Robotic Online Path Planning on Point Cloud

Ming Liu, *Member, IEEE*

***Abstract*—This paper deals with the path-planning problem for mobile wheeled- or tracked-robot which drive in 2.5-D environments, where the traversable surface is usually considered as a 2-D-manifold embedded in a 3-D ambient space. Specially, we aim at solving the 2.5-D navigation problem using raw point cloud as input. The proposed method is independent of traditional surface parametrization or reconstruction methods, such as a meshing process, which generally has high-computational complexity. Instead, we utilize the output of 3-D tensor voting framework on the raw point clouds. The computation of tensor voting is accelerated by optimized implementation on graphics computation unit. Based on the tensor voting results, a novel local Riemannian metric is defined using the saliency components, which helps the modeling of the latent traversable surface. Using the proposed metric, we prove that the geodesic in the 3-D tensor space leads to rational path-planning results by experiments. Compared to traditional methods, the results reveal the advantages of the proposed method in terms of smoothing the robot maneuver while considering the minimum travel distance.**

***Index Terms*—Mobile robots, navigation, path planning.**

I. Introduction

THE DEVELOPMENT of robotics is always inspired by human experience and activities [1]. Taking the typical scenario shown in Fig. 1 for instance: an elder man is trying to fetch the blue cup which is filled with his favorite coffee. With rational consideration, he probably would take the cyan (light color) detour rather than the red (dark color) bumpy path. For this situation, it hardly makes sense to take the red path, even though the red path leads to an accumulated shorter distance.

Derived from that, such a concept is usually extended to robotic planning for navigation, using various cost or potential functions [2] depending on selected criteria or applications. The generated path ought to not only regard the shortest Euclidean distance, but also ease the maneuver.

A. Robotic Challenges

Most existing navigation algorithms for mobile robots assume that the configuration space of the traversable path lies on a developable surface [3], [4]. It means that the map can be simply considered as a 2-D plane [5] without distortion, or directly projected to a 2-D plane without information loss. It is originated from the widely applied representations derived by mature 2-D SLAM techniques [6], such as filtering-based methods [7], [8] or pose-graph-based methods [9]. Such an assumption exempts the requirement on detailed analysis of the local shape of the terrain. However, the robot ought to deal with these dynamics introduced from the 3-D terrain shape, considering the complexity of the real environment [10].

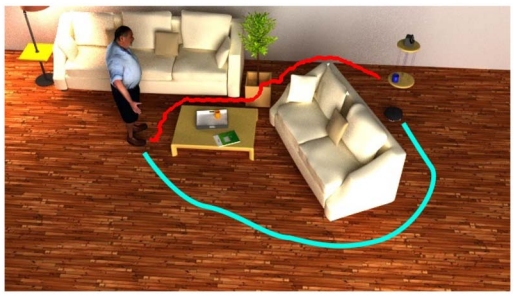


Fig. 1. Typical scenario that implies rational planning.

In recent years, the fast development of 3-D mapping techniques [11], [12] and sensors [13] enables the modeling of the multiterrain environment [14] for practical applications [15], [16]. As the raw output from these 3-D mapping techniques, point cloud has been widely studied. The following major aspects make the analysis on raw point cloud a challenging problem, let alone the navigation on it.

1) *Unreliability of Observation:* The unreliability is multifold. In Fig. 2, we show a cropped part of point cloud observed from an indoor semi-structured environment. Outliers, missing points and nonuniform distribution of points are the major drawbacks in real applications.
2) *Large Amount of Sparse Data:* The 3-D sensor usually generates a large amount of points per scan. However, points are not continuously defined in the 3-D space. The missing information among points, especially latent structural information, must be recovered by subtle filters.
3) *Computational Complexity:* Due to the large amount of data as shown in Fig. 2, the high-computational cost is a bottleneck for most applications.

Because of these challenges, especially the lack of structural information, many works tried to avoid direct operations on points, leading to various representations, e.g., meshed surface [17], [18] or tree-based structures [11], which were inspired from topological segmentation of 2-D environments [19]–[21]. Despite these difficulties, here we

Manuscript received January 7, 2015; revised April 22, 2015; accepted May 4, 2015. Date of publication May 20, 2015; date of current version April 13, 2016. This work was supported in part by the Hong Kong University of Science and Technology under Project IGN13EG03, in part by the General Research Fund by Research Grants Council Hong Kong under Grant 16206014, and in part by the National Natural Science Foundation of China under Grant 6140021318. This paper was recommended by Associate Editor S. X. Yang.

The author is with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong (e-mail: liu.ming.prc@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2015.2430526

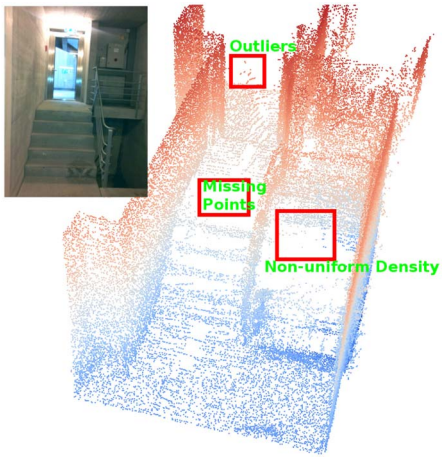


Fig. 2. Clip of typical point cloud and common observation noises.

tackle the navigation problem as manifold modeling, by using raw point cloud as input. It is because we aim at solving the related problems in real time, and the raw point cloud is the most readily available representation. In order to cope with the large amount of sparse data, we adopt tensor voting framework (TVF) [22] for this paper. For the purpose of reducing the computational complexity, a graphics processing unit (GPU)-based implementation is introduced, which enables the TVF calculation in near real time. After that, we construct a novel local Riemannian metric based on saliency components derived from TVF, which aids the further analysis of the surface properties and the corresponding tensor field. Please note that our recent work [10][1] has extensively studied the feasibility to plan a path in 3-D using point cloud representation. Comparing with grid-based algorithms [23], it has shown better flexibility with additional information embedded in a tensor field. This paper is based on these existing findings and develops a theoretically closed-form representation for similar problems. The proposed work is designed for global path planning with real-time constraint [24].

As a typical application, we present how such a Riemannian metric can be used for trajectory planning, especially for calculating the distance between the two points in the point cloud. We take advantage of the stick saliency of TVF, which indicates the strength of local planarity. This information is used to help them generate a trajectory that can be viewed as a "rational path," i.e., the resulted path tries to drive along flat areas, to avoid climbing redundant slops and to avoid paths that lead to much vibration on the way-path. The mathematical criteria for the evaluation are introduced in Section V.

B. Contributions

In this paper, the following aspects are addressed.

1) A GPU-accelerated implementation of TVF for 3-D point cloud. We show significant improvement in performance comparing with optimized CPU implementation and other related methods.

[1]Colas *et al.* [10] were the winner of the best RoboCup Paper Award for IROS 2013.

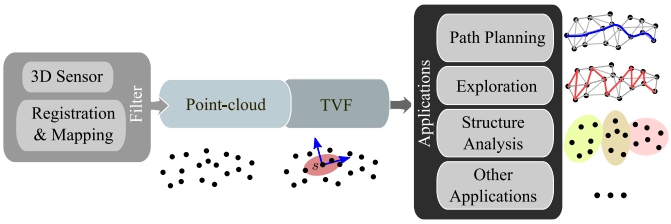


Fig. 3. Proposed pipeline to process point cloud in robotic mapping and navigation.

2) Surface normal estimation using sparse tensor voting (STV). We compare two different algorithms in terms of precision and complexity. We show that the STV-based algorithm is more precise and flexible.
3) A Riemannian metric defined on tensor voting vector field is introduced, which indicates the local similarity of the surface orientation and curvature.
4) A generic modeling approach based on TVF for nonplanar surfaces is introduced.
5) Trajectory is planned on raw point cloud without the preprocessing such as mesh reconstruction [18], [25].
6) A path planning pipeline for mobile robots on point cloud, validated by both simulation and tests on real data.

C. Proposed Pipeline to Process Point Cloud in Robotics

We propose to manage robotic navigation problem using raw point cloud as input, as shown in Fig. 3. The point cloud representation is inherently sparse, which means not any arbitrary point in the configuration space has definition. This sparse representation is hard to deal with, since the missing information among points lead to miss representation for path-planning algorithm to work on. Therefore, a dense representation, such as a tensor field or a mesh, is necessary, before designing navigation algorithms. Additionally, when a dense representation is available, applications such as structure analysis [26] and segmentation [20] could be handily treated.

D. Organization

In Section II, the related works are reviewed. The notations and general summary of TVF are introduced in Section III. The GPU implementation (released as open-ware) of tensor voting is introduced and evaluated in Section IV. After that, we introduce the novel Riemannian local metric in Section V, where the distance function is extensively studied. In Section VI, the pipeline to achieve optimized trajectory planning using the proposed framework is presented. The simulation results and tests on real data are introduced in Sections VII and VIII, respectively. The conclusion is given in Section IX.

II. Related Work

A. Robotic Navigation in 3-D

Navigation problem is mostly about path generation under various constraints, such as observability [27], resources [28],

coverage optimization [29], [30], and space constraints [31]. For 3-D navigation of a robot, researchers working on unmanned aerial vehicle have proposed several practical approaches [32], [33] to generate path in a full 3-D space, regarding different kinematic constraints. In these works, a primary concern is how to maintain a reliable pose estimation of the robot. Nowadays, computer vision usually worked as the primary method to get the pose estimation [34], [35]. Belta and Kumar [36] proposed a theoretical perspective into path generation problems, where the motion generation was considered as a projection to $\mathcal{SE}(3)$ by minimizing the acceleration. The results were validated via simulation or in test environments where ground-truth poses were available.

For the robot considered in this paper, point cloud generated from a rotating laser rangefinder on a mobile robot is used as the only extroceptive sensory information. It results in a more challenging problem due to two primary reasons: first, point cloud is sparse information, which means that not all locations in the space have definition; second, it has constrained view angle from the ground level. Usually, only the objects near the robot can be sensed.

Another typical path generation problem in 3-D is trajectory generation for robotic arms. A state-of-the-art library named "MoveIt!" was proposed in [37]. In this scenario, the pose estimation of the arm is usually not of great concern, since it could usually be recovered by geometric constraints from the 3-D rigid body motion. It generally aims at generating a path to avoid the collision from the robot body to the obstacles from the environment [38] (usually denoted by point cloud as well). In this paper, we aims at an optimal path which enables the robot to move along the latent surface governed by the point cloud, which is theoretically more complex system.

In our previous work [10], we proposed a heuristic method to generate a path on point cloud using the TVF, where we did not consider the vibration and the smoothness for the robot to realize the path. In this paper, we mainly deal with this remaining problem. Moreover, the previous work required dense voting which is computational expensive. We propose in this paper a Riemannian metric to summarize local information by using only the results from sparse voting, leading to a much light-weight algorithm.

B. Path Planning for Complex Environment

Path planning on 2-D developable plane has been a solved problem, since several open-source libraries were available to the community [39]. Several researchers have tried to adopt similar algorithms also apply in outdoor scenarios like autonomous cars, where the problem mostly lies in perception and the latent driving rules [40]. With identified free space, the planning can be performed by a generic graph path planners [41]–[43]. Such that the planning problem is considered as an optimization problem that minimized the cost function between search nodes. This idea is also adopted by the proposed approaches, where the cost is defined to facilitate the robotic motion as well as minimizing the path length.

On the other hand, 3-D path planning for aerial [44] or underwater [45] robots have also been studied, leading to stochastic approaches that cope with the increased complexity in space like rapidly random trees [46] and neural network-based planning [23]. Those approaches allow for the exploration of a more complex search space which is not easily feasible with deterministic methods.

C. Surface Normal and Segmentation

Surface normal is a local consistent feature. Therefore, it is widely used for point cloud analysis [47], [48]. Regarding segmentation, the work by Pulli and Pietikäinen [49] aimed at segmenting range images into homogeneous regions, by decomposing x- and y-component of the normal vectors. It assumes perfect dense point clouds and the resulting algorithm only deal with segmentation in 2.5-D. Normal estimation can also be based on local constrained least square modeling [50]. These normal estimation results often lead to clustering or segmentation of point cloud, such as by an initial segmentation in normal space, then refines in distance space [51]. Teutsch *et al.* [52] presented a clustering algorithm for subset segmentation, which targets at segmentation of point clouds without plane-assumption. Reference [53] introduced an incremental way to model different clusters by using both angular and distance constraints. Further region-based segmentation algorithms were introduced in [20], [54], and [55]. Reference [56] is a recent report report on different criteria for surface normal estimation of 3-D range data. In this paper, we show the advantage of STV for surface normal estimation using the proposed GPU implementation.

D. Tensor Voting Framework

Tensor voting [22] is originated in computer vision. It has been applied to applications such as segmentation [57], [58]. Through these works, tensor voting has shown its importance in reconstructing missing structures and local information registration [59], [60]. We consider it as one of the most important algorithms for structural analysis, because it outperforms other methods by its tolerance to noise and missing data, its consistency for local information and intuitive extraction of evidence saliency, etc. Preliminary work using TVF for planning has been proposed using an iterative algorithm over the dense voting grids [61], where the generated path adapts to smooth local curvature due to the nonholonomic motion constraints. In this paper, we use the results from sparse voting to facilitate the robot manoeuvre. The local information is represented by a Riemannian metric.

Nevertheless, the computational cost of tensor voting is high. The original algorithm has complexity $O(N^2)$, where N is the number of points in the point cloud. In this paper, we proposed a parallel computation, which was further optimized considering the advanced calculation characteristics of compute unified device architecture, in order to improve calculation efficiency, e.g., using coalesced memory access and avoiding atomic operation.

E. Geodesic on Point Cloud

Provided with the surface model, the geodesic calculation is relatively easy, such as using iterative midpoint

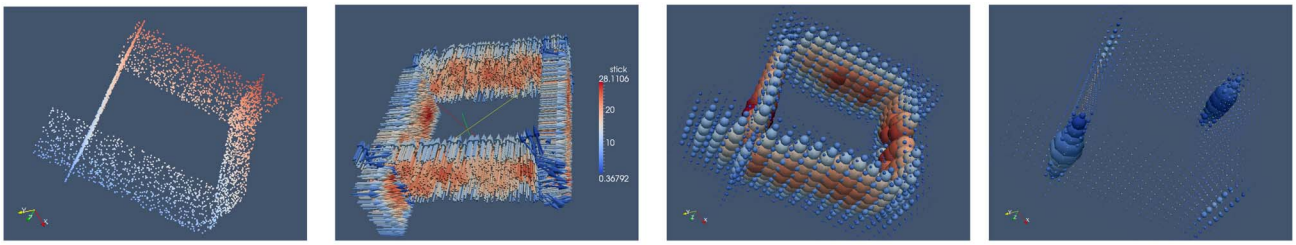


Fig. 4. Typical results from TVF. (a) Raw point cloud. (b) Result of STV. The color map indicates the stick saliency of each point. (c) and (d) Results of DTV in terms of stick and plate components, where the radius of the spheres indicate the stick and plate saliency, respectively.

search or approximating by descent gradient [62]. An efficient method based on contour propagation was proposed in [63]. Meanwhile, several works have put effort to calculate geodesic on a point cloud. Ruggeri *et al.* [64] proposed an approximation method for energy minimization, regarding errors for local surface fitting. In our case, the local surface information is embedded in TVF, which is computationally more efficient since the voting procedure is linear in time on GPU. Mémoli and Sapiro [65] explained the intrinsic properties for geodesic on point clouds, considering manifold sampling and noise.

However, all these existing works deal with the point cloud only in Euclidean space, i.e., $\mathbb{E}^3$, which equips a Riemannian metric as a rank-three identity matrix. It introduces inherent drawbacks for robotic navigation, since it assumes the whole configuration space is linked by straight geometry. Nevertheless, as long as the robot is moving on the "ground," its motion is strictly constrained. In order to ameliorate that, we define a Riemannian metric in the tensor space $\mathbb{T}^3$ introduced by TVF. The smoothness of such a space indicates the similarity of local smooth structures instead of direct distances. By analyzing the metric of TVF ($\mathbb{T}^3$) in terms of the direction of eigen vectors, we obtain the weights for edges of a graph constructed by sample points.

III. Tensor Voting Framework

In this section, we briefly summarize basic concepts of tensor voting over 3-D point cloud and the notations.

A. Overview

Tensor voting [22] is a computational framework used for structural extraction based on saliency of basic geometrical elements. It originated in computer vision problems. King [60] extended its application regarding point cloud-based terrain modeling and proposed an optimized stick voting field. Following a generic pipeline described in [66], we construct sparse ball voting fields eyes(3), and broadcast it through each neighboring point by a decay function, for each point:

$$k(d,\sigma) = e^{-\frac{d^2}{\sigma^2}} \tag{1}$$

where d is the Euclidean distance between the voter and votee, and σ is a selected kernel size. The decay function defines the decayed strength in broadcasting local information to the neighborhood. For robotic navigation applications, the kernel size can be chosen as the size of the navigation footprint.

The collected votes from each point lead to a 3×3 tensor containing the neighboring structural information. The eigen decomposition of the 3×3 tensor T can be formulated as

$$\begin{aligned} T &= \alpha_1 \hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \alpha_2 \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T + \alpha_3 \hat{\boldsymbol{e}}_3 \hat{\boldsymbol{e}}_3^T \\ &= (\alpha_1 - \alpha_2)\hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T && \text{(stick component)} \\ &\quad + (\alpha_2 - \alpha_3)\left(\hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T\right) && \text{(plate component)} \\ &\quad + \alpha_3\left(\hat{\boldsymbol{e}}_1 \hat{\boldsymbol{e}}_1^T + \hat{\boldsymbol{e}}_2 \hat{\boldsymbol{e}}_2^T + \hat{\boldsymbol{e}}_3 \hat{\boldsymbol{e}}_3^T\right) && \text{(ball component)} \end{aligned} \tag{2}$$

where α_i's are eigenvalues sorted in decreasing length sequence and $\hat{\boldsymbol{e}}_i$'s are the corresponding eigenvectors. The stick saliency can be represented by $\lambda_1 = \alpha_1 - \alpha_2$. The stick saliency for each point indicates how confident that a point can be considered as lying on a local plane. The corresponding tensor, indicating the plane, is characterized by the normal direction of the local plane $\hat{\boldsymbol{e}}_1$. Similarly, $\lambda_2 = \alpha_2 - \alpha_3$ denotes the saliency of a point on an edge or curve; $\lambda_3 = \alpha_3$ denotes the saliency of a point as a free point, mostly an outlier. For the interested readers of the detail calculation of TVF, please refer to [60].

This process is called STV, because the voting procedure is only performed on sparse location of the points. Generally, a dense tensor voting (DTV) process is also implemented based on the results of STV. DTV will cast for each point the tensor from STV to its neighborhood, which leads to more reliable structure representation. Typical results of TVF on a toy dataset is depicted in Fig. 4, where Fig. 4(a) shows the raw point cloud which includes four plane structures. Fig. 4(b) shows the computed stick components, where the stick saliency, highlighted by a color map, indicates the belief that a point is lying in a local plane. Fig. 4(c) and (d) demonstrates the results of DTV. The scales of the sphere markers indicate the strength of stick saliency (for planes) and plate saliency (for edges).

B. Information Embedding

Usually, the so-called dense voting is applied after sparse voting, because sparse voting only provides information at the exact positions of points in the point cloud. Due to the general nonuniform density of the points, caused by view-point changes of the sensor, these sparse information may not be easily usable. However, even with online optimization [10],

TABLE I
PERFORMANCE GAIN BY OPTIMIZATION

Method	Exe. time (ms)	Speed-up	Accu. speed-up	Comment
CPU	728479	/	/	optimized code
naive GPU	11165.6	65.24x	65.24x	using Atomic operations
apply fast math	2287.4	4.88x	318.45x	e.g. __fsqrt_rn
using voter loop	1119.4	2.04x	650.72x	
using shared memory	1063.6	1.05x	684.85x	

the dense voting process is generally computationally very expensive, since it requires regeneration of the tensor field at each sampled position. In order to realize path generation in near real time, it is preferable to not include the dense voting procedure. In this paper, we propose a metric embedding method to recover the local information as shown in the next section. At the same time, it enables to generate an optimal path which is a compromise between minimum distance and smoothness. Therefore, the components from STV are used for the path-planning algorithm introduced in this paper. Each point in a point cloud V can be defined as a surfel v_i

$$V = \left\{ \boldsymbol{v_i} | \boldsymbol{v_i} = (x_i, y_i, z_i, \boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}, \alpha_1, \alpha_2, \alpha_3)^T \right\}. \quad (3)$$

The position of point is further denoted by $\boldsymbol{s_i} := (x_i, y_i, z_i)^T$. In the neighborhood of $\boldsymbol{s_i}$, a smooth manifold of the same dimension as the voting tensors can be embedded [67], because of the continuity of the tensor voting procedure. The local manifold is represented by M_i, which is equipped with a Riemannian tensor g_i. For the convenience of representation, we omit the subscript and denote the Riemannian manifold as (M, g) for the rest of this paper, where g is the corresponding Riemannian metric.

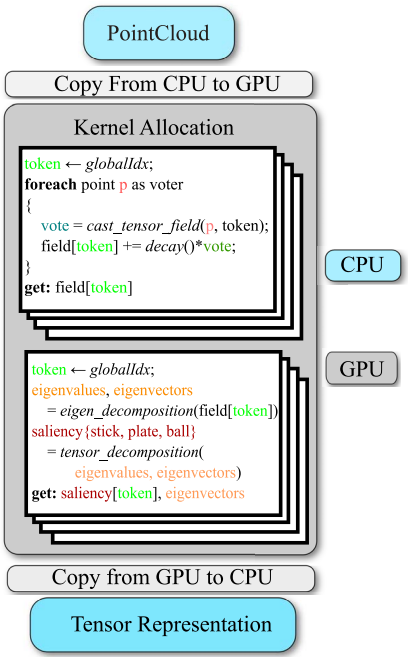


Fig. 5. Algorithm overview for GPU-based STV.

IV. TENSOR VOTING AND TENSOR SPLIT ON GPU

We summarize the parallel TVF in this section. At the same time, we address several technique details which are directly related to the performance.

A. Structure Overview

The sparse voting kernel is depicted as in Fig. 5. There are two main blocks executed on GPU lying in the middle part of Fig. 5, which are designed for tensor field propagation and tensor split, respectively. The tensor split algorithm is using orthogonality constraints proposed in [68].

B. Implementation

In order to improve the performance, we used several conducted techniques. The comparison by applying these techniques (mean of five runs) are shown in Table I, where we executed sparse tensor ball voting with the same kernel size for 14 K points. We start with the result of an optimized CPU code, followed by the naive GPU implementation [69]. Then by applying different techniques, we reach to the proposed algorithm. There has been other work reported on GPU-based tensor voting [69]. Comparing with this state-of-the-art [69], several optimizations have been conducted in this paper. Besides ordinary measures such as by using fast math library, one major optimization is that the iterations of voter's are taken as base-loop instead of votee-based iterations. This change is critical for the following two reasons.

1) The final tensor component is gathered information by each votee. When use voter-based iteration as base loop, the generation of output needs noncoalesced access of memory space. It is extremely inefficient for most GPU hardware comparing to coalesced access [70].
2) Because of the noncoalesced access, atomic operations are required, which is again a performance blocker for GPU computation. It is reported to be thousand times slower than direct cycle [71]. The proposed voter-based loop will alleviated this by direct shared memory access.

We could see that the proposed framework, which uses votee-based loop instead of voter-based loop, greatly reduces the calculation time as shown in the next section.

C. Effect of Kernel Size and GPU Performance

The size of the voting kernel σ greatly affects the computational complexity. A greater σ leads to quadratically more points to vote. By varying the size of σ, we show the execution time of STV and tensor split in Fig. 6. Since σ also defined

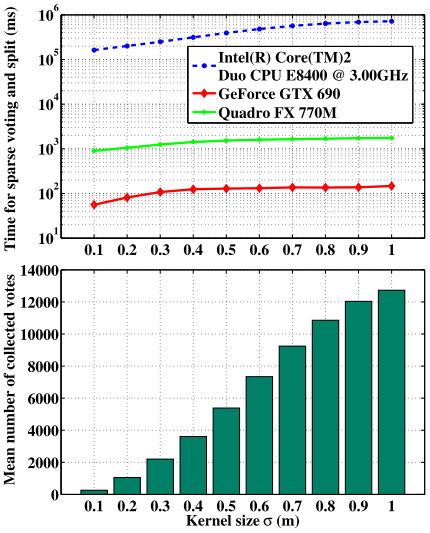


Fig. 6. Top: execution time for different implementation. Bottom: number of votes by changing kernel size σ.

the size of neighborhood for the robot, we want to make sure it can span at the minimum the rotating diameter of the robot. In our case, it is 0.7 m. The dataset is the stair point cloud shown in Fig. 2 which contains 14 K points. Please note that for standard tensor voting, points within the range of three times σ are considered. The number of votes is illustrated in Fig. 6 (bottom). It depicts the results on CPU and two different types of GPUs. We could see that the GPU implementation is superior to CPU in terms of calculation speed. Moreover, since the allocation of GPU computation is in unit of blocks, the increment of computational time is less than CPU for large number of points.

V. Riemannian Metric Equipped on TVF

In order to enable the navigation on point cloud, a metric defined on TVF is proposed. The definition of the metric is introduced and its properties are discussed in this section.

A. Projection and Transition Function

The embedded manifold M can be interpreted as shown in Fig. 7. For each point s in the point cloud, subjecting to the local latent surface, a neighborhood homeomorphic ψ^{-1} to an open subset in the TVF space can be defined. The corresponding manifold is equipped with a Riemannian metric g, which determines the inherent relations between a point and its projected neighborhood. Adopting the results from TVF, we build the metric g in such a way that local roughness of the latent surface can be reflected. This property is specifically interesting for the use case of trajectory planning. Usually, the optimal trajectory is not necessarily the shortest; instead, the smoothness in curvature is also important. For example, if the one with the shortest Euclidean distance passes through rough terrain or induces complex morphological adaptations, a relatively longer but flatter trajectory is preferable, considering the manipulation risks and power consumption for the robot.

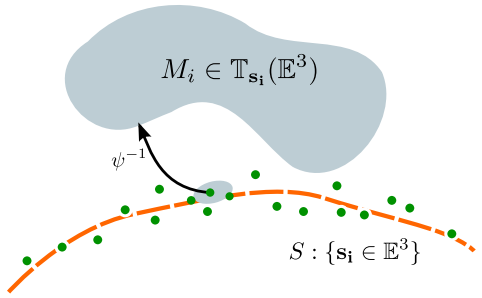


Fig. 7. Intuition of the embedded local manifold for points in $\mathbb{E}^3$. The dashed line indicates the latent surface, here we use a 10-nearest neighbour density-filter for point sampling.

B. Metric Definition

The Riemannian metric on M is a family of inner products on each tangent space T_pM, such that it depends smoothly on p, $\forall p \in M$. However, the specification of the T_pM is trivial if and only if it possesses a frame of global sections, e.g., a vector field is defined on M.[2] Therefore, for the case that a tensor field has already expanded the point cloud S, the Riemannian metric g can be arbitrarily defined, as long as it is canonical.

Regarding the geometric components introduced in TVF, the stick component indicates the flatness of a local latent surface, with eigenvector expanding $\hat{\boldsymbol{e}}_1{}^T\hat{\boldsymbol{e}}_1$; on the other hand, the orthogonal plane is expanded by $\hat{\boldsymbol{e}}_2{}^T\hat{\boldsymbol{e}}_2 + \hat{\boldsymbol{e}}_3{}^T\hat{\boldsymbol{e}}_3$. Inspired by the 2-D metric introduced in [67] and [73], with symmetric positive-definite tensors, we propose the following Riemannian metric:

$$g(s) \coloneqq \mathcal{I}_3 + \phi(\lambda_1(s))\left(\hat{\boldsymbol{e}}_1(s)\hat{\boldsymbol{e}}_1(s)^T\right) + \psi(\lambda_1(s))\left(\hat{\boldsymbol{e}}_1(s)\hat{\boldsymbol{e}}_1(s)^T + \hat{\boldsymbol{e}}_2(s)\hat{\boldsymbol{e}}_2(s)^T\right) \quad (7)$$

where $\mathcal{I}_3$ is an identity matrix, derived from the Euclidean space; $\lambda_1(s)$ is the normalized stick saliency, and $\hat{\boldsymbol{e}}_i(s)$ indicates the ith component of the local tensor for site s derived from TVF. Now, we would like to design a Riemannian metric on the TVF ambient space, so that by using the proposed metric, the following two properties are guaranteed.

1) A unit vector is along the most desirable curve when the vector is normal to $\hat{\boldsymbol{e}}_1(s)$.
2) A unit vector is along the most undesirable curve when the vector is parallel to $\hat{\boldsymbol{e}}_1(s)$.

For example, if the embedded vector is the control velocity to the robot, it means that we want to keep the robot moving on a flat surface as much as possible; at the same time, avoiding climbing redundant hills, etc.

Guided by these properties, we define the co-efficiencies $\phi(\cdot)$ and $\psi(\cdot)$ using the combination of exponential functions, where we need to ensure $(\phi(x)/\psi(x))$ is monotonically increasing and $(\phi(0)/\psi(0)) = 1$. We set

$$\phi(x) = \frac{e^{kx}}{e^{kx} + e^{-kx}}$$

and

$$\psi(x) = \frac{e^{-kx}}{e^{kx} + e^{-kx}} \quad (5)$$

[2]The routine proof for this proposition is omitted here. The readers are referred to [72] for details.

$$
\begin{aligned}
|\mathcal{C}|_{\mathcal{M}} &= \int_{\mathcal{C}} \sqrt{\tau_s^T g(s) \tau_s}\, ds \\
&= \int_{\mathcal{C}} \sqrt{\tau_s^T \left(\mathcal{I}_3 + \frac{e^{k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \hat{\boldsymbol{e}}_{\boldsymbol{1}}(s) \hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)^T + \frac{e^{-k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \left(\hat{\boldsymbol{e}}_{\boldsymbol{1}}(s) \hat{\boldsymbol{e}}_{\boldsymbol{1}}(s)^T + \hat{\boldsymbol{e}}_{\boldsymbol{2}}(s) \hat{\boldsymbol{e}}_{\boldsymbol{2}}(s)^T \right) \right) \tau_s}\, ds \\
&= \int_{\mathcal{C}} \sqrt{\tau_s^T \mathcal{I}_3 \tau_s + \frac{e^{k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{1}} \right|^2 + \frac{e^{-k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{1}} \right|^2 + \frac{e^{-k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{2}} \right|^2}\, ds \\
&= \int_{\mathcal{C}} \sqrt{\tau_s^T \mathcal{I}_3 \tau_s + \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{1}} \right|^2 + \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{2}} \right|^2 \left(\frac{e^{-k\lambda_1(s)}}{e^{k\lambda_1(s)} + e^{-k\lambda_1(s)}} \right)}\, ds \\
&= \int_{\mathcal{C}} \sqrt{\tau^T \tau_s + \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{1}} \right|^2 + \left| \tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{2}} \right|^2 \left(\frac{1}{e^{2k\lambda_1(s)} + 1} \right)}\, ds
\end{aligned} \tag{6}
$$

Please notice that the ball saliency is not considered in (7), because the free points are less of interest for structural information.

C. Distance Derivation

Using the proposed metric g, the Riemannian curve length $\mathcal{C}$ is obtained by the integral over inner products on M, as in (6), shown at the top of the page, where τ_s defines the direction derivatives on the manifold. Given the results in (6), the following properties can be drawn.

1) The length decreases when the direction of $\hat{\boldsymbol{e}}_{\boldsymbol{1}}$ diverges from τ_s, where $|\tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{1}}|^2$ indicates the cosine of the separation angle.
2) When the stick saliency λ_1 grows, the length decreases, which coincidentally means that the integral path lies on a flatter surface.
3) The local minimal direction is orthogonal to the secondary eigen vector $\hat{\boldsymbol{e}}_{\boldsymbol{2}}$, where $|\tau_s^T \hat{\boldsymbol{e}}_{\boldsymbol{2}}|^2$ is minimized. It implies that the local geodesic is along the $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$ direction (or its opposite) in $\mathbb{T}_s$. This information may help the exploration task for mobile robots, since the corresponding direction is the easiest one to approach. Further discussion is not included in this paper.
4) Based on the last item, when the robot is moving in the direction of $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, the Euclidean distance is used. Otherwise, the curve length is greater than the Euclidean distance.
5) It does not depend on parametrization of the curve, that is
$$|\mathcal{C}|_{\mathcal{M}} = \int_{\mathcal{C}} \sqrt{\tau_s^T g(s) \tau_s}\, ds = \int_{\mathcal{C}'} \sqrt{\tau_{s'}^T g(s') \tau_{s'}}\, ds'.$$
6) It does not depend on coordinates on Riemannian manifold M, that is
$$|\mathcal{C}|_{\mathcal{M}} = \int_{\mathcal{C}} \sqrt{\tau_s^T g(s) \tau_s}\, ds = \int_{\mathcal{C}} \sqrt{\tau_s'^T g(s) \tau_s'}\, ds.$$
7) The sum rule of integral also shows the additivity of $|\mathcal{C}|$. As basic properties of a Riemannian manifold, the proof of these three last properties is omitted. The readers are referred to [74] for further discussions.

Though the shortest distance is provided by the following local $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$'s, the geodesic to realize such distance does not necessarily exist due to physical constraints in $\mathbb{E}^3$ space, e.g., the robot cannot tele-transport. Therefore, we need to define a numeric solution to generate the feasible optimal path, which best realizes the proposed concept. To this end, we adopt the k-nearest-neighbor (k-NN) concept which is mostly applied in manifold learning methods such as Isomap [75]. The projected weights can then be evaluated in a projected space onto the plane supported by $\hat{\boldsymbol{e}}_{\boldsymbol{2}}$ and $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, which are direction vectors in $\mathbb{E}^3$. In the next section, we introduce the typical trajectory planning for a mobile robot on a point cloud-based representation.

VI. Recipe for Point Cloud-Based Trajectory Planning

A. Graph Construction and Distance Mapping

Planning in robotics is ubiquitous with various representations [76]. The proposed representation is based on the resulted tensor field from TVF. A typical application using the proposed model is the trajectory planning on point clouds. However, the closed form of planning using the given metric is mathematically intractable. Utilizing the fact that the local geodesic is along the direction of local $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$, we propose an accessible approximation as shown in Algorithm 1, which equivalently maintains the major properties of (4).

Considering that $g(s)$ is built in canonical form, following the discussion in Section IV-C, the optimal direction is along the direction of $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$. By using (7), the Euclidean distance is used in this case. Furthermore, weights among sites are defined in a way, such that the diversity between the directions of the path and $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$ is exponentially punished.

B. Pipeline

As a summary, the pipeline of point cloud based robotic planning is as shown in Fig. 3. We first process the raw point cloud by TVF. Then Riemannian embedding maps the points from $\mathbb{E}^3$ to $\mathbb{T}_s$ space, by which the local structural information is coded. The geodesic measurement, mapped in Algorithm 1, is used by target applications such as path planning.

Algorithm 1 Algorithm of Trajectory Planning on Point Cloud

1 Create kNN graph $\mathcal{G}(\mathcal{V}, \mathcal{D})$, where the nodes are positions of points $\boldsymbol{s_i} \in \mathbb{E}^3$

2 The edges $\mathcal{D}$ is calculated by embedding of Riemannian metric in Euclidean space, using the fact that a direction along $\hat{\boldsymbol{e}}_{\boldsymbol{3}}$ is preferred. For site $\boldsymbol{s_i}$ and its neighbor $\boldsymbol{s_i^j}$, distance d_i^j is defined as:

$$d_i^j = \left|\boldsymbol{t_i^j}\right| e^{1-\left|\boldsymbol{t_i^j}^T \hat{\boldsymbol{e}}_{\boldsymbol{3}i}\right|}, \quad \text{where} \quad \boldsymbol{t_i^j} = \frac{\boldsymbol{s_i^j} - \boldsymbol{s_i}}{\left|\boldsymbol{s_i^j} - \boldsymbol{s_i}\right|} \tag{7}$$

3 For each query that consists of a pair of starting and ending points, Dijkstra [41] search is implement on $\mathcal{G}(\mathcal{V}, \mathcal{D})$.

C. Metrics for Evaluation

To validate or compare various algorithms for trajectory planning, the following metrics can potentially be used.

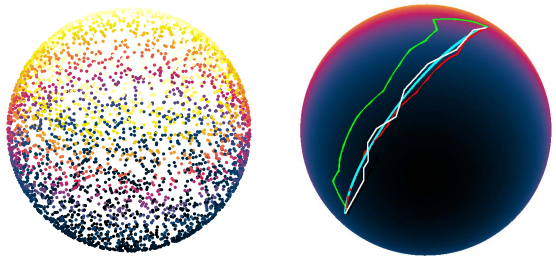
1) *Number of Site Visits:* It indicates the number of intermediate points along the path. It can also be interpreted as the shortest distance by considering all weights among sites are unity.
2) *Length of Trajectory:* The accumulated Euclidean distance of the calculated path.
3) *Mean Curvature (MC):* It is the average of the principal curvatures at a point, i.e., MC $= (\kappa_1 + \kappa_2/2)$. It reflects to which extent the local neighborhood can be considered as a minimal surface. Intuitively, it represents the local flatness. The smaller mean of MC along the path indicates that the path lies on a flatter terrain.
4) *Gaussian Curvature (GC):* It is the product of the two principal curvatures at a point, i.e., GC $= \kappa_1\kappa_2$. Formally, it depends only on the Riemannian metric of the surface. It represents the local shape changes. The smaller mean of GC suggests less local shake along the path.

For practical cases, MC and GC are important for mobile robots, especially for the planning of a rational path, as described in the introduction.

VII. Simulation

In this section, we evaluate three related methods on simulated dataset. Each simulated point cloud is uniformly sampled from a parametric surface. The three methods are all based on k-NN graph of point sites, but with different weight definitions among point sites. Specifically, the following methods are compared.

1) *k-NN Search:* The weights are set to unity. The optimal path is the one with the minimum number of visited points.
2) *Shortest Euclidean Geodesic:* The weights are set to Euclidean distance. The optimal path is the one with the smallest accumulated Euclidean distance.
3) *Proposed Method (Algorithm I):* The weights are derived from (7), representing the optimal curve length of (7) constrained in $\mathbb{E}^3$. The optimal path is a rational path, which is with the least curvature change, and tends to obey the motion on a plane at the same time.



(a) (b)

Fig. 8. Result of trajectory planning on a unit sphere using sampled point cloud (k-NN $= 10$). Please note that the right image depicts the calculated trajectories overlaid on the raw surface only for visualization. The surface information is not used in this paper. (a) Raw 4000 points uniformly sampled from a sphere. (b) Generated path for various methods. Green: nearest-neighbor search (2.65); red: shortest Euclidean geodesic (2.44); white: proposed method (2.54); cyan: great-circle distance (2.41). The numbers in parentheses indicate the total length of the path.

A. Sphere Dataset

A sphere dataset is used to show basic behaviors of different methods on typical minimal surfaces. The point cloud is sampled from a unit sphere as shown in Fig. 8(a). The resulting trajectories are shown in Fig. 8(b).

We can see that the trajectories mostly fulfill the great-circle distance, except the green one. It shows that the regarding methods reasonably approximate the geodesic distance on typical minimal surface. We can also notice that the trajectory using the proposed method leads to minor zig-zag behaviors. It is mainly because the sampling from the surface is not perfectly uniform, and local nonuniformity leads to poor performance of TVF [60]. It can be improved by normalized dense voting on uniform sites or use denser sample points.

B. Complex Surface

1) Representations of the Surface: In order to validate the proposed method in a more general case, we evaluate the related methods on the point cloud as shown in Fig. 9. It is sampled from a latent surface, which is arbitrarily defined as

$$\begin{aligned} z = {} & 2 \cdot \log(x+5) + 30 \cdot \mathcal{N}\left(\begin{bmatrix} 15 \\ 6 \end{bmatrix}, \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}\right) \\ & - 40 \cdot \mathcal{N}\left(\begin{bmatrix} 5 \\ 6 \end{bmatrix}, \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}\right) + 35 \cdot \mathcal{N}\left(\begin{bmatrix} 10 \\ 16 \end{bmatrix}, \begin{bmatrix} 3 & 2 \\ 2 & 3 \end{bmatrix}\right) \end{aligned} \tag{8}$$

where $\mathcal{N}(\mu, \mathcal{C})$ defines a Gaussian kernel located at μ with covariance matrix $\mathcal{C}$. The advantage of a parametric surface is that GC and MC can be easily derived. The details for calculation are omitted in this paper. The readers are referred to [72] for hints.

2) Rational Path Planning: Inspired by the definition of a metric tensor, we consider the trace of the Riemannian metric g as the local measure of feasibility for a rational path. Greater trace of g implies a longer incremental curve length,

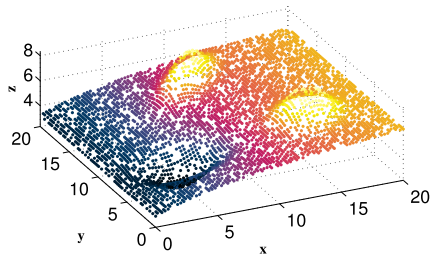


Fig. 9. 4000 points sampled from the surface defined by (8). Here, we use a ten-nearest neighbor density-filter for point sampling.

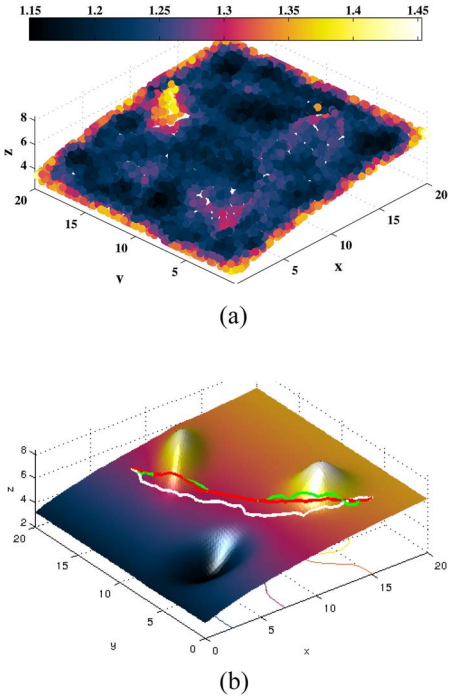


Fig. 10. Simulation with complex surface. (a) Trace of the proposed Riemannian metric. It indicates the roughness of the estimated neighborhood surface, highlighted by color. (b) Generated path for different methods. Green: nearest-neighbor search (21.42); red: shortest Euclidean geodesic (20.41); white: proposed method (21.90). The numbers in parentheses indicate the total length of the path.

which further induces more divergence of surface orientation and curvature. Fig. 10(a) shows the trace of local metric tensor.

The primary observation is that the proposed method (in white) leads to flat trajectory by actively avoiding the bumps in reasonable fashion, though with relatively longer path. A summary of the variations in z (elevation) direction along the path is shown in Fig. 11. It can be inferred that the proposed trajectory requires less adaptation to the terrain, which leads to easier motion or morphological planning for the mobile robot. Regarding the energy consumption, it can also benefit from the less variation in elevation as well.

3) Validation by the Direction of $\hat{\mathbf{e}}_3$*:* The property of the proposed method in Algorithm 1 is supposed to optimize the path by the following local $\hat{\boldsymbol{e}}_3$ direction as much as possible. It is because $\hat{\boldsymbol{e}}_3$ directs the direction of local geodesic, which leads to the most terrain similarity. A qualitative result is shown in Fig. 12. The $\hat{\boldsymbol{e}}_{3i}$ directions are represented by short lines, and the color definition of paths follows in Fig. 10(b).

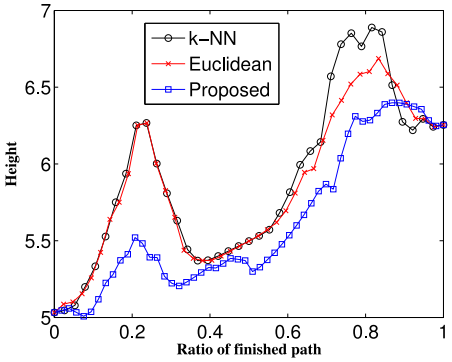


Fig. 11. Height variation along the path for different methods.

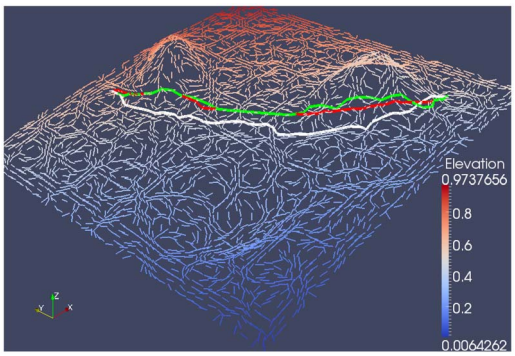


Fig. 12. Comparison to the direction of local $\hat{\boldsymbol{e}}_3$.

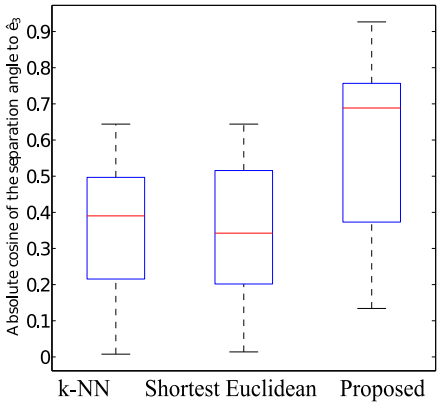


Fig. 13. Boxplot to show the divergence to $\hat{\boldsymbol{e}}_{3\,i}$ directions along the path.

The yellow trajectory calculated in Algorithm 1 mostly follows the short lines. A quantitative comparison is depicted in Fig. 13. It shows the statistics of the absolute cosine values of the path separation angle to local $\hat{\boldsymbol{e}}_3$ along the path. A higher value indicates the direction is more similar to $\hat{\boldsymbol{e}}_3$. It indicates that the proposed method fulfills the most coincidence aligning with the vector field of $\hat{\boldsymbol{e}}_3$.

4) Curvature Statistics: Recalling the metrics defined in Section V-C, the comparison of curvature dynamics is shown in Table II, where the outperforming value is highlighted by gray background. Despite of the longer traveled length, the proposed method holds the minimum MC and GC, as they are the main goal of the design.

TABLE II
STATISTICS OF THE TEST RUN. MC INDICATES MC; GC INDICATES GC

Method	k-NN	Shortest Euclidean	Proposed
# Nodes	39	43	47
Length	21.42	20.41	21.90
mean MC	0.0946	0.0853	0.0578
std-dev MC	0.1208	0.0989	0.0636
mean GC	-0.0249	-0.0254	-0.0089
std-dev GC	0.0246	0.0232	0.0116

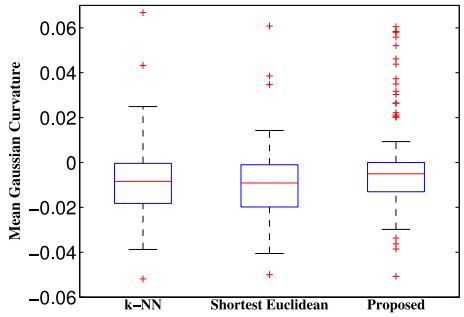


Fig. 14. Boxplot of mean GC of 200 times simulation.

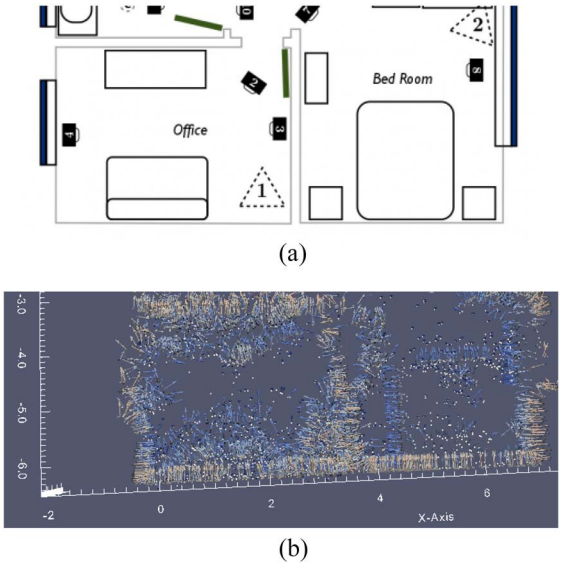


Fig. 15. Estimated normal for a typical apartment environment, 63.9 K points, taking 645 ms, $\sigma = 0.2$ m, considering mean 275 neighbors' votes. (a) Overview of the area. (b) Estimated surface normal.

5) Multiple Tests: Using 200 pairs of starting and ending points, multiple simulations are carried out. As part of the results, Fig. 14 shows the mean of mean GC of all the tests. The proposed method has smaller variance and absolute mean of GC. It indicates that the generated path can reduce robot shake during the maneuver.

VIII. TESTS ON REAL DATA

A. Qualitative Normal Estimation Results on Datasets

Several tests for normal estimation using the proposed algorithm are carried out. We show the qualitative results based on "apartment" and "mountain plain" from [77] in Figs. 15 and 16, respectively.

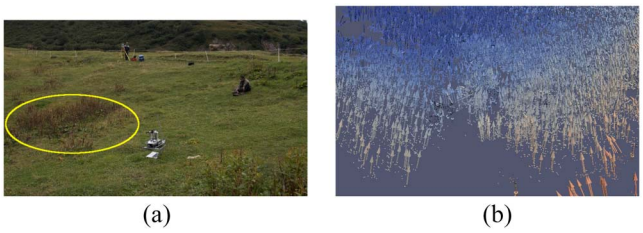


Fig. 16. Estimated normal for a typical outdoor field environment, 40.8 K points, taking 337 ms, $\sigma = 1.0$ m, considering mean 2113 neighbors' votes. (a) Overview of the area. (b) Estimated surface normal.

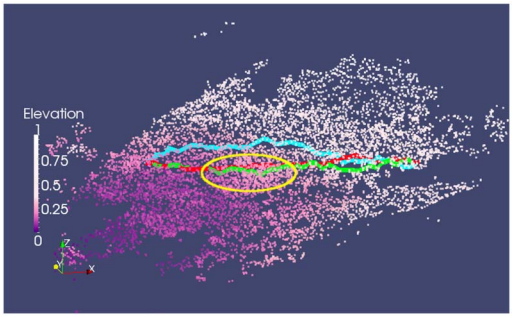


Fig. 17. Calculated trajectories for various methods. Green: nearest-neighbor search (24.7); red: shortest Euclidean geodesic (22.5); cyan: proposed method (29.3). The numbers in parentheses indicate the total length.

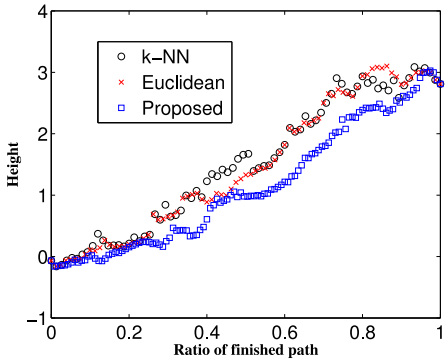


Fig. 18. Dynamics of height changes along the trajectory.

B. Test on the Planning Algorithm

As a sample of use case, we utilize one of the dataset introduced by [78] for the test on real data. The scenario is shown in Fig. 16(a). Please notice that the yellow eclipse marks a fire-weed area, which may cause trouble for the robot to traverse.

In order to reduce the cost of planning, we sample 10 K points out of the raw point cloud (around 504 K points). Adopting the GPU implementation of TVF introduced in [79], the computation time for sparse voting is 24.79 ms. Based on Algorithm 1, the calculated path is shown in Fig. 17. The corresponding height changes are depicted in Fig. 18. It can be observed that the cyan trajectory, by the proposed method, actively avoids the fire-weed area, leading to a more rational path for the mobile robot.

The estimation of curvatures on point cloud is a subtle problem. The readers are referred to [80] for curvature estimation by TVF. The evaluation in terms of GC and MC for real data is omitted in this paper due to missing ground truth.

IX. Conclusion

In this paper, a GPU implementation of TVF for 3-D point cloud was first introduced. It enables the structure analysis based on tensor voting to be feasible in real time. The whole library was available online, with robotic operation system (www.ros.org), at: https://sites.google.com/site/mingliurobot. The test dataset is available at: http://projects.asl.ethz.ch/datasets. Based on that, we introduced a Riemannian metric for the representation of point cloud, which helps the modeling of the environment, especially aiding the path planning for mobile robots directly on raw point cloud. Geometrical properties of the proposed metric tensor are discussed, which provide hints for further research related to the modeling problems using raw point clouds. The proposed framework was validated by path planning task on point clouds, using both simulated data and real dataset. The results showed that the proposed method is able to calculate rational paths for the robots, which facilitate the robotic navigation. Since the TVF is a generic representation of a 3-D environment, we plan to use this representation for further applications such as multirobot exploration and planning.

References

[1] H. Chen and D. Sun, "Moving groups of microparticles into array with a robot-tweezers manipulation system," *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1069–1080, Oct. 2012.

[2] A. Pamosoaji and K.-S. Hong, "A path-planning algorithm using vector potential functions in triangular regions," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 4, pp. 832–842, Jul. 2013.

[3] M. Liu, C. Pradalier, F. Pomerleau, and R. Siegwart, "The role of homing in visual topological navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vilamoura, Portugal, Oct. 2012, pp. 567–572.

[4] M. Liu, C. Pradalier, and R. Siegwart, "Visual homing from scale with an uncalibrated omnidirectional camera," *IEEE Trans. Robot.*, vol. 29, no. 6, pp. 1353–1365, Dec. 2013.

[5] G. Pereira *et al.*, "Robot navigation in multi-terrain outdoor environments," *Int. J. Robot. Res.*, vol. 28, no. 6, pp. 685–700, 2009.

[6] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, Sep. 2006.

[7] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[8] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A factored solution to the simultaneous localization and mapping problem," in *Proc. Nat. Conf. Artif. Intell.*, Menlo Park, CA, USA, 2002, pp. 593–598.

[9] K. Konolige *et al.*, "Efficient sparse pose adjustment for 2D mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Taipei, Taiwan, 2010, pp. 22–29.

[10] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart, "3D path planning and execution for search and rescue ground robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Tokyo, Japan, 2013, pp. 722–727.

[11] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. ICRA Workshop Best Practice 3D Perception Modeling for Mobile Manipulation*, vol. 2. Anchorage, AK, USA, 2010, pp. 1–8.

[12] R. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, 2011, pp. 1–4.

[13] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments," in *Proc. 12th Int. Symp. Exp. Robot. (ISER)*, vol. 20. Delhi, India, 2010, pp. 22–25.

[14] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6D SLAM3D mapping outdoor environments," *J. Field Robot.*, vol. 24, nos. 8–9, pp. 699–722, 2007.

[15] G.-J. Kruijff *et al.*, "Experience in system design for human-robot teaming in urban search & rescue," in *Field and Service Robotics*. Berlin, Germany: Springer, 2012.

[16] A. Macwan, G. Nejat, and B. Benhabib, "Target-motion prediction for robotic search and rescue in wilderness environments," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 5, pp. 1287–1298, Oct. 2011.

[17] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *ACM SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 163–169, 1987.

[18] J. Y. Hwang, J. S. Kim, S. S. Lim, and K. H. Park, "A fast path planning by path graph optimization," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 33, no. 1, pp. 121–129, Jan. 2003.

[19] J. Choi, M. Choi, S. Nam, and W. Chung, "Autonomous topological modeling of a home environment and topological localization using a sonar grid map," *Auton. Robots*, vol. 30, no. 4, pp. 351–368, 2011.

[20] M. Liu, F. Colas, L. Oth, and R. Siegwart, "Incremental topological segmentation for semi-structured environments using discretized GVG," *Auton. Robots*, vol. 38, no. 2, pp. 143–160, 2015.

[21] M. Liu and R. Siegwart, "Topological mapping and scene recognition with lightweight color descriptors for an omnidirectional camera," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 310–324, Apr. 2014.

[22] G. Medioni, M. Lee, and C. Tang, *A Computational Framework for Segmentation and Grouping*, vol. 1. Amsterdam, The Netherlands: Elsevier, 2000.

[23] A. Willms and S. X. Yang, "An efficient dynamic system for real-time robot-path planning," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 4, pp. 755–766, Aug. 2006.

[24] J. Yang, Z. Qu, J. Wang, and K. Conrad, "Comparison of optimal solutions to real-time path planning for a mobile vehicle," *IEEE Trans. Syst., Man, Cybern. A, Syst., Humans*, vol. 40, no. 4, pp. 721–731, Jul. 2010.

[25] S. Kim and J. Kim, "Occupancy mapping and surface reconstruction using local Gaussian processes with kinect sensors," *IEEE Trans. Cybern.*, vol. 43, no. 5, pp. 1335–1346, Oct. 2013.

[26] M. Liu and R. Siegwart, "Information theory based validation for point-cloud segmentation aided by tensor voting," in *Proc. Int. Conf. Inf. Autom. (ICIA)*, Yinchuan, China, 2013, pp. 1–6.

[27] X. Kang, H. Ren, J. Li, and W.-P. Yau, "Statistical atlas based registration and planning for ablating bone tumors in minimally invasive interventions," in *Proc. Int. Conf. Robot. Biomimet. (ROBIO)*, Guangzhou, China, 2012, pp. 606–611.

[28] L. Wang, M. Liu, and M. Q.-H. Meng, "An auction-based resource allocation strategy for joint-surveillance using networked multi-robot systems," in *Proc. IEEE Int. Conf. Inf. Autom. (ICIA)*, Yinchuan, China, 2013, pp. 424–429.

[29] H. Ren *et al.*, "Treatment planning and image guidance for radiofrequency ablation of large tumors," *IEEE J. Biomed. Health Informat.*, vol. 18, no. 3, pp. 920–928, May 2014.

[30] H. Ren, W. Guo, S. S. Ge, and W. Lim, "Coverage planning in computer-assisted ablation based on genetic algorithm," *Comput. Biol. Med.*, vol. 49, pp. 36–45, Jun. 2014.

[31] L. Wang, L. Liu, C. Hu, and M.-H. Meng, "A novel RF-based propagation model with tissue absorption for location of the GI tract," in *Proc. Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. (EMBC)*, Aug. 2010, Buenos Aires, Argentina, pp. 654–657.

[32] S. A. Bortoff, "Path planning for UAVs," in *Proc. Amer. Control Conf.*, vol. 1. Chicago, IL, USA, 2000, pp. 364–368.

[33] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proc. 41st IEEE Conf. Decis. Control*, vol. 2. Las Vegas, NV, USA, 2002, pp. 1301–1306.

[34] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. 6th IEEE/ACM Int. Symp. Mixed Augment. Real. (ISMAR)*, Nara, Japan, Nov. 2007, pp. 1–10.

[35] A. J. Davison, Y. G. Cid, and N. Kita, "Real-time 3D slam with wide-angle vision," in *Proc. IFAC/EURON Symp. Intell. Auton. Veh.*, Lisbon, Portugal, 2004, pp. 1–6.

[36] C. Belta and V. Kumar, "Euclidean metrics for motion generation on SE(3)," *Proc. Inst. Mech, Eng. C, J. Mech. Eng. Sci.*, vol. 216, no. 1, pp. 47–60, 2002.
[37] S. Chitta, I. Sucan, and S. Cousins, "Moveit!" *IEEE Robot. Autom. Mag.*, vol. 19, no. 1, pp. 18–19, Mar. 2012.
[38] Y. Nagata, Y. Murai, H. Tsuji, and S. Tokumasu, "3D robot navigation under unknown environments," in *Proc. Int. Conf. Intell. Technol.*, 2003, pp. 746–754.
[39] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2010, pp. 300–307.
[40] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray, "Autonomous driving in urban environments: Approaches, lessons and challenges," *Philosoph. Trans. Roy. Soc. A, Math. Phys. Eng. Sci.*, vol. 368, no. 1928, pp. 4649–4672, 2010.
[41] E. Dijkstra, "A note on two problems in connexion with graphs," *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.
[42] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.
[43] S. Koenig and M. Likhachev, "D* lite," in *Proc. Nat. Conf. Artif. Intell.*, Edmonton, AB, Canada, 2002, pp. 476–483.
[44] J. Tisdale, Z. Kim, and J. Hedrick, "Autonomous UAV path planning and estimation," *IEEE Robot. Autom. Mag.*, vol. 16, no. 2, pp. 35–42, Jun. 2009.
[45] C. Petres *et al.*, "Path planning for autonomous underwater vehicles," *IEEE Trans. Robot.*, vol. 23, no. 2, pp. 331–341, Apr. 2007.
[46] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, vol. 2. San Francisco, CA, USA, 2000, pp. 995–1001.
[47] K.-C. Chan, C.-K. Koh, and C. Lee, "A 3-D-point-cloud system for human-pose estimation," *IEEE Trans. Syst. Man, Cybern., Syst.*, vol. 44, no. 11, pp. 1486–1497, Nov. 2014.
[48] J. Dai and C.-K. Chung, "Touchscreen everywhere: On transferring a normal planar surface to a touch-sensitive display," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1383–1396, Aug. 2014.
[49] K. Pulli and M. Pietikäinen, "Range image segmentation based on decomposition of surface normals," in *Proc. Scandinav. Conf. Image Anal.*, vol. 2. Espoo, Finland, 1993, p. 893.
[50] E. Castillo and H. Zhao, "Point cloud segmentation via constrained nonlinear least squares surface normal estimates," Dept. Comput. Appl. Math., Univ. California, Los Angeles, CA, USA, Tech. Rep. CAM09-104, 2009.
[51] D. Holz, S. Holzer, R. Rusu, and S. Behnke, "Real-time plane segmentation using RGB-D cameras," in *Proc. 15th RoboCup Int. Symp.*, Berlin, Germany, 2011, pp. 306–317.
[52] C. Teutsch, E. Trostmann, and D. Berndt, "A parallel point cloud clustering algorithm for subset segmentation and outlier detection," *Proc. SPIE Opt. Metrol.*, vol. 8085, Jun. 2011, Art. ID 808509.
[53] K. Klasing, D. Wollherr, and M. Buss, "Realtime segmentation of range data using continuous nearest neighbors," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Kobe, Japan, 2009, pp. 2431–2436.
[54] M. Liu, F. Colas, F. Pomerleau, and R. Siegwart, "A Markov semi-supervised clustering approach and its application in topological map extraction," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vilamoura, Portugal, 2012, pp. 4743–4748.
[55] M. Liu, F. Colas, and R. Siegwart, "Regional topological segmentation based on mutual information graphs," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Shanghai, China, May 2011, pp. 3269–3274.
[56] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Kobe, Japan, 2009, pp. 3206–3211.
[57] Y. Dumortier, I. Herlin, and A. Ducrot, "4-D tensor voting motion segmentation for obstacle detection in autonomous guided vehicle," in *Proc. IEEE Intell. Veh. Symp.*, Eindhoven, The Netherlands, 2008, pp. 379–384.
[58] J. Jia and C. Tang, "Inference of segmented color and texture description by tensor voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 6, pp. 771–786, Jun. 2004.
[59] H. Schuster, "Segmentation of LIDAR data using the tensor voting framework," *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 35, no. B3, pp. 1073–1078, 2004.
[60] B. King, "Range data analysis by free-space modeling and tensor voting," Ph.D. dissertation, Rensselaer Polytech. Instit., Troy, NY, USA, 2009.
[61] E. Stumm, A. Breitenmoser, F. Pomerleau, C. Pradalier, and R. Siegwart, "Tensor-voting-based navigation for robotic inspection of 3D surfaces using LIDAR point clouds," *Int. J. Robot. Res.*, vol. 31, no. 12, pp. 1465–1488, 2012.
[62] J. Baek, A. Deopurkar, and K. Redfield, "Finding geodesics on surfaces," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep., Nov. 2007. [Online]. Available: http://cs.stanford.edu/people/jbaek/18.821.paper2.pdf
[63] R. Kimmel, A. Amir, and A. M. Bruckstein, "Finding shortest paths on surfaces using level sets propagation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 6, pp. 635–640, Jun. 1995.
[64] M. R. Ruggeri, T. Darom, D. Saupe, and N. Kiryati, "Approximating geodesics on point set surfaces," in *Proc. Symp. Point-Based Graph.*, 2006, pp. 85–93.
[65] F. Mémoli and G. Sapiro, "Distance functions and geodesics on submanifolds of R^d and point clouds," *SIAM J. Appl. Math.*, vol. 65, no. 4, pp. 1227–1260, 2005.
[66] P. Mordohai and G. Medioni, "Tensor voting: A perceptual organization approach to computer vision and machine learning," *Synth. Lect. Image Video Multimedia Process.*, vol. 2, no. 1, pp. 1–136, 2006.
[67] H. Koo and N. Cho, "Graph cuts using a Riemannian metric induced by tensor voting," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Kyoto, Japan, 2009, pp. 514–520.
[68] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. Matrix Anal. Appl.*, vol. 20, no. 2, pp. 303–353, 1998.
[69] C. Min and G. Medioni, "Tensor voting accelerated by graphics processing units (GPU)," in *Proc. 18th IEEE Int. Conf. Pattern Recognit. (ICPR)*, vol. 3. Hong Kong, 2006, pp. 1103–1106.
[70] NVIDIA, Advanced CUDA Webinar. (Jan. 5, 2015). *Memory Optimizations*. [Online]. Available: http://developer.download.nvidia.com/CUDA/training/NVIDIA_GPU_Computing_Webinars_CUDA_Memory_Optimization.pdf
[71] S. Robertson. (Jan. 5, 2015). *CUDA Atomics: A Practical Analysis*. [Online]. Available: http://strobe.cc/cuda_atomics/
[72] J. Gallier, *Geometric Methods and Applications: For Computer Science and Engineering*, vol. 38. New York, NY, USA: Springer, 2011.
[73] Y. Boykov and V. Kolmogorov, "Computing geodesics and minimal surfaces via graph cuts," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 1. Nice, France, 2003, pp. 26–33.
[74] H. M. Khudaverdian. (Jan. 5, 2015). *Riemannian Geometry*. [Online]. Available: http://www.maths.manchester.ac.uk/∼khudian/Teaching/Geometry/GeomRim11/riemgeom11.pdf
[75] M. Balasubramanian and E. Schwartz, "The Isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, p. 7, 2002.
[76] J. Su and W. Xie, "Motion planning and coordination for robot systems based on representation space," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 41, no. 1, pp. 248–259, Feb. 2011.
[77] *ASL Datasets Repository*, Auton. Syst. Lab., ETH Zürich, Zürich, Switzerland. [Online]. Available: http://http://projects.asl.ethz.ch/datasets/doku.php
[78] F. Pomerleau, M. Liu, F. Colas, and R. Siegwart, "Challenging data sets for point cloud registration algorithms," *Int. J. Robot. Res.*, vol. 31, no. 14, pp. 1705–1711, 2012.
[79] M. Liu, F. Pomerleau, F. Colas, and R. Siegwart, "Normal estimation for pointcloud using GPU based sparse tensor voting," in *Proc. IEEE Int. Conf. Robot. Biomimet. (ROBIO)*, Guangzhou, China, 2012, pp. 91–96.
[80] W.-S. Tong and C.-K. Tang, "Robust estimation of adaptive tensors of curvature by tensor voting," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 434–449, Mar. 2005.

Ming Liu, photograph and biography not available at the time of publication.