2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Precise Point Set Registration Using Point-to-Plane Distance and Correntropy for LiDAR Based Localization

Guanglin Xu[1], Shaoyi Du[1]*, Dixiao Cui[1], Sirui Zhang[2], Badong Chen[1], Xuetao Zhang[1], Jianru Xue[1], Yue Gao[3]*

***Abstract*— In this paper, we propose a robust point set registration algorithm which combines correntropy and point-to-plane distance, which can register rigid point sets with noises and outliers. Firstly, as correntropy performs well in handling data with non-Gaussian noises, we introduce it to model rigid point set registration problem based on point-to-plane distance; Secondly, we propose an iterative algorithm to solve this problem, which repeats to compute correspondence and transformation parameters respectively in closed form solutions. Simulated experimental results demonstrate the high precision and robustness of the proposed algorithm. In addition, LiDAR based localization experiments on automated vehicle performs satisfactory for localization accuracy and time consumption.**

I. Introduction

Point set registration is one of the most important research issues in computer vision, pattern recognition, robotics and many other fields. There are many algorithms that can solve the registration problem, which could be divided into two classes. The first one is point-wise algorithms, in which the correspondence between two point sets is established by building up one-to-one links, such as iterative closest point (ICP) algorithm [1] and its varied algorithms. The second one is full correspondence based algorithms, where the point sets are represented as Gaussian mixture models (GMMs) and the correspondence are full connection, such as coherent point drift (CPD) [2] and GMM-L2 based method [3] , etc. In practice, most of GMM based algorithms consume too much time and it is difficult to be applied to localization problems, which need a good performance in real-time efficiency.

In this paper, to get a better performance as well as efficiency, we use the first way to solve the registration problem. The ICP algorithm is well-known for its accuracy and efficiency and more and more related work has been done for the robustness and speed of the ICP algorithm. Chetverikov et al. [4] proposed the trimmed ICP (TrICP) algorithm to handle partially overlapping problem. Granger et al. [5] proposed a coarse-to-fine method to speed up the algorithm. Du et al. [6] proposed the probability ICP algorithm for point sets with noises. However, the methods mentioned above depend on the parameters greatly and can't handle outliers and noises at the meantime. In addition, these methods do not work well when dealing with a large-scale map in LiDAR based localization problems.

Chen et al. [7] introduced point-to-plane distance to the registration problem. This algorithm is sensitive to noises and outliers even though surface normal information is introduced. Segal et al. [8] proposed the generalized-ICP algorithm, which uses the plane-to-plane distance to increase the robustness of the algorithm. However, the algorithm has a high computational complexity and it's hard to meet the need of time consumption of the localization problem. In this paper, we also introduce surface normal information, and propose a new model based on correntropy [9] .On this basis, we propose a novel algorithm to solve the registration problem. Experimental results show that the algorithm performs accurately and efficiently.

The rest of this paper is organized as follows. In section II, point set registration problem and correntropy are stated briefly. In section III, an optimization model of the registration problem based on point-to-plane distance and correntropy is proposed, and then the proposed algorithm is given. In section IV, some simulation experiments and two LiDAR based localization experiments are shown. Finally, a conclusion is drawn in the last section.

II. Background

A. The ICP Algorithm

In this part, we will take a brief review of the registration problem and the ICP algorithm. Assume that there are two point sets in $\mathbb{R}^m$, a source point set $S \triangleq \{\vec{s}_i\}_{i=1}^{N_s}(N_s \in \mathbb{N})$ and a destination point set $D \triangleq \{\vec{d}_i\}_{i=1}^{N_d}(N_d \in \mathbb{N})$, the goal of rigid registration is to find a transformation function $T(\cdot)$, with which the source point set S can be in the best alignment with destination point set D under a certain similarity criterion.

In the traditional point set registration model, the similarity is estimated by squared Euclidean distance, and the objective function is formulated as:

$$\min_{\mathbf{R},\vec{t},j\in\{1,2,\ldots,N_y\}} \sum_{i=1}^{N_x} ||(\mathbf{R}\vec{x}_i + \vec{t}) - \vec{y}_j||_2^2 \quad s.t. \quad \mathbf{R}^T\mathbf{R} = \mathbf{I}_m,\ \det(\mathbf{R}) = 1. \tag{1}$$

The ICP algorithm iteratively calculates $\mathbf{R}$ and $\vec{t}$. In the k^{th} iteration, two steps are included. Firstly, set up correspon-

[1]Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, Xi'an, China.
[2]School of Software Engineering, Xi'an Jiaotong University, Xi'an, China.
[3]School of Software, Tsinghua University, Beijing, China.
[1]*Shaoyi Du: dushaoyi@xjtu.edu.cn
[3]*Yue Gao: kevin.gaoy@gmail.com

dence between two point sets:

$$c_k(i) = \underset{j\in\{1,2,...,N_d\}}{\arg\min} (||(\mathbf{R}_{k-1}\vec{s}_i+\vec{t}_{k-1})-\vec{d}_j||_2^2) \text{ for } i = 1, ..., N_s \tag{2}$$

Secondly, compute $\mathbf{R}_k$ and $\vec{t}_k$ by minimizing squared distance:

$$(\mathbf{R}_k, \vec{t}_k) = \underset{\mathbf{R}^T\mathbf{R}=\mathbf{I}_n, \det(\mathbf{R})=1, \vec{t}}{\arg\min} (\sum_{i=1}^{N_s} ||\mathbf{R}\vec{s}_i + \vec{t} - \vec{d}_{c_k(i)}||_2^2) \tag{3}$$

The ICP algorithm runs iteratively until the registration error converges to a local minimum or the iteration number reaches the maximum. The ICP algorithm is fast but it can be easily affected by noises or outliers.

B. Correntropy

Given two arbitrary scalar random variables *X* and *Y*, a general form of correntropy between *X* and *Y* is defined as:

$$V_\sigma(X, Y) = \mathbf{E}[\frac{1}{\sqrt{2\pi}\sigma} \exp(-\frac{(X-Y)^2}{2\sigma^2})] \tag{4}$$

In most cases, the joint probability density function (pdf) is unknown and a finite number of data $\{(x_i, y_i)\}_{i=1}^N$ are available, thus a sample estimator of correntropy is drawn as:

$$\hat{V}_{N,\sigma}(X, Y) = \frac{1}{\sqrt{2\pi}\sigma N} \sum_{i=1}^{N} \exp(-\frac{(x_i-y_i)^2}{2\sigma^2}) \tag{5}$$

A basic property is that correntropy is positive and bounded, and it reaches its maximum $1/\sqrt{2\pi}\sigma$ if and only if $X = Y$. Correntropy has a good performance on resisting noises and outliers, that's because the correntropy of inliers will acquire a value close to the maximum while the outliers will be infinitely close to 0. That means the outliers or noises will be assigned a lower weight, which is influenced by their deviation and the parameter σ.

III. Precise Rigid Registration Using Point-to-Plan Distance and Correntropy

In this section, a mathematical model is established for the point set registration problem, where point-to-plane distance is taken into consideration based on maximum correntropy criterion. After that, our algorithm is proposed in an iterative way.

A. Problem Statement

The traditional point-to-point distance consider little about the surface normal information of the point sets, so that the structured parts like plane or curved surface cannot help to get a better registration result. Especially in 3D or 2D LiDAR scanned data, the walls and other similar elements are quite usual, and physical points in the environment cannot be guaranteed to be sampled continuously. However, the local plane that these points belonging to is stable in a series of consecutive frames. Therefore, we introduce point-to-plane distance to the registration problem so as to use the surface normal information. For each point $\vec{s}_i$ in S , the point-to-plane distance is defined as follows:

$$dist_{pt2pl} = ||(\vec{s}_i - \vec{d}_{c(i)}) \cdot \vec{n}_{c(i)}||^2, \tag{6}$$

where $\vec{d}_{c(i)}$ matches $\vec{s}_i$, $c(\cdot)$ is corresponding function, $\vec{n}_{c(i)}$ is normal vector at $\vec{d}_{c(i)}$ and "$\cdot$" represents the dot product of two vectors. In this paper, for each point in D , we use principle component analysis (PCA) of its 10 closest points to estimate its surface normal, where the eigenvector corresponding to the minimum eigenvalue is considered as the surface normal vector at $\vec{d}_i$.

With point-to-plane distance, we define the similarity of two point sets by means of correntropy because of its excellent resistance to noises. The cost function of the rigid registration is defined as follows:

$$H = \frac{1}{\sqrt{2\pi}\sigma} \sum_{i=1}^{N_s} \exp(-\frac{((T(\vec{s}_i) - \vec{d}_{c(i)}) \cdot \vec{n}_{c(i)})^2}{2\sigma^2}), \tag{7}$$

where $\vec{d}_{c(i)}$ matches $\vec{s}_i$, and $\vec{n}_{c(i)}$ is normal vector at $\vec{d}_{c(i)}$.

To minimize the error of registration, the objective function is shown as follows, which should maximize the sum of correntropy:

$$J = \max_{T, c(i)\in\{1,...,N_y\}} \sum_{i=1}^{N_s} \exp(-\frac{((T(\vec{s}_i) - \vec{d}_{c(i)}) \cdot \vec{n}_{c(i)})^2}{2\sigma^2}). \tag{8}$$

The constant term $1/\sqrt{2\pi}\sigma$ in (7) can be ignored because it does not influence the optimization result of (8). Moreover, we mainly discuss the rigid registration problem in this paper, so T is a rigid transformation.

B. The Proposed Algorithm

In the objective function (8), both correspondence between two point sets and the rigid transformation are unknown. In this paper, an iterative algorithm is proposed, which includes two steps in each iteration.

Step 1. Assuming that the $(k-1)^{th}$ rigid transformation is known, we set up the k^{th} correspondence between two point sets by finding closest points:

$$c_k(i) = \underset{c(i)\in\{1,2,...,N_s\}}{\arg\min} (||T(\vec{s}_i) - \vec{d}_{c(i)}||_2^2) \text{ for } i = 1, ..., N_s. \tag{9}$$

In step 1, there are many algorithms can speed up this search process, such as k-d tree [10] and k-nearest neighbor search [11], etc. In this paper, we use the latter one to find the closest point and build up correspondence for its efficiency.

Step 2. Computing the k^{th} rigid transformation according to the known correspondence of step 1:

$$T_k = \arg\max_T \sum_{i=1}^{N_s} \exp(-\frac{((T(\vec{s}_i) - \vec{d}_{c_k(i)}) \cdot \vec{n}_{c_k(i)})^2}{2\sigma^2}). \tag{10}$$

In (10), there are mainly two different ways to express a rigid transformation. One is an algebraic approach, in which $T(\vec{s}_i)$ is expressed as $\mathbf{R}\vec{s}_i + \vec{t}$, where the constraints are $\mathbf{R}^T\mathbf{R} = \mathbf{I}_n$ and $det(\mathbf{R}) = 1$. In this way, we

can compute rotation matrix $\mathbf{R}$ and translation vector $\vec{t}$ directly by Lagrange multiplier method and singular value decomposition (SVD). Another way is a geometric approach, where we consider that $T(\cdot)$ is a function of Euler angles and translation vector. In this paper, we choose the second method and it will be introduced in the next section.

The algorithm repeats the iteration until the registration error is small enough or the iteration number k reaches the maximum. It's easy to prove that this algorithm is local convergent.

C. Solve the Transformation

As mentioned above, we consider that 2D point is first rotated by one Euler angle α and 3D point cloud is by three Euler angles α, β and γ, and then translated by a translation vector. What we should solve are three parameters $[\alpha, t_x, t_y]^T$ in 2D and six parameters $[\alpha, \beta, \gamma, t_x, t_y, t_z]^T$ in 3D cases respectively. By this means, the rigid constraint is implicit in the rotation matrix derived by Euler angles. Since there are some differences in 2D point sets and 3D point sets, the solving process will be introduced respectively in the following parts.

a) For 2D point sets: In 2D point sets, transformation function $T(\vec{s}_i)$ can be defined as:

$$T(\vec{s}_i) = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \begin{bmatrix} s_{ix} \\ s_{iy} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}, \quad (11)$$

where α is the rotation radians. Substituting (11) into (10), the objective function becomes a nonlinear function. Algorithms such as LM method [12], [13] can solve this kind of question. However, to speed up the solving process, we suppose the rotation angle is small, so that the nonlinear problem can be linearized approximately, i.e., we suppose that $\cos\alpha \approx 1$ and $\sin\alpha \approx \alpha$. In this way, $(T(\vec{s}_i) - \vec{d}_{c_k(i)}) \cdot \vec{n}_{c_k(i)}$ in (10) can be simplified as:

$$\left(\begin{bmatrix} 1 & -\alpha \\ \alpha & 1 \end{bmatrix} \begin{bmatrix} s_{ix} \\ s_{iy} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} d_{c_k(i)x} \\ d_{c_k(i)y} \end{bmatrix} \right) \cdot \begin{bmatrix} n_{c_k(i)x} \\ n_{c_k(i)y} \end{bmatrix}. \quad (12)$$

Furthermore, we define $\vec{x} = [\alpha, t_x, t_y]^T$ as the variable to be solved, the objective function (10) can be simplified as:

$$J = \max_x \sum_{i=1}^{N_s} \exp\left(-\frac{(\vec{A}_i\vec{x} - b_i)^2}{2\sigma^2}\right), \quad (13)$$

where $\vec{A}_i = [\ s_{ix}n_{c_k(i)y} - s_{iy}n_{c_k(i)x} \quad n_{c_k(i)x} \quad n_{c_k(i)y}\ ]$ and $b_i = (\vec{s}_i - \vec{d}_{c_k(i)}) \cdot \vec{n}_{c_k(i)}$. Taking derivation of J with respect to x , we can get:

$$\frac{\partial J}{\partial \vec{x}} = \sum_{i=1}^{N_s} \exp\left(-\frac{(\vec{A}_i\vec{x} - b_i)^2}{2\sigma^2}\right) \frac{\vec{A}_i\vec{x} - b_i}{-\sigma^2} \vec{A}_i. \quad (14)$$

There is an exponential term in (14), which will impede the solving of $\vec{x}$, so in each iteration, we substitute the $\vec{x}$ in the exponential term with the temporary result $\vec{x}_{k-1}$ at the last iterative step, and denote the exponential part as follows:

$$g_i = \exp\left(-\frac{(\vec{A}_i\vec{x}_{k-1} - b_i)^2}{2\sigma^2}\right). \quad (15)$$

With (15), the function (14) can be rewritten as:

$$\frac{\partial J}{\partial \vec{x}} = -\frac{1}{\sigma^2}(\mathbf{A}\vec{x} - \vec{b})^T \mathbf{D}(\vec{g})\mathbf{A}, \quad (16)$$

where $\mathbf{A} = [\vec{A}_1, ..., \vec{A}_{N_s}]^T$, $\vec{b} = [b_1, ..., b_{N_s}]^T$, $\vec{g} = [g_1, ..., g_{N_s}]^T$ and $\mathbf{D}(\vec{g}) = diag(\vec{g})$. Let the derivation be 0, we have the following solution for $\vec{x}$:

$$\vec{x} = (\mathbf{A}^T\mathbf{D}(\vec{g})\mathbf{A})^{-1}\mathbf{A}^T\mathbf{D}(\vec{g})\vec{b}. \quad (17)$$

Equation (17) gives a closed form solution of $\vec{x}$ in the k^{th} iteration, the algorithm should iteratively calculate correspondence and $\vec{x}$ till its convergence.

b) For 3D point sets: In 3D point sets registration, $\vec{x}$ is denoted as:

$$\vec{x} = [\alpha, \beta, \gamma, t_x, t_y, t_z]^T \quad (18)$$

According to Lie algebra [14], when α , β , and γ are small, we can get an approximate result of $T(\vec{s}_i)$ as follows:

$$T(\vec{s}_i) = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix} \begin{bmatrix} s_{ix} \\ s_{iy} \\ s_{iz} \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \quad (19)$$

Substitute $T(\cdot)$ in (10) with (19) and take derivation with respect to $\vec{x}$, we can get a similar result to (14), where $\vec{A}_i$ have a different form:

$$\vec{A}_i = [a_{i1}, a_{i2}, a_{i3}, \eta_{c_k(i)x}, \eta_{c_k(i)y}, \eta_{c_k(i)z}], \quad (20)$$

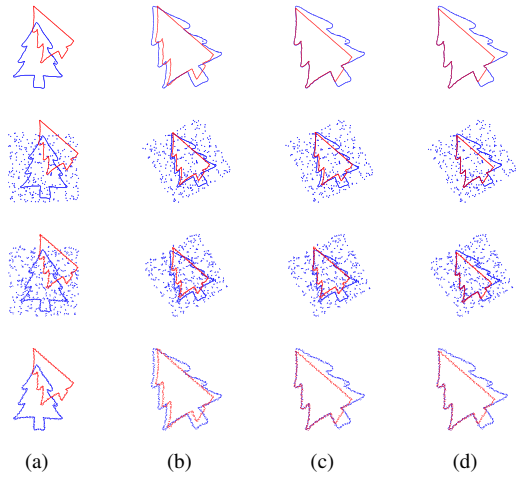
with $a_{i1}, a_{i2}, a_{i3}] = \vec{s}_i \times \vec{n}_{c_k(i)}$. Rewrite the sum function in a matrix formulation, we can get a similar equation as (16), where $\mathbf{A}$ is a $N_s \times 6$ matrix and $\vec{b}$ is a $N_s \times 1$ vector. Let the derivation be 0, we get the following solution:

$$\vec{x} = (\mathbf{A}^T\mathbf{D}(\vec{g})\mathbf{A})^{-1}\mathbf{A}^T\mathbf{D}(\vec{g})\vec{b}. \quad (21)$$

From above, we can generally sum up that the solutions of transformation parameter are quite similar in 2D and 3D point set registration and there is only a little difference in the definition of $\mathbf{A}$. In practice, the parameter σ changes in each step. At the beginning, is recommended to be a large number to speed up the convergence, and in the following iterations, σ should be smaller with simulated annealing algorithm because a smaller σ leads to a better ability to resist noises. However, σ cannot be too small to prevent the matrix from being singular. An exercisable method to decide σ is proposed as follows: for each point in the destination point set, estimating the distance between itself and its nearest neighbour, and setting 30 times median distance as the initial value of σ; with the iteration goes on, σ gets smaller but with a lower limit of 3 or 5 times median distance.

IV. Experimental Result

To prove the good performance of the proposed algorithm, we test the algorithm on some widely used public datasets. Moreover, the algorithm is applied to deal with LiDAR based localization. The proposed algorithm is implemented by MATLAB and run on PC with Intel Core i7-6700 CPU @ 3.40GHz and 16G RAM.



(a) (b) (c) (d)

Fig. 1. Compared 2D simulation results. (a) The source set (blue) and the model set (red). (b) The results of ICP algorithm. (c) The results of ICP algorithm based on MCC. (d) The registration results of proposed algorithm.

TABLE I
REGISTRATION ERROR OF 2D SIMULATIONS

Algorithm		Proposed algorithm	ICP based on MCC	ICP
Rotation	$\varepsilon_{\mathbf{R}}$	0.0010	0.0002	**0.0003**
	$\varepsilon_{\vec{t}}$	**0.1266**	0.9806	12.7572
Salt & Pepper	$\varepsilon_{\mathbf{R}}$	**0.0024**	0.0995	0.0294
	$\varepsilon_{\vec{t}}$	**0.2592**	7.9054	23.8770
Gaussian	$\varepsilon_{\mathbf{R}}$	0.0098	0.0060	**0.0013**
	$\varepsilon_{\vec{t}}$	**1.6687**	1.7484	12.6498
Gaussian & Salt	$\varepsilon_{\mathbf{R}}$	**0.0081**	0.0934	0.0160
	$\varepsilon_{\vec{t}}$	**1.8434**	5.5717	18.0897

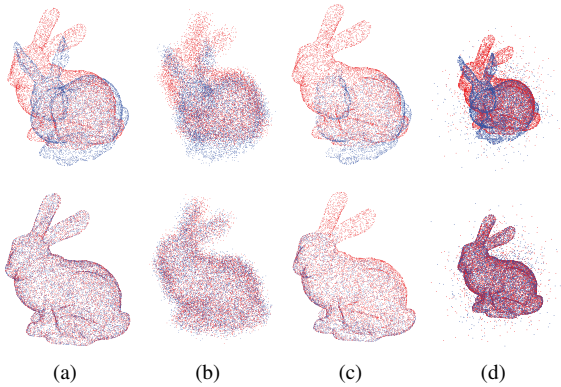
A. Simulations

In this part, we use both 2D images and 3D point clouds for simulation experiments.

c) 2D simulations: We choose a tree image in part B of CE-shape-1 [15] for 2D simulation, the image undergoes several procedures to generate the test data, which are shown in Fig. 1. In the first column, the model set is cut from the source set and it is rotated 30 around the origin, the overlapping ration of these two point sets is 48%; in the second column, 1% salt & pepper noises are added on the source set; in the third column, 1dBW white Gaussian noises are added on the both sets and in the last column, both kind of noises mentioned above are added.

In this part we compare the proposed algorithm with ICP algorithm and ICP algorithm based on MCC [16], where correntropy is introduced with point-to-point distance.

To quantize the registration error, we define $\varepsilon_{\mathbf{R}} = ||\mathbf{R} - \mathbf{R}_{gt}||_2$ and $\varepsilon_{\vec{t}} = ||\vec{t} - \vec{t}_{gt}||_2$, where $\mathbf{R}_{gt}$ and $\vec{t}_{gt}$ are the ground truth while $\mathbf{R}$ and $\vec{t}$ are computation results of the algorithms. Table I shows the detailed registration error of 2D simulations. It's obvious that the proposed algorithm has a better registration accuracy.



(a) (b) (c) (d)

Fig. 2. 3D registration results on "bunny" point set. (a) Random down sampled point set. (b) Point set with noises. (c) Point set with missing parts. (d) Point set with outliers.

TABLE II
REGISTRATION ERROR OF 3D SIMULATIONS

Error	Down sample	Noises	Missing parts	Outliers
$\varepsilon_{\mathbf{R}}$	0.000286	0.077707	0.000639	0.004031
$\varepsilon_{\vec{t}}$	2.15E-05	0.000363	7.67E-05	0.000333

d) 3D simulations: For 3D simulate experiment, we choose reconstructed 3D bunny point set from the Stanford 3D scanning repository [17]. The experiment results are shown in Fig. 2. In the column (a), two sets are generated by randomly down sampling the bunny model, where the sample ratio is 20%. On this basis, in column (b), random noises are added on each point in both sets, which shows the robustness to noise of our algorithm; (c) shows the partial registration results and (d) proofs the robustness to outliers. The quantized error of rotation matrix $\mathbf{R}$ and translation vector $\vec{t}$ is shown in Table II, which shows the robustness and efficiency of the proposed algorithm.

B. LiDAR based Localization

Both point-to-line distance and point-to-plane distance are helpful in autonomous vehicle. For example, 2D point set registration can be used in curb-based drivable zone detection on the road and 3D point set registration is widely used in SLAM problem. In this part, we test our algorithm on two different data sets to test the ability on LiDAR based localization. Firstly, we test our algorithm on KITTI odometry data sets [18]. Here we only use 3D raw point data captured by Velodyne LiDAR to get the vehicle's pose and attitude, and the data is captured in outdoor open space with RTK-GPS ground truth. Secondly, we test the algorithm on our intelligent vehicle platform while the data is captured in an underground parking lot. The following two parts will introduce the experiments respectively.

TABLE III
THE LOCALIZATION RESULTS OF DIFFERENT ALGORITHMS

	Our algorithm	ICP algorithm	Generalized ICP	NDT algorithm
Frames	1349	1227	1305	454
Ratio	0.8916	0.8110	0.8625	0.3001

e) Outdoor localization using KITTI datasets: The KITTI odometry benchmark includes 22 LiDAR data sequences and sequences 00-10 are provided with ground truth. In this section we will register raw point clouds in "sequence 00", not only because there are enough frames but also it is captured in urban, where there are many structured objects on roadside which can be helpful to localization. In addition, each frame is registered with the third frame after it.

We compared the proposed algorithm with point-to-plane ICP algorithm, NDT algorithm and generalized ICP algorithm, which all are implemented in Point Cloud Library (PCL)[1]. To speed up the registration, the scanned points are randomly down sampled to 10% of the original point sets. In each registration task, the initial values are set to $[\alpha, \beta, \gamma] = [0, 0, 0]$ and $\vec{t} = [0; 0; 0]$.

To compare the localization accuracy, we define an error as $\varepsilon_\theta = \arccos((trace(\mathbf{R}_{gt}^{-1}\mathbf{R}) - 1)/2)$ to estimate the error of Euler angle, it can express the attitude angle difference between the ground truth and the computation result. In addition, the error of translation is still defined as $\varepsilon_{\vec{t}} = ||\vec{t} - \vec{t}_{gt}||_2$. Besides, for each frame, if its angular error is less than 1° and the translation error is less than 0.5 meters, we consider that this frame is correctly registered. Under this criterion, we count the correct frames and calculate the correct ratio and the results are shown in Table III. In addition, among all of the correctly registered frames, we estimate the mean error and standard deviation (std) of angle and translation vector, which is shown in Fig. 3. Fig. 3 demonstrates the good performance of the proposed algorithm.

For more detailed results, such as the registration procedure and the cases that the algorithm fails, please see the attached video.

f) Indoor localization on autonomous vehicle: In this part, we test our algorithm on a sequence of 3D LiDAR scanned data without GPS ground truth. It is captured by KUAFU-1 intelligent vehicle in a parking lot, where the car is displayed in Fig. 4. It is equipped with various sensors such as LiDAR, radar, camera and GPS-IMU integrated navigation system.

Before we test the algorithm, a map of the underground parking lot is established using graph SLAM method and it is shown in Fig. 5. The parking lot's entrance is on the ground and the parking area is under ground. The trajectory of the vehicle is that the vehicle started at the entrance,

[1]http://www.pointclouds.org

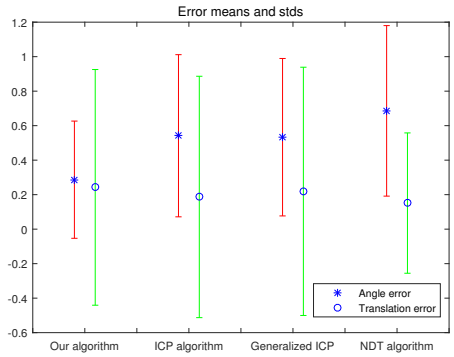


Fig. 3. The mean error and std of rotation angle and translation vector.

Fig. 4. The intelligent vehicle platform KUAFU-1.

down to the parking lot through a slope and finally came out of the parking lot. At the starting point, GPS signal is available which is used as the initial value and in the following frames, the initial value is set as the registration result of last frame.Beyond that, there are no more relations in closed two frames.

Note that the data for creating the map are different from the data we use for localization, cars in the parking lot are placed in different positions which can be seen as outliers in the process of registration. The localization result is shown in Fig. 6, from which we can see that the trajectory is continuous and smooth. Besides, at the terminal point of the trajectory, the localization error is less than ± 0.1 m with respect to the GPS result. And for more detailed results, please see the attached video.

V. Conclusion

In this paper, we propose a novel point set registration model by introducing correntropy and point-to-plane distance at the same time. After that, an iterative algorithm is proposed to solve the problem. Experimental results show that the proposed algorithm performs fast and precisely in 2D and 3D registration. Specifically, this algorithm is suit for LiDAR based localization in urban, school, indoor environments, etc., where the environments that contains enough planes of different orientation, even if there are dynamic obstacles.

Acknowledgment

This work was supported by the National Natural Science Foundation of China under Grant Nos. 61573274, 61627811 and 61751308, and the Fundamental Research Funds for the Central Universities under Grant Nos. xjj2017005 and

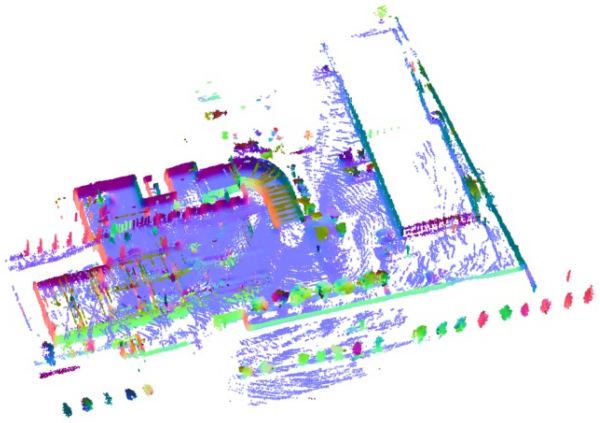


Fig. 5. 3D cloud map of the parking lot.

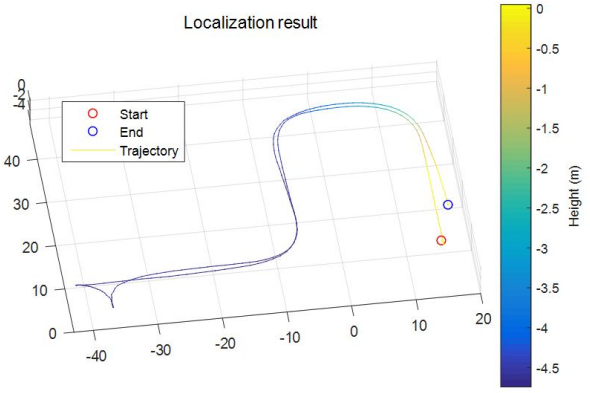


Fig. 6. The localization result by the proposed algorithm.

xjj2017067. In addition, thanks for the help about the experiments of Di Wang and Zhongxing Tao.

REFERENCES

[1] P. J. Besl and N. D. Mckay, "A method for registration of 3-d shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 2002.

[2] A. Myronenko and X. Song, "Point set registration: coherent point drift," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 32, no. 12, p. 2262, 2010.

[3] B. Jian and B. C. Vemuri, "Robust point set registration using gaussian mixture models," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 33, no. 8, pp. 1633–1645, 2011.

[4] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *International Conference on Pattern Recognition, 2002. Proceedings*, 2002, pp. 545–548 vol.3.

[5] S. Granger and X. Pennec, "Multi-scale em-icp: A fast and robust approach for surface registration," in *Computer Vision - ECCV 2002, European Conference on Computer Vision, Copenhagen, Denmark, May 28-31, 2002, Proceedings*, 2002, pp. 418–432.

[6] S. Du, J. Liu, C. Zhang, J. Zhu, and K. Li, "Probability iterative closest point algorithm for m -d point set registration with noise," *Neurocomputing*, vol. 157, no. CAC, pp. 187–198, 2015.

[7] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Image & Vision Computing*, vol. 10, no. 3, pp. 145–155, 1991.

[8] A. Segal, D. Hhnel, and S. Thrun, "Generalized-icp," 2009.

[9] W. Liu, P. P. Pokharel, and J. C. Principe, "Correntropy: Properties and applications in non-gaussian signal processing," *IEEE Transactions on Signal Processing*, vol. 55, no. 11, pp. 5286–5298, 2007.

[10] J. L. Bentley, "K-d trees for semidynamic point sets," in *Symposium on Computational Geometry*, 1990, pp. 187–197.

[11] Friedman, J. H, Bentley, J. Louis, Finkel, and R. Ari, "An algorithm for finding best matches in logarithmic expected time," *Acm Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, 1977.

[12] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Journal of Heart & Lung Transplantation the Official Publication of the International Society for Heart Transplantation*, vol. 2, no. 4, pp. 436–438, 1944.

[13] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[14] V. S. Varadarajan, *Lie Groups, Lie Algebras, and Their Representations*. Prentice-Hall, Inc, 1974.

[15] L. J. Latecki, R. Lakamper, and T. Eckhardt, "Shape descriptors for non-rigid shapes with a single closed contour," in *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, 2000, pp. 424–429 vol.1.

[16] G. Xu, S. Du, and J. Xue, "Precise 2d point set registration using iterative closest algorithm and correntropy," in *International Joint Conference on Neural Networks*, 2016, pp. 4627–4631.

[17] G. Turk and M. Levoy, "Zippered polygon meshes from range data," *Computers & Graphics*, pp. 311–318, 1994.

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