NDT Localization with 2D Vector Maps and Filtered LiDAR Scans

Maxime Escourrou, Joelle Al Hage, Philippe Bonnifait

NDT Localization with 2D Vector Maps and Filtered LiDAR Scans

Maxime Escourrou, Joelle Al Hage and Philippe Bonnifait

***Abstract*—High accuracy localization is a basic requirement for autonomous vehicles navigation. However, in urban environments, Global Navigation Satellite Systems (GNSS) suffer from Non-Line of Sight (NLoS) signals, multipath and sometimes a limited number of visible satellites, degrading the localization accuracy. Maps with georeferenced features are a means to address this issue. In this paper, an open access map with cadastral footprints of the buildings is used for localization. Buildings are stable over time and provide visible features in cities. Using 2D footprints of the buildings provides little detailed information, but when they are matched with long range omnidirectional LiDARs, a good quality estimated pose can be achieved. We present a method that uses the Normal Distributions Transform (NDT) to match several layers of a LiDAR scan with the map. A fast filtering method based on local linear regression is proposed to extract aligned points in the LiDAR scans which filters out the largest part of the outliers before applying the NDT optimization. The performance of the approach is evaluated on real data recorded with an experimental vehicle equipped with a ground truth. The results show that this approach is able to provide high accuracy consistent with autonomous navigation tasks.**

***Keywords*—Map aided localization, Point cloud filtering, Normal Distributions Transform, Autonomous vehicles.**

I. Introduction

Autonomous driving is an active research field presenting a lot of progress in the last years. In order to achieve the decimetric localization accuracy requirement needed for safe control, GNSS needs to be augmented by other sources of information, classically merged with a Kalman Filter [1], [2]. Indeed, in urban environments, GNSS suffers from mutlipath and NLoS signals which affect the localization accuracy. An alternative relies on methods based on Simultaneous Localization And Mapping (SLAM) [3] that can achieve good localization accuracy in the absence of prior map [4], [5]. To perform well, these methods rely on loop closure to improve the localization and to reduce the drift [5]. The main limitation of loop closure is that it requires returning to a previously visited place.

These issues can be handled by the use of a map-aided localization method. In [6], a map-aided localization is proposed where the localization is done with respect to prior constructed map. This map is built by a Mobile Mapping System (MMS) using a high accuracy localization system. The map obtained from MMS allows to achieve high accuracy localization but its construction requires a first pass which makes it limited to small region and expensive to obtain.

The authors are with Heudiasyc, UMR CNRS 7253, Université de Technologie de Compiègne, France.

978-1-6654-1213-1/21/$31.00

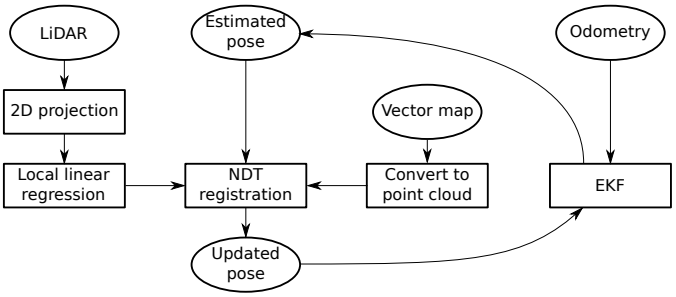

Fig. 1. Localization architecture.

In this paper, a map-aided localization method using cadastral building vector map is proposed. Indeed, these maps are usually accessible and cover almost all the country's buildings. However, they are represented in 2D by their footprint on the ground. Buildings are the most stable and easily observable features in urban environments, the aim is therefore to localize the vehicle with high accuracy by matching the LiDAR measurements with those buildings. For this purpose, a local linear regression method is proposed to extract the aligned points associated to buildings from the LiDAR point clouds. This method has the advantage to be very fast and to keep enough points that have a correct match with the map. The NDT registration is then applied to the filtered LiDAR point cloud while using the map converted to 2D point cloud. The pose estimated by the NDT is then used in an Extended Kalman Filter (EKF) as a measurement update. The general architecture of the proposed method is shown in Figure 1.

The main contributions of this paper are:

- The proposition of a new filtering method based on a local linear regression to extract the buildings from a LiDAR point cloud,
- The use of vector maps of the buildings to perform NDT localization in urban environments,
- An experimental evaluation with real data.

This paper is organized as follows: section II details the state of the art of the registration and point-cloud filtering methods. Section III presents the proposed linear regression method for aligned points extraction. Section IV shows the localization procedure based on the use of the vector map and the NDT, in addition to the data fusion method. The results of the experimental study based on data acquired in urban environment are presented in section V. Conclusion and future work are given in section VI.

II. Related Works

Before matching the LiDAR point cloud to the map, it should be filtered in order to optimize the performance of the registration step by removing the non-building points. In this section, we will present the state of the art on registration and filtering methods.

A. Registration

Regarding the registration step, the most known method for registration is the Iterative Closest Point (ICP), introduced in [7], [8]. It is a registration algorithm which mathematically converges toward the nearest local minimum. It minimizes the sum of the square distances between each point and its closest neighbor in the other scan. Another version based on the point-to-line minimizes the distance between each point and its projection on the closest line [8]. A generalized version of the ICP is proposed in [9] that attaches a probabilistic model to the minimization problem. Another extension namely probabilistic data association was designed to align sparse and dense point cloud together [10]. It uses a full probabilistic model in order to improve the robustness against noise. Each point in the source point cloud is associated to a set of points in the target one and then weighted based on a probabilistic model. In [11], the authors perform a graph-SLAM using as additional constraints the buildings data from OpenStreetMap (OSM). They convert the 2D LiDAR scan into a set of polylines which are then match with the OSM using ICP. Some improvements were proposed in [12] in order to take into account the noisy measurements and observation integrity.

Another way to register two point clouds is proposed in [13], [14] and is known as the Normal Distributions Transform (NDT). Unlinke ICP, there is no need for a direct correspondence between points. The principle of this method is to subdivide the space into a grid and to attach a normal distribution to each cell depending on the distribution of the points of the reference point cloud inside this cell. The goal is then to maximize the sum of the log-likelihood of the second point cloud in those cells to find the transformation between the two point clouds. The method was extended to 3D in [15] and [16]. Other extensions such as the clustered NDT [17] or multiple cell size [18] have been proposed to solve the duality between accuracy and convergence. The NDT has the advantage of assigning distribution to points, which results in better performance and remove the need of expensive computation search of nearest-neighbour as in the case of ICP. Indeed, it performs an efficient probabilistic optimization where no direct association between points is needed. It is proven to be more robust than ICP when the initial transformation is of poor quality and to perform better with poor overlap [19].

B. Point cloud pre-filtering

Filtering is an important step to reduce the size of the point cloud and to ameliorate the performance of the registration by removing the outlier points.

In [20], Random Sample Consensus (RANSAC) is used to extract planes, lines, or other types of shapes. First, the shape is characterized with n parameters. Then, n points are selected randomly in the point cloud to define the shape. The consensus is computed by counting the number of points whose projected distance to the shape is below a defined threshold. After a fixed number of iterations, the set of points with the maximum consensus is chosen. An ameliorate version is presented in [21] where the number of iterations, instead of being fixed, is based on the probability of having already found the best consensus. The main disadvantage of RANSAC appears in the execution time needed to extract multiple lines. Moreover, the RANSAC method can be disturbed by aligned trees, bushes, etc.

Another method for filtering, known as the Hough transform, is presented in [22]. It detects characterized 2D shapes by converting the points from the Cartesian space to a parameter space. This method is often used in computer vision but it suffers from high computational cost. Other methods like region growing and facet segmentation [23] allow to detect the borders of a region by selecting neighboring points which satisfy a given property (coplanarity for example). These methods are not well adapted for sparse point clouds.

In [5], the LiDAR Odometry and Mapping (LOAM) is presented. The method is based on the extraction of feature points by evaluating the smoothness of the local surface (sharp edges with small smoothness and planar surfaces with high smoothness). This method extracts very few points in order to match them with their accumulative map.

In addition to geometric methods, deep-learning is used to extract classified features from point clouds as in the case of SegMatch [24]. However, in this paper the focus is on geometric methods.

III. Point cloud pre-filtering

Local linear regression is proposed for filtering out LiDAR points that have little chance to correspond to buildings. It does not require high computations and can remove significant outliers. As in the case of LOAM filtering [5], the method that we propose uses the neighbors to decide if a point is to be kept or not. In this work, neighbors on the same layer are used, while in some other works those on other layers are considered as was done in the LOAM method. As we consider the buildings facades as vertical in the LiDAR frame (the rotation axis of LiDAR has to be approximately aligned with gravity), a layer presents a nearly horizontal section of the surrounding buildings. Therefore, the close neighbors on a same layer can be locally considered on the same horizontal plane.

To remove a maximum of non-building points, the only layers used are those above the horizontal plane of the LiDAR (positive laser elevation angle). In this work, with a 32 layers Velodyne VLP32-C, the 12 upper ones are used. The point cloud is projected to 2D (keeping only x and y coordinates), then the following presented algorithm is applied on each layer, independently. The points on each layer are ordered by

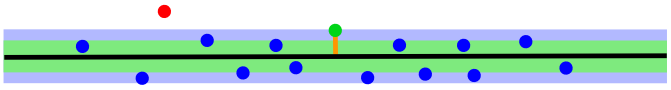

Fig. 2. For the linear regression, the green point is considered and M points are taken before and after (for example, $M = 7$ is this figure). In black, the line obtained from the linear regression.

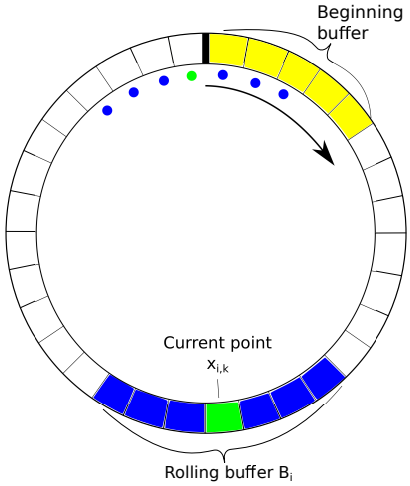

Fig. 3. Rolling buffer and "beginning buffer" : Here is the representation of the processing of the layer i with a rolling buffer of 7 points ($M = 3$). In green and blue, the standard case of the rolling buffer (buffer B_i in Algorithm 1), with in green the considered point. Treatments of the points at the beginning and at the end are special cases and need the "beginning buffer" in yellow.

their azimuth angle, which correspond to ordering by time of arrival on a rotary LiDAR.

A linear regression is computed around each point with its neighboring points on the same layer (M points before and M after). The characteristics of this linear regression (variance and residual (distance to the line)) define if the middle point has to be rejected or not (figure 2). A rolling buffer is used to compute those characteristics for all the points, as shown in figure 3 and algorithm 1. The proposed method is an efficient way to process each point using its neighbors. For each layer i, a buffer B_i is associated. Let's take the trivial case that corresponds to the middle of the scan. In this case, the buffer is filled with $2M + 1$ points where a linear regression can be applied. The middle point ($B_i[M + 1]$) can be added or removed to the final point cloud according to the acceptance criterion. Actually, a point is accepted if its distance d to the regression line is under a threshold (th_D) and if the standard deviation σ of the distance of the points to the line is under another threshold (th_σ). Those thresholds need to be carefully tuned, which will be discussed in the results section. The distance d between the point (x, y) and the line $y = a + bx$ is given by [25] :

$$d = \frac{|bx - y + a|}{\sqrt{b^2 + 1}} \tag{1}$$

Once the point is processed, the oldest point of B_i is removed and a new point is added. The linear regression is then repeated. It should be noted that all points ($2 \times M$) around

Algorithm 1: Local linear regression trivial case

Input: Point cloud X in layers ordered by azimuth angle ($\mathbf{x}_{i,k}$ the k-th point of the i-th layer)
Output: Filtered point cloud Y
$Y \leftarrow$ empty point cloud
for *each point* $\mathbf{x}_{i,k}$ ***in*** X **do**
  Add $\mathbf{x}_{i,k}$ at the end of buffer B_i
  Remove first point of (B_i)
  $(d, \sigma) =$ linear regression on point $B_i(M + 1)$
  if $d < th_D$ *and* $\sigma < th_\sigma$ **then**
    $Y \leftarrow Y \cup B_i(M + 1)$
return Y

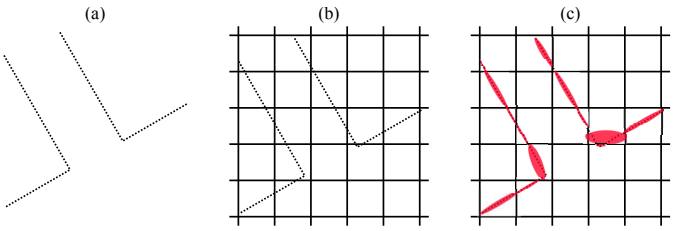

Fig. 4. Conversion of a vector map to a ND map: (a) Sampled vector map to get a point cloud, (b) Division into a grid of cells, (c) ND map with ellipsoids.

the considered one are used to compute the linear regression, even if a point was rejected in an earlier regression.

When the buffer in not full (for the first M points) and when the buffer can no longer be filled (at the end of the scan), an adaptation of the method must be made. To start with a full buffer, the first $2M + 1$ points are put in both B_i and a "beginning buffer" (figure 3). Therefore, the algorithm starts to process the $M + 1$-th point. This means that the first M points are not yet processed. At the end of the scan, the buffer B_i is progressively filled with the points contained in the "beginning buffer". In this way, all points of the scan are processed and added or not to a final point cloud.

IV. Localization using vector map

A. Vector Map

The vector map used in this work comes from the French cadastral maps, and has an accuracy between 10 cm and 20 cm. These maps are retrieved from OpenStreetMap (in France, the cadastral maps are given to OSM by the tax office)[1] which has global format. The buildings are extracted around the estimated vehicle's pose and converted into a local ENU (East North Up) coordinate system.

B. NDT with 2D vector map

The buildings from the map are converted into point cloud following a homogeneous positioning of the points along the segments. These points are positioned at equal distance from each others, with respect to a maximal distance defined as

[1]https://www.openstreetmap.org/copyright

$10\,\mathrm{cm}$ (figures 4 (a)). This step allows the direct use of the map with the NDT.

Once the LiDAR point cloud is filtered (section III), it is converted into the same ENU coordinate system as the map, using the estimated initial position.

The NDT is used to find the transformation between the filtered LiDAR point cloud and the one generated from buildings vector map, both in 2D. The buildings point cloud is considered to be the reference one, denoted Y, to be converted to Gaussian distribution. The LiDAR point cloud to be matched, is denoted X. Finally, we denote X' the point cloud X transformed to match Y.

1) Normal distribution (ND) map from the vector map: The first step of the NDT is to divide the space into a grid of cells (figure 4 (b)). It should be noted that the size of the cells affects the convergence area and the final accuracy. Therefore, after a trial and error approach, a cell of dimension $1\,\mathrm{m}$ has been chosen. For each cell k containing enough building points (at least 3 [14]), the 2D normal distribution is created (figure 4 (c)). The mean μ_k and covariance P_k of cell k are computed on the set of the points Y contained in this cell (denoted Y_k) :

$$\mu_k = \frac{1}{N_k} \sum_{\mathbf{y} \in Y_k} \mathbf{y} \tag{2}$$

$$P_k = \frac{1}{N_k - 1} \sum_{\mathbf{y} \in Y_k} (\mathbf{y} - \mu_k)(\mathbf{y} - \mu_k)^T \tag{3}$$

with N_k the number of point in Y_k. In the case that the points are aligned, the covariance matrix P_k will present a singularity and will not be invertible. This case often appears with vector maps. A correction on the smallest singular value is then applied to keep the quotient between the biggest and the smallest singular values under 10^3 [13].

2) Registration: Once the normal distribution is created, the purpose of the NDT matching is to find the rotation R and translation T between the point cloud Y and the LiDAR point cloud X. The transformed point cloud X' is defined as :

$$\mathbf{x_i}' = R\mathbf{x_i} + T \tag{4}$$

with $\mathbf{x_i} \in X$.

Since the transformations are done with respect to the building map, R and T correspond to the LiDAR pose expressed in the ENU coordinate system where building map has already been converted into.

The probability $p(\mathbf{x_i}')$ of measuring a sample at the 2D point $\mathbf{x_i}'$ contained in the cell k can be modeled as a mixture between a Gaussian distribution and a uniform distribution to be more robust against outliers. The optimal transformation between the two point clouds is obtained by minimizing the log-likelihood function:

$$-\ln L(R, T) = -\sum_{i=1}^{N_X} \ln(p(R\mathbf{x}_i + T)) \tag{5}$$

More details can be found in [16].

C. Data Fusion

Localization is done by merging the output of the NDT algorithm with the dead-reckoning of the vehicle through an Extended Kalman Filter (EKF). At instant k, the state vector is considered to be the position and the heading of the vehicle located in the middle of the rear wheel axis of the vehicle:

$$\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T \tag{6}$$

The pose obtained from NDT is converted from the LiDAR frame to the body frame and it will be considered as the NDT output. For this transformation, the heading obtained from NDT is used.

The prediction step is based on the odometric model using the wheel speed sensors and the gyro. The update step uses the output of the NDT algorithm and is given by:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - H\hat{\mathbf{x}}_{k|k-1}) \tag{7}$$

$$P_{k|k} = (I - K_k H)P_{k|k-1}(I - K_k H)^T + K_k R_k K_k^T \tag{8}$$

$$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R_k)^{-1} \tag{9}$$

with $\hat{\mathbf{x}}_{k|k}$ and $\hat{\mathbf{x}}_{k|k-1}$ the updated and predicted state vector respectively, $P_{k|k}$ and $P_{k|k-1}$ the updated and predicted covariance matrix respectively, H the observation matrix, K_k the Kalman gain, $\mathbf{z}_k$ the NDT observation and R_k the covariance associated to the observation noise β considered as white Gaussian.

The observation model associated to NDT observations is

$$\mathbf{z} = \mathbf{x} + \beta \tag{10}$$

and thus $H = I_{3\times 3}$.

Regarding the observation covariance matrix R_k, an empirical estimation is obtained by computing the error covariance between the transformed LiDAR point cloud and their nearest points in the reference point cloud (within $1\,\mathrm{m}$ maximum) :

$$R_k = \frac{1}{N_{(X',1)} - 1} \sum_{i=1}^{N_{(X',1)}} (\mathbf{x_i}' - \mathrm{NN}(\mathbf{x_i}'))\,(\mathbf{x_i}' - \mathrm{NN}(\mathbf{x_i}'))^T \tag{11}$$

where:

- $N_{(X',1)}$ is the number of points in the transformed point cloud (X') that have their nearest neighbor in the reference point cloud (Y) within $1\,\mathrm{m}$,
- and $\mathrm{NN}(\mathbf{x_i}')$ is the function that returns the Nearest Neighbor of $\mathbf{x_i}'$ in the reference point cloud Y.

It should be noted that the NDT computation takes some time compared to odometry. To avoid applying corrections of past estimation on the present ones, a rewind procedure is implemented. Therefore, the NDT output is applied at the time of the LiDAR capture and then, the prediction steps are reapplied to come back to the present time.

V. Results

For the validation of the approach, an experiment of $600\,\mathrm{m}$ was carried out in the downtown of Compiègne city (France) using an experimental vehicle equipped with wheel speed

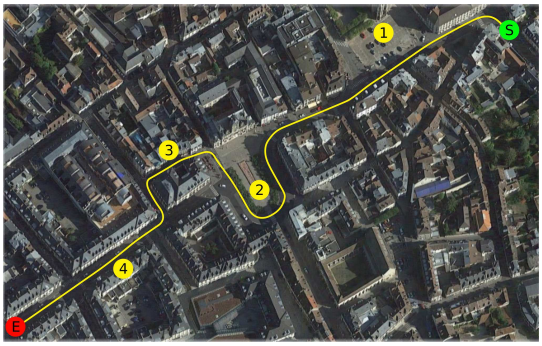

Fig. 5. Google Earth view of the path, starting at the green point: (1) open sky parking, (2) large square, (3) narrow street, (4) long straight street.

TABLE I
PRE-FILTERING PARAMETERS

Point cloud size	NDT mean error (m)	th_D	th_σ
100%	0.23	∞	∞
89%	**0.213**	**0.2**	**0.9**
48.8%	0.237	0.2	0.5
43.3%	0.307	0.3	0.3
29%	0.551	0.1	0.2
23%	0.415	0.05	0.2

sensors, a yaw rate, a 32 layers Velodyne VLP32-C, a low-cost Ublox GNSS used for comparison, and a NovAtel SPAN-CPT with post-processed kinematics (PPK) corrections used as a ground truth. Data acquisition was done at a frequency of $50\,\mathrm{Hz}$ for the dead-reckoning and $10\,\mathrm{Hz}$ for the LiDAR in an environment characterized by buildings of around 4 or 5 stories high. As shown in figure 5, the trajectory includes an open sky parking (1), a large square (2), a narrow street (3) and a long straight street (4).

The algorithm runs on an Ubuntu 18 with ROS melodic. The Point Cloud Library (PCL)[2] was used in this work for the NDT algorithm. After multiple tries, the cell size was fixed to $1\,\mathrm{m}$, the maximum number of iteration to 50 and the value used to end the convergence to $0.1\,\mathrm{m}$. The EKF presented in section IV is used.

A. LiDAR filtering effects

In this part, we show the effect of point cloud filtering on the localization accuracy. For this purpose, the local linear regression is tested with different threshold values for points acceptance applied on the distance (th_D) and the standard deviation (th_σ) as mentioned in section III. The number of points used for the regression is fixed to 31 : 15 on each side in addition to the considered point. Figure 6 shows the vector map with an example of the filtered point clouds obtained using the local linear regression.

Table I shows the performance of tests using NDT. The best parameters tuning lead to a mean error of $21.3\,\mathrm{cm}$ that increases to $23\,\mathrm{cm}$ without filtering. A similar accuracy is obtained after reducing the average size of the point cloud

[2]https://pointclouds.org

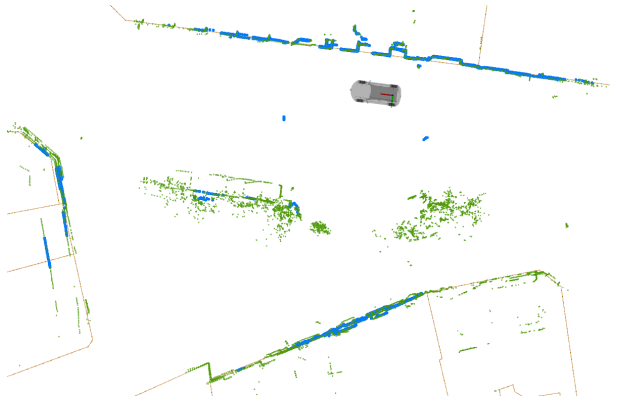

Fig. 6. Local linear regression filtering. In green the non filtered point cloud, in blue the filtered point cloud obtained using the local linear regression with $0.1\,\mathrm{m}$ as the distance threshold, $0.2\,\mathrm{m}$ as the standard deviation threshold (29% of the original size). In orange, the generated point cloud from the vector map. The car is displayed in grey.

to 48.8% of the initial size. It can be noticed that the accuracy improvement before and after filtering is not significant in our case study. However, another advantage of filtering is the point cloud size reduction, which involves a faster NDT optimization. When the distance and standard deviation thresholds decrease, the point cloud size is further reduced but the localization performance is degraded due to poor NDT associations where the error reaches $55\,\mathrm{cm}$ for a average size of the point cloud of 29%.

Regarding the RANSAC, some tests have been done and the time needed to extract multiple lines was important comparing to the linear regression. The RANSAC computation time was about 50 times longer than that of linear regression when extracting 4 to 5 lines. This delay can be explained by the large size of the point cloud which is between 15,000 and 20,000 points when considering the 12 upper layers of the Velodyne VLP32-C.

B. Localization performance

In the following, the thresholds of the linear regression filtering were tuned to $0.2\,\mathrm{m}$ and $0.9\,\mathrm{m}$ for the distance and the standard deviation respectively.

Table II shows the mean errors of the NDT. A comparison, with the low cost GNSS output is also shown. It can be seen that the mean errors of the EKF using the NDT is of $21.3\,\mathrm{cm}$, with 81% of points below $30\,\mathrm{cm}$. The obtained accuracy is very encouraging for autonomous driving. Regarding the GNSS, the accuracy is very far from the one obtained with the NDT. A comparison with the ICP was made which showed higher errors and degraded consistency.

Figure 7 shows the along track and cross track errors of NDT. The shown numbers correspond to the highlighted zones in figure 5. At the end, the NDT seems to struggle with the long straight street (4) since this area forms a corridor where the correction can only be done in the cross track direction (the part corresponding to number 4 on figure 7). This result

TABLE II
NUMERIC RESULTS

	Ublox	NDT
Mean (m)	1.664	0.213
Median (m)	1.544	0.193
Percentile at 30 cm (%)	0	81.331
95th percentile (m)	2.90	0.467
Inside 3σ (%)	100	91.1

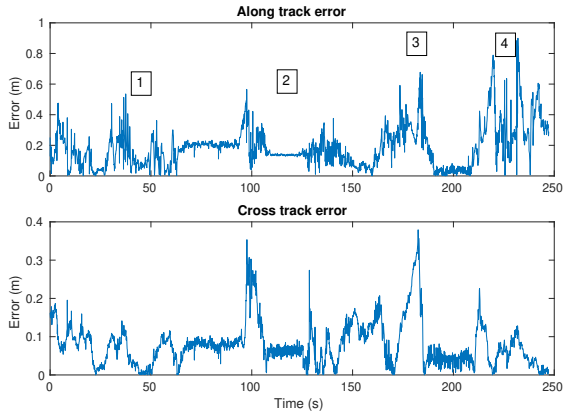

Fig. 7. Error along track and cross track.

was expected for a method which uses vector map to represent buildings.

VI. CONCLUSION

In this paper, we have proposed a map aided localization method in urban environment based on the use of cadastral maps as prior information. NDT method was used to match the LiDAR point clouds to buildings after a filtering strategy based on local linear regression to extract the buildings from the point cloud. The NDT method has shown its efficiency in dealing with such localization problem where the obtained localization accuracy was about 21 cm which is compliant with the navigation requirements of autonomous vehicles. The performance of the filtering method was studied experimentally.

In future work, in order to improve the robustness of the method, a strategy for map error detection will be added to avoid incorrect associations that will have a direct influence on the localization accuracy and convergence.

ACKNOWLEDGEMENTS

This work has been carried out within SIVALab, a joint laboratory between Renault and Heudiasyc (UTC/CNRS). It has been co-financed by the Hauts-de-France Region and Labex MS2T.

REFERENCES

[1] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*. University of North Carolina, 1997.

[2] N. Assimakis, M. Adam, and A. Douladiris, "Information Filter and Kalman Filter Comparison: Selection of the Faster Filter," *International Journal of Information Engineering (IJIE)*, vol. 2, no. 1, pp. 1–5, 2012.

[3] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous Localization and Mapping: A Survey of Current Trends in Autonomous Driving," *IEEE Trans. on Intelligent Vehicles*, pp. 194–220, Sep. 2017.

[4] W. Wen, L.-T. Hsu, and G. Zhang, "Performance Analysis of NDT-based Graph SLAM for Autonomous Vehicle in Diverse Typical Driving Scenarios of Hong Kong," *Sensors*, vol. 18, no. 11, p. 3928, Nov. 2018.

[5] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems X*. Robotics: Science and Systems Foundation, Jul. 2014.

[6] E. Javanmardi, Y. Gu, M. Javanmardi, and S. Kamijo, "Autonomous vehicle self-localization based on abstract map and multi-channel LiDAR in urban area," *IATSS Research*, vol. 43, no. 1, pp. 1–13, Apr. 2019.

[7] P. J. Besl and N. D. McKay, "Method for registration of 3-D shapes," in *Sensor Fusion IV: Control Paradigms and Data Structures*, vol. 1611. International Society for Optics and Photonics, Apr. 1992, pp. 586–606.

[8] Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, Apr. 1992.

[9] A. Segal, D. Haehnel, and S. Thrun, "Generalized ICP," *Robotics: science and systems*, vol. 2, no. 4, p. 435, 2009.

[10] G. Agamennoni, S. Fontana, R. Y. Siegwart, and D. G. Sorrenti, "Point Clouds Registration with Probabilistic Data Association," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2016, pp. 4092–4098.

[11] O. Vysotska and C. Stachniss, "Improving SLAM by Exploiting Building Information from Publicly Available Maps and Localization Priors," *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, vol. 85, no. 1, pp. 53–65, Feb. 2017.

[12] G. Bresson, Z. Alsayed, and S. Jonchery, "Graph-based Map-Aided Localization using Cadastral Maps as Virtual Laser Scans," in *IEEE Intelligent Transportation Systems Conference*, 2019, pp. 4074–4080.

[13] P. Biber and W. Strasser, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, Oct. 2003, pp. 2743–2748.

[14] P. Biber, S. Fleck, and W. Strasser, "A probabilistic framework for robust and accurate matching of point clouds," in *Joint Pattern Recognition Symposium*, 2004, pp. 480–487.

[15] E. Takeuchi and T. Tsubouchi, "A 3-D Scan Matching using Improved 3-D Normal Distributions Transform for Mobile Robotic Mapping," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2006, pp. 3068–3073.

[16] M. Magnusson, "The three-dimensional normal-distributions transform: An efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro University, 2009.

[17] A. Das and S. L. Waslander, "Scan registration with multi-scale k-means normal distributions transform," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 2705–2710.

[18] C. Ulas and H. Temeltas, "A 3D Scan Matching Method Based On Multi-Layered Normal Distribution Transform," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 602–11 607, Jan. 2011.

[19] M. Magnusson, N. Vaskevicius, T. Stoyanov, K. Pathak, and A. Birk, "Beyond points: Evaluating recent 3D scan-matching algorithms," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 3631–3637.

[20] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[21] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for Point-Cloud Shape Detection," *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, 2007.

[22] P. V. C. Hough, "Method and means for recognizing complex patterns," US Patent US3 069 654A, Dec., 1962.

[23] Y. Lin, C. Wang, B. Chen, D. Zai, and J. Li, "Facet Segmentation-Based Line Segment Extraction for Large-Scale Point Clouds," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 9, pp. 4839–4854, Sep. 2017.

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

[25] B. Spain, *Analytical Conics*. Courier Corporation, Jan. 2007.