPFH [26], NARF [27], and the rectangle-based histogram [28]. The feature-based algorithms can deal with scan pairs with partial overlap and large offset, but take a large amount of time on the computation of feature descriptors and lack proper strategies to remove the incorrectly matched features. Recently, reference [11] proposed a simple and efficient method to select points located on distinct edge lines and planar surfaces according to points' curvatures. In this way it can reduce the number of matched pairs to a lower level.

The golden standard for precise alignment was the iterative closest point algorithm (ICP) [18], which attempted to optimize the transform in the point-to-point way. One common assumption is that a good original guess is available, otherwise it is easy to get trapped in local minima. Reference [9] proposed a novel criterion to describe the point pairs in a probabilistic form and modelled a locally planar surface structure from both scans. A

major problem of point-based methods is the expensive computation of the nearest-neighbour correspondences [29]. Considering the shortcoming, three-dimensional normal distributions transform (3D-NDT) was presented in [10] and later expanded from point-to-distribution to distribution-to-distribution [30]. NDT can also get trapped in local minima in the absence of a good initial value, but provides a wider convergence basin [31]. These algorithms only exploit the lidar points of the consecutive two scans and can be regarded as a scan-to-scan alignment. Reference [1, 2] exploited sequential scans to recover the lidar's trajectory by a batch optimization between the scan and the accumulated map. The advantage lies in that the current pose of lidar depends not only on the motion estimation from the last scan but also relies on the accumulated historical information.

3. Moving vehicle detection

In this section, we propose a novel algorithm to remove all the moving vehicles in a scene. Firstly, we propose an extended 2D virtual scan to deal with all moving objects in the scene, which overcomes the drawback of the traditional virtual scan [7, 17] recording only the closest obstacles. Secondly, we combine the advantages of the likelihood-field-based vehicle measurement model and the scaling series particle filter to present a novel vehicle pose estimation approach. Our approach can deal with pose estimation of all the moving vehicles in the whole scene, while the reference [7] and its variant [17] cannot. Compared with the reference [17] which uses a gradient-descent-based optimization, our method is not relative to the initial values and it can not get trapped into local minima. Finally, the moving vehicles are detected and removed after passing the validation of the motion evidence [7] in two consecutive scans.

3.1. Extended virtual scan

The Velodyne HDL-64E spins at high frame ratio (10 HZ) and collects over 1.3 million points per second. Given such abundant data, how to efficiently process the readings to produce a suitable representation tailored for the detection of moving vehicle has become a challenge. Petrovskaya [7] proposed adopting the virtual scan to project 3D points onto a 2D ground surface, which subdivides the total space around the chosen origin into evenly angular grid cells with polar coordinates. In each grid cell it only records the range to the closest obstacle. Hence the space from the origin up to

the recorded range is free, the recorded position is occupied, and beyond the position is occluded. Every cone of an angular grid cell from the origin appears as a 2D surface laser ray. The advantage is that it saves a lot of computational resources by converting the 3D points into the 2D space, as the vehicles are restricted to moving on the 2D ground surface; meanwhile, it is convenient to fuse other laser range finders [12]. The drawback is that the more distant objects which are occluded by the closest obstacles are ignored, which is evidently unsuitable to our aligned application.

To improve the efficiency of use of raw data some pre-processes are desirable. For the purpose of vehicle detection, the ground readings and hanging obstacles are uninteresting, we follow the ground estimation approach [32] to remove them. The remaining point clouds are clustered into the segments following [33]. To deal with the raw distorted data generated due to ego-motion, we adopt the approach of Subsection 4.1 to adjust as if they are received at the same time stamp.

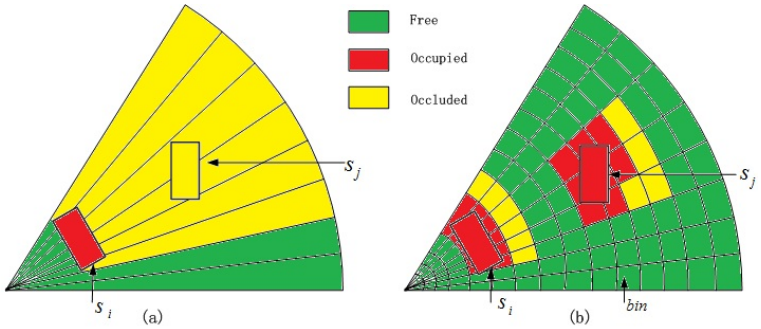


Figure 1: The initial virtual scan [7] and our extended virtual scan. (Best viewed in colour).

In this paper, we propose an extended 2D virtual scan which can reserve the information of all obstacles in the scene. Based on the grid map in polar coordinates, we subdivide the cone of each angular grid cell into 400 bins with the 0.2 meter resolution as shown in Fig. 1(b). The clustered segments are then projected into the extended virtual scan. For each segment s_i we record the nearest and farthest bin which falls into each grid cell l, marked as $b_{i,l}^{min}$, $b_{i,l}^{max}$, respectively. Hence in the lth cone the space from $b_{i,l}^{min}$ to $b_{i,l}^{max}$ is marked as occupied in red color, the space from $b_{i,l}^{max}$ to $(b_{i,l}^{max} + \lambda)$ is recorded as occluded in yellow color, the remaining is free in green color as shown in Fig. 1(b). Here λ denotes the self-occluded region of the segment (potential

vehicle) and its vicinity, which is decided with the segment's width and its vicinity. By this means the farther segment s_j occluded by the nearer s_i can also be detected because of the new definition of the attribution of bins.

For the detection of moving objects knowing the changes which take place in the scene during two consecutive scans is desirable. Through projecting the two consecutive scans into the extended virtual scan in the current lidar coordinate system with the motion compensation provided by the combination of GPS and an inertial navigation system (INS)(if available) or the local wheel encodes, the changes can be easily detected by checking the changed attribution of bins. This detection scales linearly in the size of the virtual scan and only needs to be carried out once per scan. Fig. 2 shows the results of a scan differencing operation in our extended virtual scan with the red points representing new obstacles, the blue points representing old obstacles, and the white points denotes that the attributions of the regions are unchanged.

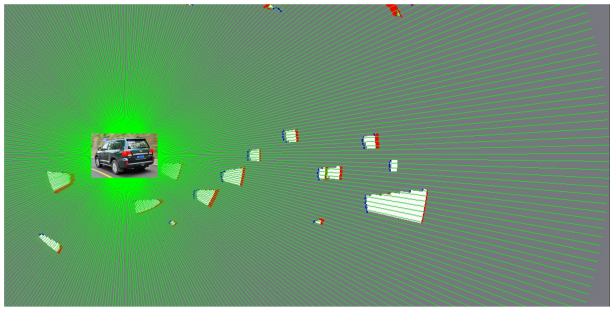


Figure 2: The result of a scan differencing operation in two scans shown in coloured points. The green lines are virtual rays, the red points represent the new obstacles which are occupied in the current scan but free in the last scan, while the blue points are old obstacles which are free in the current scan but occupied in the last scan, and the white points denote that the occupied attribution of the regions remained unchanged. (Best viewed in colour.)

3.2. Likelihood-field-based measurement model

In this paper, we assume the vehicle typically has a rectangular shape (viewed from above) on the 2D ground surface shown in Fig. 3 and follow reference [17] to model the self-occluded vehicle and varying sampling surface due to different incident angles. The number of points falling on any non-occluded surface region is approximately proportional to the incident angle

of lidar scan beam. This shows that a majority of points will fall onto the two visible vehicle sides and their weight can be represented with the cosine of the angle between the surface normal and the incident ray. There are a minority of points falling in the interior region of the rectangular box, with different distribution according to special types of vehicles, and for simplicity we assume they follow a uniform distribution in this area and assign a proper positive weight on each point. Ideally there will be no points in the vehicle's vicinity because vehicles are spatially separated from other obstacles in the scene, hence we assign a negative weight to this region.

For every moving object, a vehicle measurement model is fitted to the rectangular box, with all 3D points hitting its surface instead of just exploiting one point in each angular grid cell [7]. The model takes a cluster of measurements Z=$\{q_1, q_2, ..., q_n\}$, the constant width W, length L as the inputs and output the 2D position (x, y) and orientation θ. We follow the reference [17] to model the probability density of the local vehicle surface with a 2D Gaussian distribution $g(x, y)$ centered at the lidar point,

$$g(x, y) = \frac{1}{2\pi\sigma^2} exp\{-\frac{(x - x_i)^2}{2\sigma^2}\} exp\{-\frac{(y - y_i)^2}{2\sigma^2}\}, \tag{1}$$

where the variance σ^2 of this Gaussian includes both the uncertainty of measurement noise inherent in the lidar and the variability of the target surface due to different incident angles. It is an experimental value obtained by a series of tests.

Our measurement model is shown in Fig. 3. The green and blue rectangular regions denote the visible length edge and width edge surfaces of the vehicle model, while the grey and yellow ones are the interior and exterior regions of the vehicle, respectively. For every edge of the rectangle, the line orientation from its centre to the origin O_L of the sensor is defined as the incident vector of the ray, and the orientation which is perpendicular to the edge and pointing outside the vehicle is defined as a norm vector. Obviously, if the angle between incident vector and norm vector is less than $\pi/2$ the edge is visible, and there are, at maximum, two edges visible at a time as shown in Fig. 3(a).

In this paper, we assume that every measurement is conditionally independent of others given vehicle pose. Thus, the likelihood factors of the vehicle measurement model containing n readings in Z can be computed as

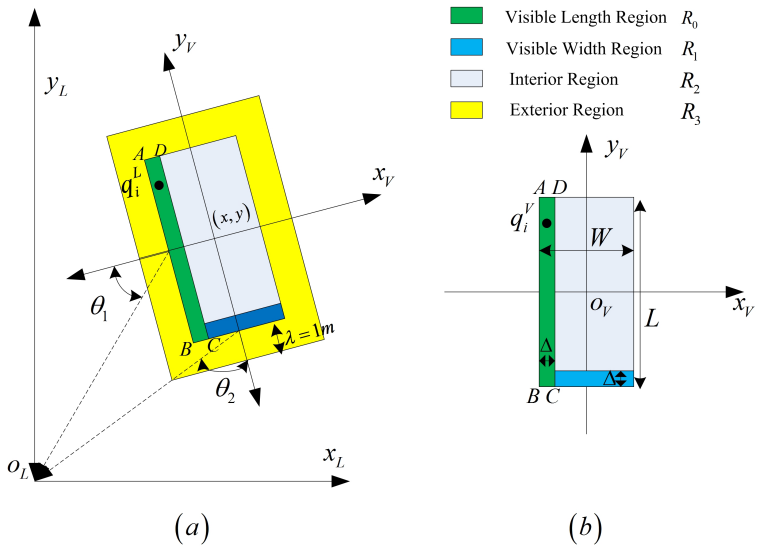


Figure 3: The view-dependent measurement likelihood computations. (a) shows the four geometric rectangular regions involved in the likelihood computation. The yellow, green, blue and grey regions represent the exterior, visible length side, visible width side and interior of the vehicle, respectively. θ denotes the angle between incident vector and norm vector. (b) shows the rotated vehicle coordinate system with its axes parallel to lidar coordinate axes. The length and width of the model are constants, and the width of the visible surface is set to Δ. (Best viewed in colour.)

follows:

$$p(Z|X) = \prod_{i=1}^{n} p(q_i|X). \tag{2}$$

Each reading q_i's likelihood is modelled as the exponent of the sum of the integrals over the four rectangular boxes in the vehicle coordinate system $o_V x_V y_V$,

$$p(q_i|X) = \eta_i exp(\alpha \sum_{j=1}^{4} c_j I(q_i^V, R_j)), \tag{3}$$

where q_i^V is obtained by transforming the measurement q_i in the lidar coordinate system $o_L x_L y_L$ to the vehicle coordinate system $o_V x_V y_V$; η_i is the normalized constant; R_0, R_1, R_2 and R_3 denote the visible length region, visible width region, interior region, exterior region of the vehicle measurement model as shown in Fig. 3, respectively; c_0, c_1, c_2 and c_3 are respectively the corresponding integral weights over the corresponding rectangular boxes with

c_0 and c_1 being selected according to the cosine of the view angle, c_2 being set to a positive value, c_3 being set to a negative value; $\alpha = 1/\sqrt{\sum_{k=0}^{3} \iint_{R_k} c_k^2 dxdy}$ is the normalized factor; $I(q_i^V, R_j)$ depicts the integral of 2D Gaussian distribution of point q_i^V over the rectangular box R_j in the vehicle coordinate system.

The computational difficulty of the last equation lies in that each measurement must be integrated over the four rectangular regions. Appendix.1 gives in detail the reasoning process of the integral over the rectangular region R_0. Similarly, the integrals $I(q_i^V, R_1)$, $I(q_i^V, R_2)$ and $I(q_i^V, R_3)$ can be computed in the same fashion as above. We note that the rectangular edges must be parallel to the coordinate axes, hence it is desirable to transform the measurement points in the lidar coordinate system to ones in the vehicle coordinate system. In this way, we can estimate the vehicle pose with our novel likelihood field model. In contrast to [7, 17], the combination of our proposed extended 2D virtual scan and the likelihood-field-based vehicle measurement model makes our algorithm able to deal with the pose estimation of all the vehicles in the whole scene.

3.3. Scaling series particle filter

The first step of vehicle pose estimation is to fit the geometric model to a virtual scan under conditions of large uncertainty: several meters in position and 180 degrees in direction. Generally, performing the importance sampling over a 3-dimensional space directly will make the number of particles too large and brings a great computational burden. Reference [7] proposed a good strategy to deal with this situation, which iteratively refines the measurement model from artificially relaxed to realistic as shown in Fig. 4. For pose estimation of a vehicle, we first inflate the normal width Δ of the vehicle visible region by 1 meter in both the interior and exterior of the measurement model, resulting in a relaxed visible width setting of $\Delta + 2$ meters. In this case the rectangular region of the vehicle model expands to fully occupy the yellow free space surrounding the vehicle and a part of the grey interior region as shown in the left image of Fig. 4. With the most relaxed model, some low likelihood weighted particles, located at 1 meter far away from the actual vehicle position, will be easily ruled out. The high likelihood weighted particles indicate a region of less than 1 meter far away from the true vehicle's position. By this means, when the relaxed width of the surface region iteratively reduces from $\Delta + 2$ to Δ, the high likelihood region will

become smaller and smaller and finally arrive at the true vehicle position (the right image of Fig. 4).

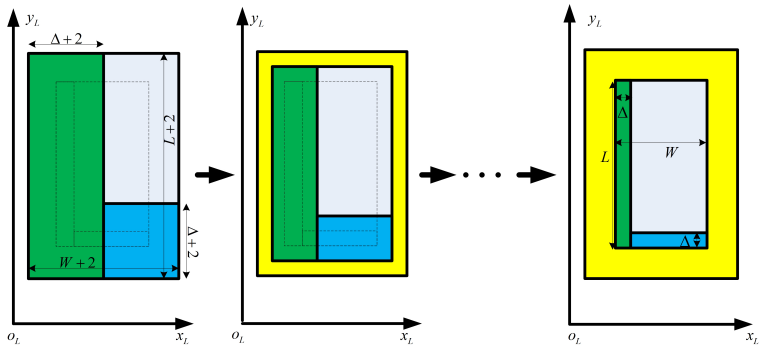


Figure 4: The iterative process of the measurement model. The meanings of four colored rectangular boxes are the same as those ones in Fig. 3. The left image denotes the most relaxed width of the surface region, which gradually reduces to the actual width in the right image. (Best viewed in colour.)

The scaling series particle filter algorithm was firstly proposed in [34] to deal with an accurate tactile localization problem with a large number of parameters to sample in real time. In [34], each particle represents a region of the state space instead of representing a single point of the space. In this paper, we exploit this algorithm to improve the computational efficiency of pose estimation. The formal algorithm listing is shown in Algorithm 1.

The algorithm takes a cluster of 3D measurements $Z=\{q_1, q_2, ..., q_n\}$, the normal width Δ of surface region of the vehicle model, the number of particles M per neighbourhood, the number of iterations N as the inputs, and outputs the optimized vehicle's pose X^*.

The first step is to find the initial values for the particle filter, which exploits the rectangle of the minimal area, obtained by the OpenCV (open computer vision library), to enclose the projection of cluster Z onto the ground surface in line 1, and obtains the 2D centre position $(\bar{C}_x, \bar{C}_y)$ and the orientation of the rectangle $\bar{\theta}$. The vehicle pose sampling space is denoted by V, which is a 3-D ellipsoid sphere constructed with the vehicle position (x, y) and the vehicle orientation θ. V_0 in line 2 is the initial sampling space, with $(\bar{C}_x, \bar{C}_y, \bar{\theta})$ as its center and $(L/2, \theta)$ as its radius in both position and orientation. The scaling factor *zoom* is set to be $1/\sqrt[3]{2}$ to make sure that the volume of the sampling region of every particle in 3D sampling space is halved during scaling.

Algorithm 1: The Scaling Series Particle Filter

Input: $Z = \{q_1, q_2, ..., q_n\}, \Delta, M, N$

Output: $X^* = (x^*, y^*, \theta^*)$;

1: $(\bar{C}_x, \bar{C}_x, \bar{\theta}) =$ minAreaRect(Z);

2: $w = 1.0, \theta = \pi/2, zoom = 1/\sqrt[3]{2}, V_0 = (\bar{C}_x, \bar{C}_x, \bar{\theta})$;

3: for i=1: N do

4: $\bar{\chi}_i =$ Even _ Density _ Sample(V_0, M);

5: $\omega_i =$ Compute_Likelihood_Weights($\bar{\chi}_i, 2w + \Delta, Z$);

6: $\chi_i =$ Prune($\bar{\chi}_i, \omega_i$);

7: $V_i =$ Union_Delta_Neighborhoods(χ_i, w, θ);

8: $w = w * zoom, \theta = \theta * zoom$;

9: end for

10: $\chi =$ Even_Density_Sample(V_N, M);

11: $\omega =$ Compute_Likelihood_Weights(χ, Δ, Z);

12: $X^* =$ Best_Particles(ω);

Lines 3-9 describe the iterative refinements that gradually get close to the actual measurement model. At each iteration i, line 4 draws a particle set with even density by sampling M particles in each ellipsoid neighborhood of V_{i-1}, which is one of the most critical features to deal with the posterior's ambiguity at early stages. We refer the reader to reference [35] for detailed discussion of algorithm features and setting. In line 5, according to the Eq.(2) and Eq.(3), *Compute_Likelihood_Weights* procedure weights the particles set $\bar{\chi}_i$ by gradually reducing the width $2w + \Delta$ of surface region, where w denotes the relaxed width of the surface. In line 6, a threshold is firstly calculated based on the log of the weights. Thus any particle whose weight is less than the threshold is pruned out. For the remaining particles, the function *Union_Delta_Neighborhoods* in line 7 constructs the sub-neighborhoods with their states as the center and (w, θ) as the sample radius of the position and orientation. Line 8 shrinks the sample radius with the scaling factor for the next iteration. After completion of the iterative refinement steps, lines 10 - 12 draw the final particle set χ, weight them again with the actual width of the surface region as an input and select the particle with the highest weight

as the final optimized pose X^*.

3.4. Moving vehicle validation

In this paper, we only utilize two consecutive scans to distinguish moving vehicles from stationary ones without the sequential tracking to validate the motion consistent of candidate objects [7, 13] according to the special demand described in Section 1 in our application.

Firstly, all the moving clusters of the current scan are distinguished with the scan differencing operation in the extend 2D virtual scan in Subsection 3.1; after that, the pose of each cluster is estimated with the scaling series particle filter algorithm in Subsection 3.3, which exploits the likelihood-field-based measurement model in Subsection 3.2 to calculate the weights of particles.

Then, we move backwards in time to find the corresponding moving vehicles in the prior scan. As the velocity of the vehicle is generally less than 35 m/s, for a lidar at a frame rate of 10 HZ, we sample from uniform distribution U(-3.5m, 3.5m) centered at the state of the vehicle in the current scan and weight the particles with our likelihood-field-based model. We deem the highest weighted particle to be the corresponding pose of the vehicle in the prior scan. The motion evidence proposed in [7] is used to rule out the false positives. The vehicles passing the validation are the ones we should remove in the scan registration.

4. Plane-based scan registration

Our novel scan registration algorithm consists of four steps: firstly, the raw distorted point clouds received at different time stamps are rectified by linearly interpolating the pose transform of the moving platform within the scan interval; secondly, two kinds of special points located on the sharp edge lines and planar surface patches are picked out and their correspondences are found by taking advantage of the identities of beams [11]; then, the plane-based metric criterion is adopted in the optimized processing to provide a high confidence in the norm orientation [9]; finally, the framework integrating the coarse scan-to-scan motion estimation with the fine scan-to-map local bundle adjustment [11] is adopted to solve the optimized pose transform.

4.1. Rectification of Raw Points

With its full 360^o horizontal field of view (FOV) and 26.8^o vertical FOV, the Velodyne HDL-64E can describe the surrounding environment around the vehicle in the range of 120 meters. The lidar unit spins at 10 HZ and every scan (lasting for 0.1 s) is generally regarded as one period. We assume that the origin of the lidar coordinate system lies in the center of its position with x-axis pointing directly forward, and z-axis pointing upward. Formally, a scan is represented by point cloud $\tilde{P}$=$\{\tilde{p}_1, \tilde{p}_2, ..., \tilde{p}_n\}$, where $\tilde{p}_i$=$\{\tilde{x}_i, \tilde{y}_i, \tilde{z}_i,\}^T$ denotes its Euclidean coordinates with respect to its local coordinate system.

As the scanner rotates to collect the data along with the moving platform, each new reading is from a new reference point due to ego-motion of the platform. Therefore, the point set can be locally distorted during scan interval. The situation is especially distinct when the vehicle is driving at high velocity or making a turn. The distorted scan will have a negative effect on the performance of scan registration and consistent mapping. To handle the problem, Nüchter et al. [36] stopped the platform to obtain the undistorted 3D environment points in the processing of each scan. This is obviously not appropriate for our intelligent vehicle's driving. one has to cope with the vehicle's movement as the scanning time interval cannot be neglected. In this paper, we model the vehicle motion as the constant angular and linear velocities during a scan. This allows us to compensate the raw points which are received at different time stamps by linearly interpolating the pose transform within the scan interval. Let t be the current time stamp, t_{k-1} be the starting time stamp of the kth scan, t_k be the end time stamp. T_k denotes the pose transform between $[t_{k-1}, t_k]$. T_k may come from the global position system such as GPS/INS or the local wheel encodes. T_k depicts the rigid motion of the lidar in 6 degree of freedom and T_k =$[\alpha_x, \alpha_y, \alpha_z, t_x, t_y, t_z]^T$, where α_x, α_y, α_z are rotation angles along the x-axis, y-axis and z-axis of lidar coordinate system, respectively, t_x, t_y, t_z are corresponding translations. Given a point $\tilde{p}_k^i$ in the kth scan, let t_i be its time stamp, and T_k^i be the pose transform between $[t_{k-1}, t_i]$. T_k^i can be computed by linear interpolation of T_k,

$$T_k^i = \frac{t_i - t_{k-1}}{t_k - t_{k-1}} \cdot T_k. \tag{4}$$

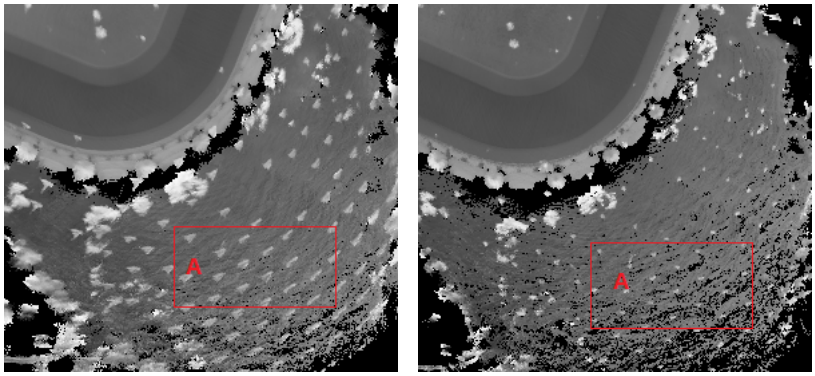
The undistorted point p_k^i after compensating $\tilde{p}_k^i$ using the transform in Eq.(4) can be derived by:

$$p_k^i = R \cdot \tilde{p}_k^i + T_k^i(4:6), \tag{5}$$

where p_k^i denotes the ith adjusted point in the kth scan, $T_k^i(a{:}b)$ is the ath to bth items of T_k^i and R is the rotation matrix defined by rotating around the Euler angles in the right-hand rule,

$$R = R\alpha_y \cdot R\alpha_x \cdot R\alpha_z. \tag{6}$$

In this way, we compensate every raw point to form a consistent undistorted point cloud. To observe intuitively, we project the 3D point clouds which are accumulated by the state-of-the-art GraphSLAM [37] algorithm onto a 2D ground surface. The intensity of the generated image is the average height of the points falling in the 2D grid cell with 20 cm's resolution. Fig. 5 shows the generated images with distorted initial point clouds and adjusted ones as respective input when the vehicle goes through a turning with many trees around the road on the campus. We can see that the projections of trees in the red rectangular area A in the left image are elongated and then produces the ghost image. By contrast, the projections in the red rectangular area A in the right image exhibit the real size of trees and no ghosting.



(a) (b)

Figure 5: The 2D ground maps generated with raw distorted points and the undistorted ones after our adjustment. The projections of the objects are elongated in (a). After our rectification, the undistorted projections show the real sizes in (b).

4.2. Finding feature points and correspondence

In this paper, we explore the way [11] to select two kinds of special points located on sharp edges and planar surface patches. Let $p^i_{k,l}$ be the ith point from the lth beam in kth scan, S be the set of consecutive points of $p^i_{k,l}$ in the lth beam and $|S|$ be the number of the points in the set. Define a variable c to evaluate the curvature of the local neighbour in the lth beam:

$$c = \frac{1}{|S| \cdot \|p^i_{k,l}\|} \sum_{j \in S, j \neq i} \|(p^i_{k,l} - p^j_{k,l})\|. \tag{7}$$

The feature points are selected based on the c values, those points with the maximum curvature are named as edge points and those with the minimum curvature are named as surface points. To avoid unreliable points, the following two rules are used: (i) the surrounding points around the feature point $p^i_{k,l}$ in the lth beam, $p^j_{k,l}(|j-l| < \tau, \tau$ is a positive integer), cannot be selected; (ii) $p^i_{k,l}$ can not lie on a surface patch which is roughly parallel to the lidar beam or the boundary of an occluded region, which is disconnected from the adjacent points by a gap and is at the same time further away the lidar.

We take advantage of the identities of beams to find edge lines or planar patches as correspondence for edge points or surface points, respectively. In Fig. 6, let an edge point p^i_{k,b_i} in the kth scan with $b_i \in [0, 1, ..., 63]$ as the identity of beam, let p^j_{k-1,b_j} be the closest neighbor point of p^i_{k,b_i} found using the 3D KD-tree [38] in (k-1)th scan, where b_j, denoting the identity of beam the point belongs to, is irrespective of the value of b_i. We again use the 3D KD-tree [38] to find the closest neighbour point of p^j_{k-1,b_j} in the (k-1)th scan and mark it as p^l_{k-1,b_l}, where $b_l = b_j - 1$ or $b_l = b_j + 1$. In this way, for every edge point p^i_{k,b_i} in the kth scan, we find an edge line $L_{j,l}$ decided by the two points p^j_{k-1,b_j} and p^l_{k-1,b_l}. Let p^i_{k-1} be the projection of p^i_{k,b_i} to $L_{j,l}$, then, the edge point p^i_{k,b_i} in the kth scan corresponds to the point p^i_{k-1} in the (k-1)th scan. For a surface point, we adopt a similar strategy to find a local small planar patch as the correspondence. We first find its closest neighbour point p^m_{k-1,b_m}, then, find another two points p^l_{k-1,b_n} and p^s_{k-1,b_s} as the closest neighbors of p^m_{k-1,b_m} with $b_n = b_m$ and $b_s = b_m - 1$ or $b_s = b_m + 1$. This guarantees that the three points are non-collinear and form a surface $S_{m,n,s}$. We mark the projection of the point p^i_{k,b_i} onto the surface $S_{m,n,s}$ as p^i_{k-1}. Through this strategy, (p^i_{k,b_i}, p^i_{k-1}) is the correspondence relationship we find for the edge points and surface points.

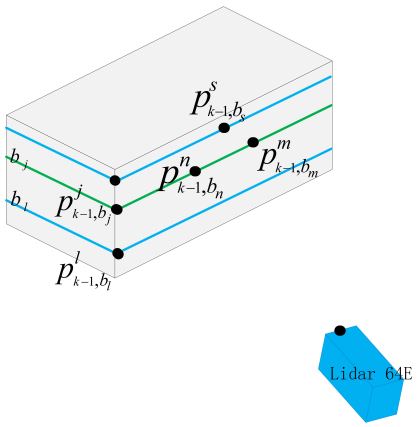


Figure 6: Finding the corresponding points for an edge point and a planar point, respectively. The colored lines represent the scan planes of the lidar's beams.

4.3. Plane-based metric criterion

No matter whether the selected feature point lies on the edge line or planar patch, the computed Euclid distance [11] always describes the attribution of the point to its associated point, without taking the geometric distribution of its local neighborhood into account. In this paper, we propose to adopt a probabilistic framework to model the locally planar plane distribution of the candidate point, which can be regarded as the plane-to-plane metric criterion. Based on the correspondences which have been found in the last subsection, we assume these corresponding points form two point sets: $S_k = \{p^i_k\}$, where $p^i_k \in \mathbb{R}^3$ for i$\in \{1, 2, ..., N\}$, and $S_{k-1} = \{p^i_{k-1}\}$, where $p^i_{k-1} \in \mathbb{R}^3$ for i$\in \{1, 2, ..., N\}$. Using the probabilistic model it is assumed that the point sets S_k and S_{k-1} are generated from an underlying set of distributions, where $p^i_k \sim N(\hat{p}^i_k, C^{S_k}_i)$ and $p^i_{k-1} \sim N(\hat{p}^i_{k-1}, C^{S_{k-1}}_i)$. Therefore given a perfect correspondence and the correct transform, Γ^*,

$$\hat{p}^i_{k-1} = (\Gamma^*)\hat{p}^i_k. \tag{8}$$

For an arbitrary rigid transform Γ, the difference between point p^i_k and p^i_{k-1} can be represented as $d^{(\Gamma)}_i = p^i_{k-1}$-$\Gamma p^i_k$. Since p^i_k and p^i_{k-1} are assumed to be drawn from independent Gaussian distributions,

$$d^{(\Gamma^*)}_i \sim N(0, C^{S_{k-1}}_i + (\Gamma^*)C^{S_k}_i(\Gamma^*)^T). \tag{9}$$

The transform can be solved by iteratively optimizing Γ using the maximum likelihood estimation and be simplified to the form:

$$\Gamma = arg \min_{\Gamma} \sum_{i} d_i^{(\Gamma)^T} (C_i^{S_{k-1}} + \Gamma C_i^{S_k} \Gamma^T)^{-1} d_i^{(\Gamma)}. \tag{10}$$

Since it is impossible to perfectly estimate the vehicle's motion, in addition, the disturbance of measurement noise, in fact, it will, in general, be impossible to sample the exact same point in two scans. Based on this observation, we model every extracted feature point to be distributed with high covariance in the direction tangent to the planar plane and with very low covariance in the normal direction. In other words, this corresponds to admitting that the point can provide very reliable information along the normal, but less certainty about its location in the surface. Define a confidence covariance matrix C as:

$$C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \gamma \end{bmatrix} \tag{11}$$

where γ is a small constant, denoting covariance along the normal direction. Given that e_k^i and e_{k-1}^i are respective normal vectors, calculated by the singular value decomposition (SVD) of the covariance matrix, at p_k^i and p_{k-1}^i, then, $C_i^{S_k}$ and $C_i^{S_{k-1}}$ can be computed by rotating the confidence matrix so that the γ term expresses uncertainty in the normal direction,

$$\begin{aligned} C_i^{S_k} &= R_{e_k^i} \cdot C \cdot R_{e_k^i}^T, \\ C_i^{S_{k-1}} &= R_{e_{k-1}^i} \cdot C \cdot R_{e_{k-1}^i}^T, \end{aligned} \tag{12}$$

where $R_{e_k^i}$ and $R_{e_{k-1}^i}$ are the rotation matrices which rotate γ to align with the matched points' normals, respectively.

In this paper, the local covariance matrix C_L is computed using the k nearest neighbors to the matched point found using [38]. The covariance at a given point can be expressed by means of the SVD of the covariance matrix, $C_L = USV^T$, the singular values are the diagonal elements of $S \in \mathbb{R}^{3\times3}$ stored in descending order and U and V are orthonormal matrices consisting of eigenvectors. The singular vector which corresponds to the smallest singular value is regarded as the normal vector. Eq.(12) can be realized by replacing its first term by U and its third term by V, respectively. The scan-to-scan lidar estimation is solved with the classical Levenberg-Marquardt method [39].

4.4. Optimized framework

The scan-to-scan motion estimation can be refined by a local bundle adjustment which takes a sequence of scans into the global map and performs a batch optimization. Let Q_{k-1} be the accumulated points in the global map until scan k-1, and T_{k-1}^{G} be the pose of the lidar in the global coordinate system at the end of scan k-1, then the local bundle adjustment algorithm iteratively optimizes T_{k}^{G} by matching current scan set S_k with Q_{k-1}, and uses optimized T_{k}^{G} to project S_k into the global map and finally form Q_k.

To register the local points with the accumulated global points, we store the global points of a certain range in cubic area. The points which overlap with S_k are cut out and stored in a 3D KD-tree for its simplicity. Let S be a point set of the local neighborhood of a feature point in Q_{k-1}, for the edge point, we first validate whether the set S is satisfied with the character of the line. We calculate the covariance matrix of S, denoted as C, and the eigenvalues and eigenvectors, respectively, are denoted as V and E. If the largest one of the eigenvalues V is significantly larger than the other two, the geometric shape of S can be described in the type of the line, and the vector associated with the largest eigenvalue expresses the orientation of the edge line. Thus we select two points along the direction of the edge line to define the line, and the corresponding point of an edge point can be determined by projecting the point to the edge line. On the other hand, if the geometric shape of set S can be described in the type of planar surface, the two larger eigenvalues are approximately equal and the third one should be significantly smaller than the former two. In a similar fashion, we can find the corresponding point for a surface point. We recall again the Levenberg-Marquardt method [39] to refine the registration. In this way, the coarse scan-to-scan motion evaluation and fine scan-to-map local bundle adjustment are grouped together to solve the pose transform.

5. Experimental validation

We validate the performance of the algorithm proposed in this paper by the qualitative tests on the data collected by our autonomous ground vehicle (AGV) on the campus and in urban environments and the quantitative comparisons with the state-of-the-art methods [9, 11, 18] on the public KITTI odometry datasets [40] (precise Ground Truth available). Our AGV is a modified Toyota Land Cruiser equipped with one Velodyne HDL-64E, one NovAtel SPAN-CPT GPS-aid INS, cameras and laser range finders et

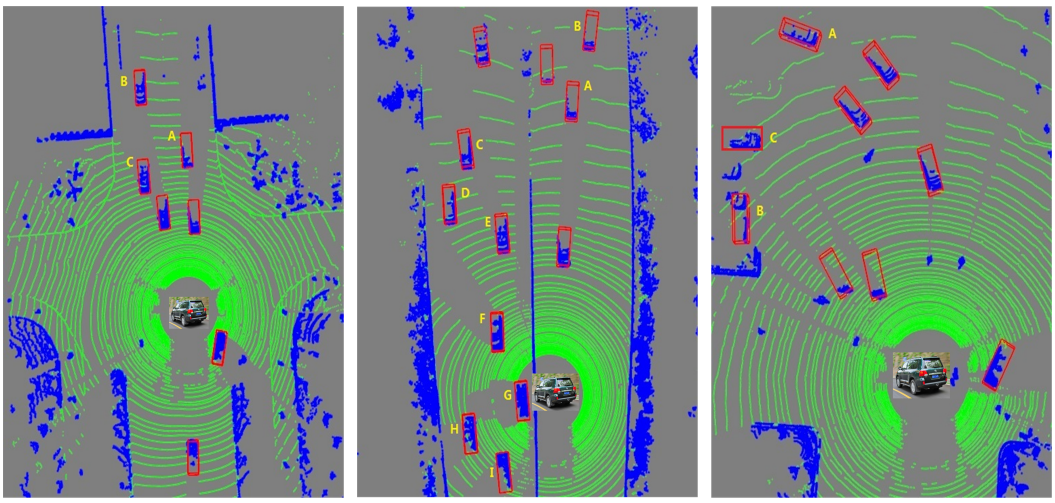
al, which is shown in Fig. 7. We run the algorithm in C++ and the point cloud library (PCL) [41] on a laptop equipped with a Core i7-2720QM central processing unit (CPU) with 2.20GHZ and 8 GB main memory in Ubuntu.

Figure 7: Our autonomous ground vehicle, equipped (on top) with a Velodyne HDL-LIDAR rotating 3D scanner, 10HZ, 64 beams, 2cm depth accuracy, collecting over 130,000 points per scan, FOV: 360^o horizontal, 26.8 ^{o} vertical, range: 120m.

5.1. Detection results

In the detection stage of moving vehicles, the ground points are removed using the Gaussian-Process-based ground segmentation [32] and the remaining obstacle points are clustered into potential objects with radially bounded nearest neighbour (RBNN) graph in [33]. The clusters are affirmed as moving vehicles through a series of processes of distinguishing, fitting, weighting and validating as described in Section 3. In our implementation, the parameters are set to $\Delta = 0.25$, $\sigma = 0.8$, $c_2 =$ -0.25, $c_3 = 1.25$, $\gamma =$0.04, $\lambda = 11$, $M = 20$, $N = 8$, $W=$ 1.8 and $L=$ 4.6.

We test our algorithm both on the crowded campus environments and in the busy urban scenes. The detection results of moving vehicles are shown in Fig. 8. On the crowded campus as shown in the left image where there are several moving vehicles occluded by others, the excellent method [7] cannot detect the occluded moving vehicles (labelled with the yellow letters A, B and C) because it only records the range to the closest object in each ray of the 2D virtual scan, However, our detection algorithm can deal with all moving vehicles according to our extended virtual scan and our novel vehicle



(a) Crowded campus (b) Busy city (c) The crossroad

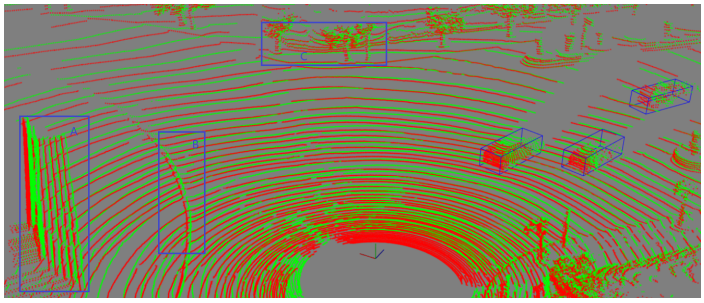
Figure 8: Results of moving vehicle detection when the test platform is driving along the road or turning on the campus or in an urban scene from left to right. The green points denote the ground, the blue objects denote the clusters of remaining points after removing the ground, the red cuboid boxes represent the detected moving vehicles. The boxes marked with yellow capital letters (from A to I) indicates the detected occluded vehicles. (Best viewed in colour.)

measurement model. In a busy urban environment, the occluded situations happen not only in the same-orientation roadway but also in the different-orientation one, which is especially obvious when there is some vegetation between the roadways such as the long blue line region shown in Fig. 8(b). Without removing the moving vehicles, it will inevitably have a negative effect on the subsequent scan registration and odometry with so many matched point-pairs falling on the corresponding moving ones. With our method, all moving vehicles consisting of the closest ones and the occluded ones (A, B, ..., I) can be detected. Fig. 8(c) illustrates the detection result in the crossroads in urban environment when the test platform is turning left. Our likelihood-field measurement model can well estimate the different poses of moving vehicles.

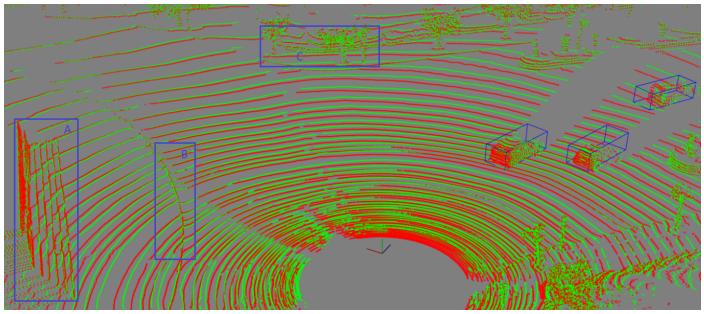
5.2. Aligned results

To evaluate the effect on the performance of scan registration with or without the point-pairs falling on the moving vehicles, we compare our algorithm with removing the moving cars, with the state-of-the-art LOAM [11]

without taking out those cars. We first test the performances of compared methods on the campus where the test platform is turning right. The result is shown in Fig. 9.



(a) LOAM on the campus.



(b) Ours on the campus.

Figure 9: The registration results of the LOAM with all feature points, our method with removing those points falling on the moving vehicles when the test platform is turning right on the campus. The red and green point clouds denote the current scan and the last scan, respectively. The blue cubic boxes represent the detected moving vehicles, while the blue rectangles indicate the details of aligned results of different methods. The shorter the distance between the corresponding points, the better the performance of the method. (Best viewed in colour).

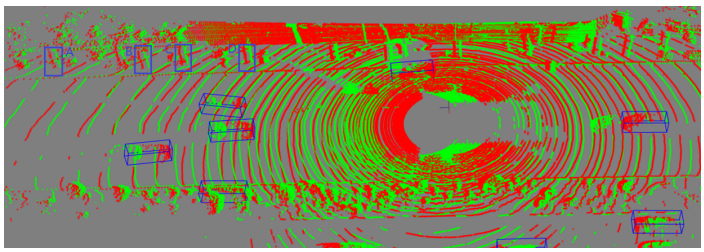
In Fig. 9, the rectangular coordinate system denotes the position of our platform with the blue axis representing the orientation of the head of the vehicle. The red and green point clouds are from the current scan and the last scan, respectively. The blue cubic boxes represent the detected moving vehicles, while the blue rectangles indicate the details of aligned results of the two algorithms. The rectangle A is a stone sculpture with several long stairs at the bottom. In Fig. 9(a), the edge lines of these stairs are ambiguous and it seems there are two lines in the same position. The curve in the

rectangle B is a road curb, there exists a space between two curves and we can obviously distinguish between the red and green curves. In the rectangle C, each red tree trunk is close to a green tree trunk. The commonality in the three regions is that it represents an real object as two virtual ones and thus generates the ghost image. By contrast, Fig. 9(b) is completely different. The edge lines of stairs in the region A are clear; the curve in the region B is well overlapped into one; the red points of tree trunks are covered with the green points in the region C and it shows the real number of trees. By removing the matched points falling on the moving vehicles, the ghost image disappears.

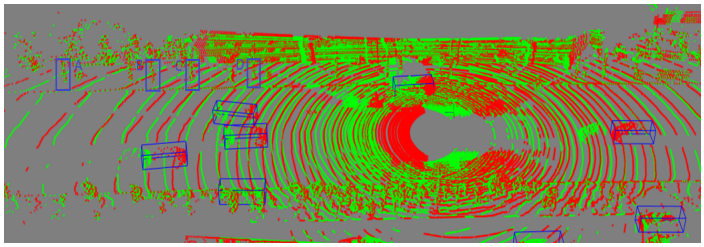
A test is also performed in busy urban environment where there are dozens of moving vehicles or even more in the sensing range of the lidar. We select a main trunk road in the rush hour and show the results of scan registration of the compared methods in Fig. 10. The blue rectangles A, B, C and D show the matched results of four trees along the road. In Fig. 10(a), each rectangle region includes a red and a green tree trunk, in contrast to the trees in Fig. 9(a), the distance between the two trees is obviously further. That's because, on the one hand, there are more moving vehicles in the scene and thus more points falling on them are mistaken as an input for scan registration, while, on the other hand, the velocities of the moving vehicles are higher in urban environment and thus every matched point pair hitting them has a highly negative effect on the optimized function. Owing to wrongly drawing those moving points at a distance together in the optimized process, it will certainly generate the error and result in that the static objects cannot perfectly overlap. In contrast, our registration algorithm with removal of the moving vehicles can handle the problem as shown in Fig. 10(b). Every region only includes one tree and cannot generate the ghost image.

5.3. Odometry results

In addition to the single-frame vehicle detection and scan registration experiments on our collected point clouds, the odometry tests of sequential frames are also carried out on the public KITTI benchmark datasets [40, 42] which are recorded with sensors mounted on the top of a passenger vehicle while driving in the real-world traffic situations around Karlsruhe, Germany. The test platform is equipped with a Velodyne 3D laser scanner, two high-resolution stereo cameras and a high-precision GPS-aid INS localization unit with RTK correction signals which ensures that the localization error is less than 5 cms in the open sky for ground truth purpose. The test odometry



(a) LOAM in busy urban environment.



(b) Ours in busy urban environment.

Figure 10: The performances in the busy urban environment where there are dozens of moving vehicles or even more in the sensing range. This figure only shows a small part of the whole scene. The meanings of coloured points and boxes are the same as the ones in Fig. 9. With the disturbance of points hitting the moving vehicles, the LOAM alignment cannot transform the static objects to overlap well, like the trees shown in the blue rectangles in (a). Our method of removing the effect can cause the static points to overlap perfectly. (Best viewed in colour).

datasets mainly consist of three kinds of structured scenarios: the urban scene with buildings around and many still vehicles stopping along the road, the country scene on small roads with different kinds of vegetation, and the highway scene on wide roads with a relatively clean environment. The former two scenarios do not have moving vehicles in the majority of the frames and only contain one or two moving vehicles with very small velocity in the minority of frames. The latter scenario has moving vehicles most of the time and their velocities are relatively high. This is the scope we are interested in and we quantitatively compare the performances of different methods on the highway.

For deeper insight into the performances and failure modes of individual algorithms, the rotation and translation errors are evaluated separately as a function of the test platform's trajectory length and velocity [40]. The error

metrics can be depicted as:

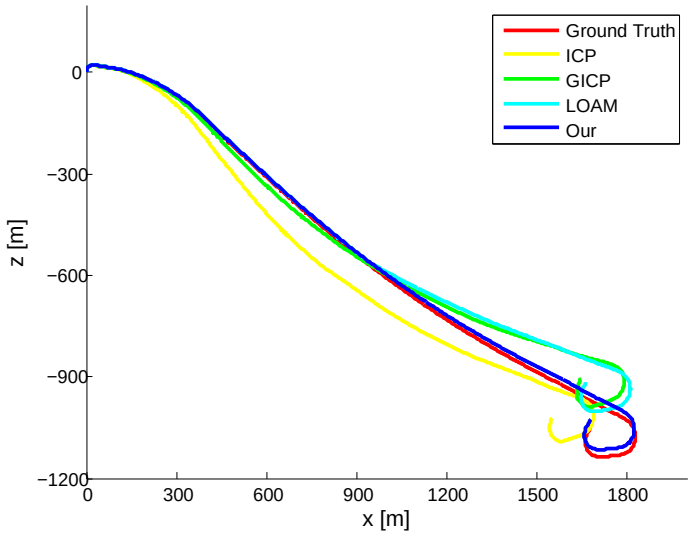
$$
\begin{aligned}
E_r(S) &= \frac{1}{|S|} \sum_{(i,j)\in S} \angle[(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)], \\
E_t(S) &= \frac{1}{|S|} \sum_{(i,j)\in S} \|(\hat{p}_j \ominus \hat{p}_i) \ominus (p_j \ominus p_i)\|,
\end{aligned}
\tag{13}
$$

where S is a set of scans$(i,\ j)$ with i denoting the first scan and j the last scan of S, $\hat{p} \in SE(3)$ is the estimated lidar pose and $p \in SE(3)$ is the ground truth, $\ominus$ expresses the inverse operation of the transform matrix [43] and $\angle[.]$ is the rotation angle.

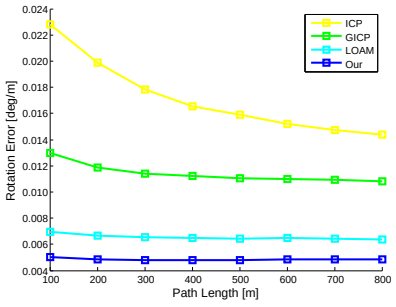
The compared methods consist of the classical ICP [18] algorithm which attempts to optimize the transform in the point-to-point way; the GICP [9] algorithm which models every point as the locally planar surface and exploits the plane-to-plane method to optimize the transform; the LOAM [11] method which combines the coarse scan-to-scan motion evaluation and fine scan-to-map bundle adjustment together to form a uniform framework to solve the transform; our method first removes all the moving vehicles in the scene with our measurement model and extended virtual scan and then exploits the plane-to-plane method in the framework to deal with the optimization of the undistorted point clouds. The results of lidar odometry of different methods on the highway are shown in Fig. 11.

Fig. 11(a) shows the calculated trajectories of the compared methods with the start position as the origin. The red curve depicts the ground truth provided by the high-precision GPS-aid INS. The yellow curve denotes the result of the ICP which matches well only in the beginning and gradually increases the drift along with the distance travelled. The drift has always risen to 93m when the x-value is 900m and finally the error reaches 134.1m in the total 2451.6m away. By contrast, the green curve is better, which matches well in the former half of the trajectory, however, the error starts to increase significantly in the latter half of the path and finally it reaches 142.3m. The cyan curve depicts the result of the LOAM which is very similar to the result of the GICP and its final error is 130.5m. The blue curve shows that our algorithm always matches well in the whole trajectory and the final drift is 19.8m. The following four figures describe the quantitative results of the compared methods in details.

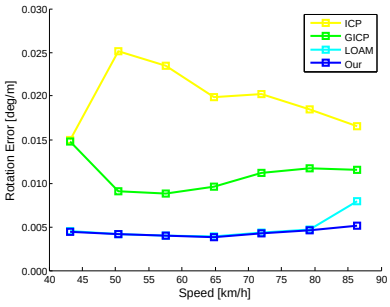
For every trajectory's position (interval 1s), the rotation error, the translation error and the velocity are calculated in every trajectory segment at



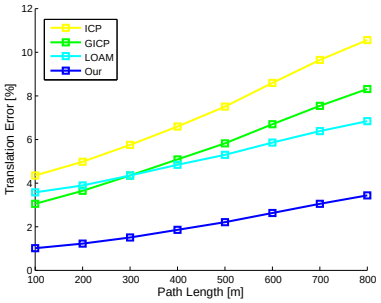
(a) The trajectory of test vehicle.



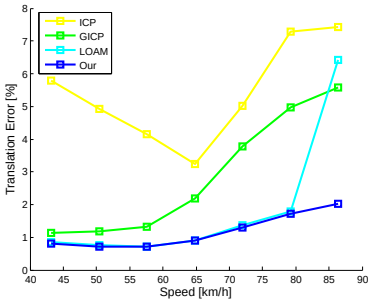
(b) Rotation error versus path.



(c) Rotation error versus speed.



(d) Translation error versus path.



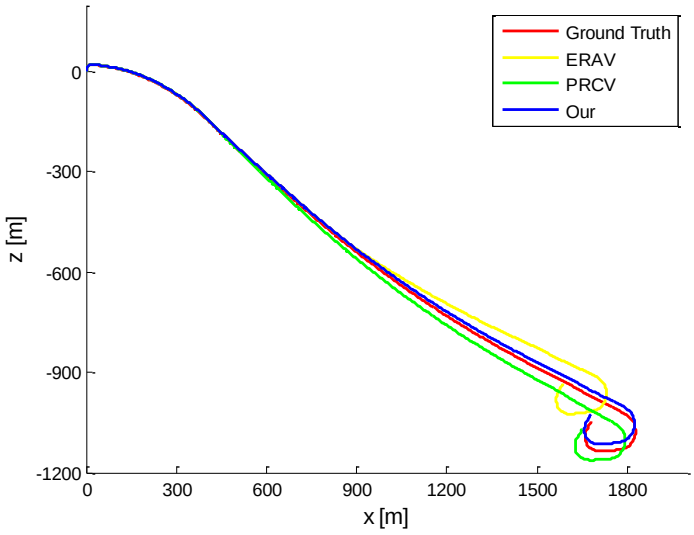
(e) Translation error versus speed.

Figure 11: Lidar odometry performances of different methods. The rotation and translation errors are the function of path traveled and driving speed. (Best viewed in colour).

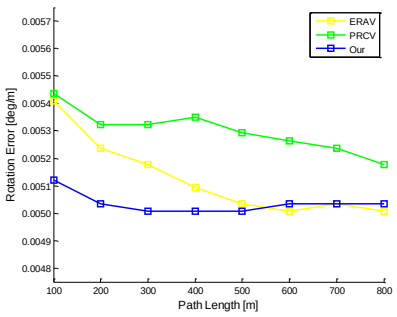
100m, 200m, ..., 800m length in 3D coordinates. The averages of the errors in all positions versus the segments and the velocities are drawn to quantitatively evaluate the performances of the methods. Fig. 11(b) shows the rotation errors of the methods versus every trajectory's length. The error of the ICP reaches 0.023 deg/m at the beginning, which is nearly twice as high as the error of the second highest GICP. The two errors gradually decrease along with the increase in the trajectory length, with the final errors 0.0144deg/m and 0.0108deg/m, respectively. The reason why the GICP is superior to the ICP lies in that the modelled locally planar surface structure provides a high confidence in the norm direction. By contrast, the errors of the LOAM and our algorithm are lower and more stable, in addition, the average error of our method is 0.0017 deg/m less than that of the LOAM, which amounts to a quarter of the average error. The rotation error versus the velocity is shown in Fig. 11(c), where we can see that the test vehicle is driving at a high speed. The errors of the former two methods are higher and change greatly, whereas the latter two results are always lower and more stable with the LOAM's error increasing only in the last stage. The average rotation errors of the ICP, the GICP, the LOAM and our algorithm are 0.0175 deg/m, 0.0115 deg/m, 0.0066 deg/m, and 0.0048 deg/m, respectively.

The translation errors versus the distances travelled using the four methods is shown in Fig. 11(d). The four errors increase linearly with the path length. The ICP's error is the biggest, the errors of the GICP and the LOAM are similar, and our error is the smallest. The linear increase in translation errors derives from the similar appearance of the objects on the highway and the accumulated errors. Fig. 11(e) shows the translation errors versus the speeds which is similar to the result in Fig. 11(d). The mean translation errors of the ICP, the GICP, the LOAM and our method are 6.95%, 5.31%, 4.96% and 1.99%, respectively. We can see that our error is less than half of the error of the second smallest method. The reasons for the excellent performance are multiple: the rectification of distorted point clouds, the plane-based measurement metric with high confidence, the coarse-to-fine optimization framework, and the key point, the removal of all the moving vehicles.

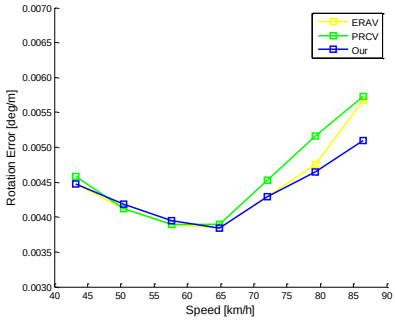
In this paper, we put the likelihood-field-based measurement model in the extended virtual scan and the probabilistic metric criterion together to register scans. To test whether both of the two components can improve the performance of the approach, we do the following experiment and the result is shown in Fig. 12.



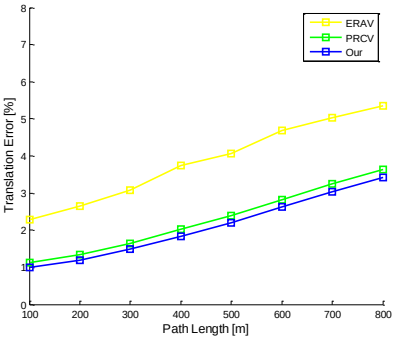
(a) The trajectory of test vehicle.



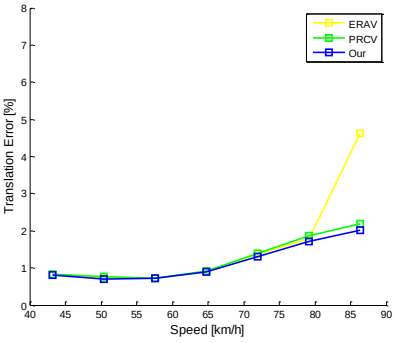
(b) Rotation error versus path.



(c) Rotation error versus speed.



(d) Translation error versus path.



(e) Translation error versus speed.

Figure 12: Lidar odometry performances of different methods. (Best viewed in colour).

In Fig. 12, the yellow curve denotes the result of odometry using Euclid distance as the criterion in the alignment as in [11] and removing all moving vehicles in the scene with our proposed extended virtual scan and the likelihood-field-based vehicle model. It is a combination of our moving vehicle detection method in Section 3 and the method in [11]. We call this combination ERAV (Euclid criterion and remove all vehicles). The green curve represents the result of odometry using the proposed probabilistic criterion in the alignment and only removing the closest moving vehicles as [7]. It is a combination of the method in [7] and our plane-based scan registration in Section 4. We use the abbreviation PRCV (probabilistic criterion and remove the closest vehicles) to describe this combination. The blue curve is the result of our proposed method. The difference between the method ERAV and our method shows the effect of our proposed probabilistic criterion of Section 4, while the difference between the method PRCV and our method shows the effect of our moving vehicle detection of Section 3.

In Fig. 12(a), the three curves match well in the former half of the trajectory, however, their performances emerge as different along the distance travelled. Finally, the drift errors of the methods ERAV, PRCV and our method reach 97.5 meters, 41.1 meters and 19.8 meters, respectively. Fig. 12(b) and Fig. 12(c) show the rotation errors versus the path length and the speed. The overall trends of the three curves are very similar. The average rotation errors of the three methods are 0.0054 deg/m, 0.0051 deg/m and 0.0048deg/m, respectively. Fig. 12(d) and Fig. 12(e) show their translation errors. The average translation errors are 3.86%, 2.28% and 1.99%, respectively. The qualitative and quantitative comparisons demonstrate that both of the two components can improve the performance of the approach.

As every scan collects such abundant points, none of the aligned methods in the Velodyne HDL-64E can attain real-time. We test the four methods on the same computer equipped with a Core i7-2720QM CPU with 2.20GHZ and 8 GB main memory. The ICP [18] method using all points to match requires an average of 5.3s to process every scan. By contrast, the GICP [9] which also uses the total points, generates an alignment of higher accuracy and takes 3.9s per frame. The LOAM [11] using feature points to match, saves greatly the computation time and reduces to 1.0s, which is the shortest in the four methods. The processing time of our algorithm with the novel criterion to measure those matched feature points from the stationary objects, is 1.36s, it consists of the removal of all moving vehicles (0.16s) and the more reliable registration (1.2s). Despite our computation time is slightly more

than the LOAM's, our aligned accuracy is obviously higher than it, which is the key point for the accumulated odometry, 3D consistent mapping and precise localization.

6. Conclusions

The scan registration using the data from a moving Velodyne HDL-64E in real-world traffic scenarios can be difficult because the problem involves the removal of moving vehicles and the estimation of motion in the abundance of point clouds. In this paper, we propose an extended 2D virtual scan to obtain all the moving objects in the scene by the scan differencing operation between two consecutive scans with motion compensation from GPS-aid INS or the local wheel encodes. For every moving object, a vehicle is fitted with our proposed likelihood-field-based vehicle measurement model. After passing the validation of the vehicle motion evidence, the points hitting the moving vehicles are removed and the remaining ones are taken as an input into our alignment.

In the alignment, we first rectify the distorted points from the moving lidar by linearly interpolating the pose transformation within the scan interval. Then we exploit the probabilistic framework to model a local plane structure of the selected two kinds of feature points, which can be regarded as the plane-to-plane metric criterion which ensures a high confidence in the normal direction. The pose estimation is solved with the combination of high frequency but coarse motion estimation and low frequency but fine batch adjustment. The effectiveness of our method is validated by a large number of qualitative experiments on our point clouds and quantitative comparisons with the state-of-the-art approaches on the highway from the public KITTI odometry datasets.

7. Appendix

To solve the integrated over the rectangular region R_0, we use a strategy similar to the integral image in the computer vision. We first derive the integral for 1D gaussian over a line and then extend it to 2D. Let the measurement $f(x)$ be subject to 1D gaussian distribution, $f(x) \sim N(x_m, \sigma)$, with mean x_m and standard deviation σ. The integral of $f(x)$ over the unit

step function occurring at x_a can be calculated:

$$I_{x_a} = \int_{x_a}^{\infty} N(x_m, \sigma)dx = \frac{1}{2}(1 + erf(\frac{x_m - x_a}{\sqrt{2}\sigma})). \tag{14}$$

The integral of this function over the line from the start point x_a to the end point x_b can be represented as:

$$I_{x_a}^{x_b} = I_{x_a} - I_{x_b}. \tag{15}$$

Now we extend 1D integral over the line to 2D integral over a rectangular region as shown in Fig. 3(b). Let I_D be the integral of Gaussian over a quarter-plane from the point D to the positive infinity and similarly, I_A, I_B and I_C are the integral over corresponding quarter-planes, respectively. The 2D Gaussian integral over each quarter-plane is the product of x and y 1D integrals:

$$I_D = I_{Dx} I_{Dy}. \tag{16}$$

Then the integral of q_i^V over the visible rectangular box R_0 can be simply depicted as follows:

$$I(q_i^V, R_0) = I_D + I_B - I_A - I_C, \tag{17}$$

where,

$$\begin{aligned}
I_A &= \frac{1}{4}(1 + erf(\frac{x_i^V - x_a^V}{\sqrt{2}\sigma}))(1 + erf(\frac{y_i^V - y_a^V}{\sqrt{2}\sigma})); \\
I_B &= \frac{1}{4}(1 + erf(\frac{x_i^V - x_b^V}{\sqrt{2}\sigma}))(1 + erf(\frac{y_i^V - y_b^V}{\sqrt{2}\sigma})); \\
I_C &= \frac{1}{4}(1 + erf(\frac{x_i^V - x_c^V}{\sqrt{2}\sigma}))(1 + erf(\frac{y_i^V - y_c^V}{\sqrt{2}\sigma})); \\
I_D &= \frac{1}{4}(1 + erf(\frac{x_i^V - x_d^V}{\sqrt{2}\sigma}))(1 + erf(\frac{y_i^V - y_d^V}{\sqrt{2}\sigma})).
\end{aligned} \tag{18}$$

References

[1] M. Bosse, R. Zlot, P. Flick, Zebedee: design of a spring-mounted 3d range sensor with application to mobile mapping, IEEE Transactions on Robotics 28 (5) (2012) 1104–1119.

[2] R. Zlot, M. Bosse, Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine, in: International Conference on Field and Service Robotics, 2012, pp. 479–494.

[3] U. Weiss, P. Biber, Plant detection and mapping for agricultural robots using a 3d lidar sensor, Robotics and Autonomous Systems 59 (5) (2011) 265–273.

[4] J. Levinson, S. Thrun, Robust vehicle localization in urban environments using probabilistic maps, in: International Conference on Robotics and Automation, 2010, pp. 4372–4378.

[5] F. Moosmann, O. Pink, C. Stiller, Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion, in: IEEE Intelligent Vehicles Symposium, 2009, pp. 215–220.

[6] Y. Yi, Y. Guang, Z. Hao, F. Meng-yin, W. Mei-ling, Moving object detection under dynamic background in 3d range data, in: IEEE Intelligent Vehicles Symposium, 2014, pp. 394–399.

[7] A. Petrovskaya, S. Thrun, Model based vehicle detection and tracking for autonomous urban driving, Autonomous Robots 26 (2-3) (2012) 123–139.

[8] B. Douillard, A. Quadros, P. Morton, J. P. Underwood, M. D. Deuge, S. Hugosson, M. Hallstrom, Scan segments matching for pairwise 3d alignment, in: International Conference on Robotics and Application, 2013, pp. 3033–3040.

[9] A. V. Segal, D. Haehnel, S. Thrun, Generalized-icp, in: Robotics: Science and Systems Conference, 2009, pp. 26–33.

[10] M. Magnusson, A. Lilienthal, T. Duckett, Scan registration for autonomous mining vehicles using 3d-ndt, Journal of Field Robotics 24 (10) (2007) 803–827.

[11] J. Zhang, S. Singh, Loam: lidar odometry and mapping in real-time, in: Robotics: Science and Systems Conference, 2014.

[12] L. E. Navarro-Serment, C. Mertz, N. Vandapel, M. Hebert, Lidar-based pedestrian detection and tracking, in: International Conference on Robotics and Automation, 2008.

[13] N. Wojke, M. Häselich, Moving vehicle detection and tracking in unstructured environments, in: International Conference on Robotics and Automation, 2012, pp. 3082–3087.

[14] A. Azim, O. Aycard, Layer-based supervised classification of moving objects in outdoor dynamic environment using 3d laser scanner, in: IEEE Intelligent Vehicle Symposium, 2014, pp. 1408–1414.

[15] A. Petrovskaya, M. Perrollaz, L. Oliveira, L. Spinello, R. Triebel, A. Makris, et al., Awareness of road scene participants for autonomous driving, in: Handbook of Intelligent Vehicles, 2012, pp. 1383–1432.

[16] F. Moosmann, C. Stiller, Velodyne slam, in: IEEE Intelligent Vehicle Symposium, 2011, pp. 393–398.

[17] D. D. Morris, R. Hoffman, P. Haley, A view-dependent adaptive matched filter for ladar-based vehicle tracking, in: International Conference on Robotics and Applications, 2009, pp. 310–318.

[18] S. Rusinkiewicz, M. Levoy, Efficient variants of the icp algorithm, in: International Conference on 3D Digital Imaging and Modeling, 2001, pp. 145–152.

[19] R. Dube, M. Hahn, M. Schutz, J. Dickmann, D. Gingras, Detection of parked vehicles from a radar based occupancy grid, in: IEEE Intelligent Vehicles Symposium, 2014, pp. 1415–1420.

[20] J. Cheng, Z. Xiang, T. Cao, J. Liu, Robust vehicle detection using 3d lidar under complex urban environments, in: International Conference on Robotics and Automation, 2014, pp. 691–696.

[21] L. Zhang, Q. Li, M. Li, Q. Mao, A. Nuchter, Multiple vehicle-like target tracking based on the velodyne lidar, in: Intelligent Autonomous Vehicles Symposium, 2013, pp. 126–131.

[22] A. Azim, O. Aycard, Detection, classification and tracking of moving objects in a 3d environment, in: IEEE Intelligent Vehicles Symposium, 2012, pp. 802–807.

[23] D. Morris, P. Haley, W. Zachar, S. McLean, Ladar-based vehicle tracking and trajectory estimation for urban driving, in: Unmanned Systems North America Conference, 2008, pp. 160–174.

[24] B. Michael, Z. Robert, Keypoint design and evaluation for place recognition in 2d lidar maps, Robotics and Autonomous Systems 57 (12) (2009) 1211–1224.

[25] A. E. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3d scenes, Pattern Analysis and Machine Intelligence 21 (5) (2002) 433–449.

[26] R. B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (fpfh) for 3d registration, in: International Conference on Robotics and Automation, 2009, pp. 3212–3217.

[27] B. Steder, R. B. Rusu, K. Konolige, W. Burgard, Narf: 3d range image features for object recognition, in: Intelligent Robots and Systems, 2010.

[28] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, M. Beetz, Towards 3d point cloud based object maps for household environments, Robotics and Autonomous Systems 56 (11) (2008) 927–941.

[29] D. Borrmann, J. Elseberg, K. Lingemann, A. N*ü*chter, J. Hertzberg, Globally consistent 3d mapping with scan matching, Robotics and Autonomous Systems 56 (2) (2008) 130–142.

[30] T. Stoyanov, M. Magnusson, A. Lilienthal, Point set registration through minimization of the l2 distance between 3d-ndt models, in: International Conference on Robotics and Automation, 2012, pp. 1050–5201.

[31] T. Stoyanov, R. Mojtahedzadeh, H. Andreasson, A. J. Lilienthal, Comparative evaluation of range sensor accuracy for indoor mobile robotics and automated logistics applications, Robotics and Autonomous Systems 61 (10) (2012) 1094–1105.

[32] T. Chen, B. Dai, R. Wang, D. Liu, Gaussian-process based real-time ground segmentation for autonomous land vehicle, Journal of Intelligent and Robotic Systems 76 (3-4) (2014) 563–582.

[33] K. Klasing, D. Wollherr, M. Buss, A clustering method for efficient segmentation of 3d laser data, in: International Conference on Robotics and Automation, 2008, pp. 4043–4048.

[34] A. Petrovskaya, O. Khatib, S. Thrun, Y. N. Andrew, Bayesian estimation for autonomous object manipulation based on tactile sensors, in: International Conference on Robotics and Automation, 2006, pp. 707–714.

[35] A. Petrovskaya, O. Khatib, Global localization of objects via touch, IEEE Transaction on Robotics 27 (3) (2011) 569–585.

[36] A. Nüchter, K. Lingemann, J. Hertzberg, H. Surmann, 6d slam-3d mapping outdoor environments, Journal of Field Robotics 24 (8-9) (2007) 699–722.

[37] J. Sprickerhof, A. N*ü*chter, K. Lingemann, J. Hertzberg, A heuristic loop closing technique for large-scale 6d slam, in: European Conference on Mobile Robots, 2012.

[38] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, Computation geometry: algorithms and applications, 3rd Edition, Springer, 2008.

[39] R. Hartley, A. Zisserman, Multiple view geometry in computer vision, 2nd Edition, Cambridge University, 2004.

[40] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? the kitti vision benchmark suite, in: Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.

[41] R. B. Rusu, S. Cousins, 3d is here: point cloud library (pcl), in: International Conference on Robotics and Automation, 2011, pp. 1–4.

[42] A. Geiger, P. Lenz, R. Urtasun, Vision meets robotics: the kitti dataset, International Journal of Robotics Research 32 (11) (2013) 1229–1235.

[43] R. Kuemmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, A. Kleiner, On measuring the accuracy of slam algorithms, Autonomous Robots 27 (4) (2009) 387–407.