Sensors and Materials, Vol. 36, No. 12 (2024) 5521–5534
MYU Tokyo

S & M 3883

Distance-constrained Scale-space Point Cloud Surface Reconstruction Algorithm

Ming Huang, Wenjie Xiong,* Rui Wu, and Lei Wang

Engineering Research Center of Representative Building and Architectural Heritage Database, Ministry of Education, Key Laboratory for Urban Geomatics of National Administration of Surveying, Mapping and Geoinformation, Beijing University of Civil Engineering and Architecture, Beijing 100044, China

(Received September 2, 2024; accepted December 6, 2024)

Keywords: 3D reconstruction, point cloud, space of scale, distance constraint, urban underground tunnels

With the continuous improvement of living standards in modern society, the demand for refined 3D object representation models has grown considerably. Concurrently, the diversification of methods for acquiring 3D point clouds has significantly enhanced our ability to digitally express the surrounding environment. Consequently, the efficient and straightforward generation of high-precision 3D mesh models from point clouds has become particularly vital. As a typical production method for 3D mesh models, point cloud surface reconstruction technology has gained widespread application across various industries owing to its convenience and efficiency. Therefore, in this paper, we propose a distance-constrained scale-space surface reconstruction algorithm, which utilizes the neighborhood distances of point clouds to constrain the anomalous triangular mesh generated by the scale-space reconstruction algorithm, thereby enabling the more precise and efficient surface reconstruction of point clouds. Initially, the raw point set obtained from 3D laser scanning undergoes preprocessing, followed by point cloud reconstruction and mesh construction using the distance-constrained scale-space algorithm. Finally, the constructed mesh is back-projected onto the original point set, resulting in a more refined 3D model. Experiments conducted on the urban underground tunnel dataset successfully reconstructed the surface model of the tunnels, validating the effectiveness of the proposed method and enhancing the visual quality of the model.

1. Introduction

With the advancement of 3D laser scanning technology, particularly with its rapid maturation and widespread application, 3D laser scanning has emerged as a crucial tool for urban underground measurement.[1,2] Owing to its noncontact, high-precision, and high-efficiency characteristics, 3D laser scanning enables the swift acquisition of vast amounts of point cloud data, facilitating the construction of 3D models of target objects and significantly enhancing the accuracy and efficiency of data collection.

*Corresponding author: e-mail: xiongwj1123@163.com
https://doi.org/10.18494/SAM5360

ISSN 0914-4935
https://myukk.org/

Point cloud 3D reconstruction refers to the establishment of topological relationships between points based on the coordinate information of a 3D point set, resulting in the generation of a mesh model, which is then rendered.[3] Scholars both local and international have continuously innovated and improved reconstruction algorithms. Currently, 3D reconstruction algorithms can be broadly categorized into two types: explicit and implicit.

Explicit surface reconstruction involves the reconstruction of objects with arbitrary geometric features. However, this process entails extensive calculations involving circumcircles, angles, edge lengths, and so forth. When dealing with large point clouds, the algorithm's operational efficiency may be compromised, and it can also require substantial memory space. Yang *et al.* proposed a mesh reconstruction method based on adaptive grids, which necessitates the segmentation of point clouds into grid cells prior to reconstruction.[4] This method is well-suited for high-density point cloud reconstruction and demonstrates commendable robustness. The Delaunay triangulation technique is a method for constructing a triangulation from a given set of points, characterized by its ability to minimize the interior angles of the triangles, thereby avoiding the generation of excessively elongated triangles and optimizing the geometric quality of the mesh.[5]

Regarding implicit surface reconstruction, Crivellaro *et al*. proposed a multi-level interpolation radial basis function reconstruction algorithm that enhances the accuracy of the reconstructed surfaces.[6] However, this improvement comes at the cost of increased computational complexity and longer runtime. Hamza *et al.* introduced an implicit curve and surface iterative algorithm, providing both optimal and practical acceleration factors, which effectively reduces computational costs.[7] Xu *et al.* improved the Poisson reconstruction algorithm, enabling it to address the challenges of 3D reconstruction from point clouds with missing data.[8] Implicit surface reconstruction methods exhibit a degree of insensitivity to noise, allowing them to function without being significantly affected by noise present in the point cloud, resulting in smoothly reconstructed surfaces. The efficiency of these algorithms is contingent upon the selection of implicit functions, and the quantity of point clouds can also affect the processing time. Additionally, because the reconstructed surfaces tend to be smooth, capturing the geometric features of the objects can be challenging, which presents an avenue for further research in this area.

Scale-space point cloud surface reconstruction falls under the category of implicit reconstruction methods. Proposed by Digne *et al.* in 2011, this approach enhances the processing capabilities and reconstruction speed of point clouds through the introduction of scale space and a sphere-rotation-based algorithm.[9] In 2015, Digne further advanced this work by presenting a parallelized scale-space meshing algorithm aimed at reconstructing high-precision meshes from oriented input point sets.[10] However, both of these scale-space algorithms often lead to the generation of unnecessary anomalous triangular meshes during surface reconstruction. To address this issue, we present in this paper a distance-constrained scale-space surface reconstruction algorithm that preserves geometric features while constraining the formation of anomalous triangles. The effectiveness of this algorithm is validated using a point cloud modeling system on the latest data focused on urban underground spaces.

This paper is organized as follows. In Sect. 2, we utilize a point cloud modeling system to preprocess the raw point cloud data and elaborate on the principles of scale space and its

implementation. In Sect. 3, we focus on the application of our proposed algorithm for 3D reconstruction, followed by an analysis of the experimental results, which are compared with those obtained from the original and the parallelized scale-space algorithms. Finally, in Sect. 4, we present the conclusions drawn from this study.

2. Methods and Techniques

In Fig. 1, the constrained scale-space point cloud reconstruction method is demonstrated, which consists of three main steps:

(1) Point Cloud Preprocessing: This step includes the segmentation, denoising, and simplification of the point cloud.
(2) Implementation of Distance-Constrained Scale-Space Surface Reconstruction: This involves iterative smoothing and the constraint of anomalous triangle generation.
(3) Reprojection: The generated triangular mesh is projected back onto the original point set.

2.1 Preprocessing

Given the elongated and irregular contours of underground tunnels, as depicted in Fig. 2, along with the large volume and incompleteness of point cloud data, segmenting and recognizing shapes from the entire point cloud can be very time-consuming. Therefore, dividing the point cloud data of the tunnel into multiple blocks presents an effective approach.

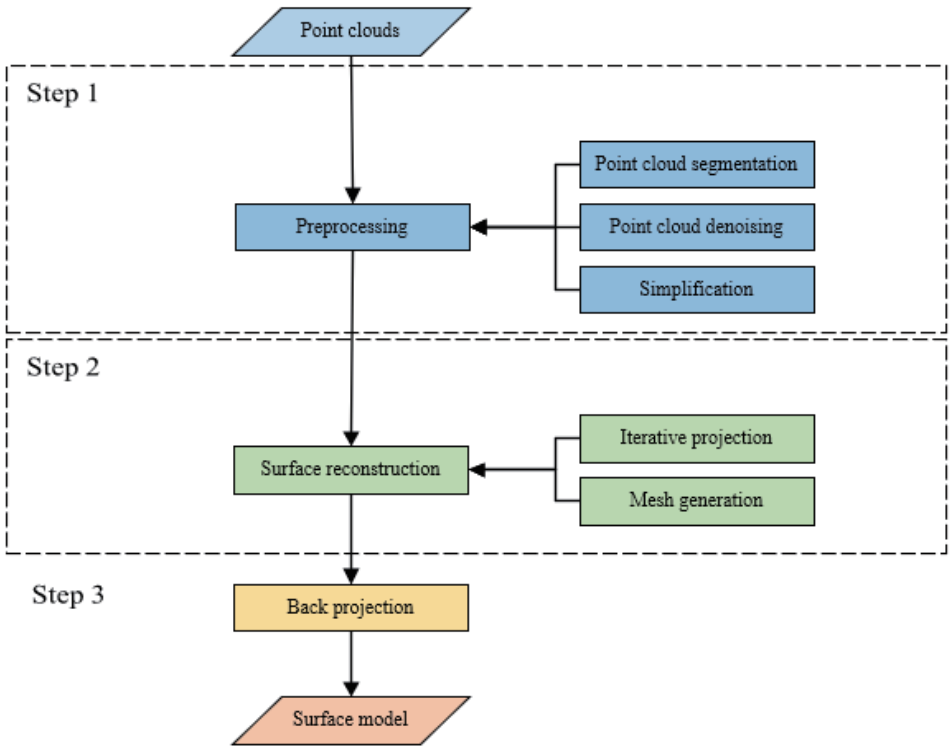


Fig. 1. (Color online) Workflow for distance-constrained scale-space point cloud reconstruction.

Fig. 2. (Color online) Top view of the raw point cloud data *P*.

Point cloud denoising is a critical step in the preprocessing stage, as 3D laser scanning results may include outliers caused by environmental factors and instrument limitations.[11] These outliers not only degrade the overall quality of the point cloud but also distort the estimation of local features, leading to inaccurate computational results. In this study, a manual denoising technique is initially employed to remove the most prominent noise from the raw point cloud data. This process involves selecting and removing erroneous points using bounding boxes and polygonal selections. Manual denoising is chosen here for its precision in removing significant outliers in complex datasets. However, further automated denoising steps will be applied subsequently to refine the data and handle more subtle noise patterns, ensuring optimal data quality for downstream tasks.

After completing the point cloud denoising, the substantial volume of point cloud data can make direct processing computationally intensive and slow. To address the challenges posed by large-scale point cloud data, in this study, we employed the resample method from the octree structure.[12] This method reduces the data volume by randomly selecting a subset of points from the point cloud. A sampling ratio of 50% is set on the basis of the desired point cloud density or level of downsampling. This approach not only decreases the overall data volume but also mitigates the effects of uneven point cloud distribution (or varying densities). The results of the preprocessing are illustrated in Fig. 3. Table 1 provides relevant information regarding the parameters for each module.

2.2 Distance-constrained scale space

2.2.1 Definition of scale space

Consider a smooth surface S within a 3D Euclidean space, which is assumed to exhibit C^2 smoothness. At any given point x on the surface, a normal vector $n(x)$ can be defined, which is perpendicular to the tangent plane. This normal vector can orient itself in either an inward or outward direction. In the case of continuous surfaces, the normal vector $n(x)$ consistently points towards the concave side of the shape. At each point x, one may select a normal plane that encompasses both the normal vector and a chosen tangent direction (that is, a plane containing a vector from the tangent plane). The intersection of this normal plane with the surface generates a

Fig. 3. (Color online) Top view of the four blocks partitioned from cloud *P*: (a) HD0816_1, (b) HD0816_2, (c) HD0816_3, and (d) HD0816_4.

Table 1
Parameters of segmentations.

Parameters	Size after preprocessing			Points
Figure	*X*/m	*Y*/m	*Z*/m	
HD0816_1	7.85	39.88	3.83	10885622
HD0816_2	10.14	39.87	4.01	10884145
HD0816_3	10.25	39.90	3.97	10885259
HD0816_4	9.71	39.88	4.55	10886548
Total		\		43541574

planar curve, and the curvature of this curve at point x represents the directional curvature of the surface in that specified tangent direction. The principal curvatures of the surface at point x, denoted as $k_0(x)$ and $k_1(x)$, are defined as the minimum and maximum of the directional curvatures, respectively. The mean curvature of the surface S at point x is given by

$$H(x)=\frac{1}{2}\left(k_1(x)+k_0(x)\right). \tag{1}$$

The scale space of point sets is introduced in Refs. 13 and 14, which show the application of mean curvature motion (MCM) on the point sets. The expression for mean curvature is given by

$$\frac{\partial x}{\partial t}=H(x)n(x). \tag{2}$$

This corresponds to all points moving towards the concave side of the shape, with a velocity equal to the speed of the surface's mean curvature.

2.2.2 MCM implementation

MCM can be approximated through an iterative process that projects each point in the dataset onto its local regression plane. By iteratively applying this projection process across multiple scale spaces, robust geometric information can be computed and traced back to the initial scale. For instance, the curvature of a point calculated at a certain scale can be associated with its position at the initial scale 0. The implementation steps of the MCM algorithm include the following.

The algorithm requires the input of a point set P, a query point p, and a radius r, ultimately outputting the distance d and a point p', which is the result of applying one iteration of MCM to p. The steps are as follows:

Step 1: Retrieve the set of neighbor points $N_r(p)$ within radius r from the point set P.

Step 2: If the number of neighbor points is less than 5, p is considered an outlier and is removed. If there are more than 5 points, calculate the distance d between each pair of neighboring points.

Step 3: Compute the centroid $\overline{p}$ and the average distance $\overline{d}$:

$$\overline{p} = \frac{\sum_{e \in N_{r(p)}} w(e)e}{\sum_{e \in N_{r(p)}} w(e)}, \tag{3}$$

where $w(e)$ is the weight function defined as a Gaussian weight, given by

$$w(e) = \exp\left(-\frac{\|p-e\|^2}{2r^2}\right). \tag{4}$$

Calculate the covariance matrix C:

$$C = \sum_{e \in N_r(p)} w(e)(e-\overline{p})(e-\overline{p})^T. \tag{5}$$

Step 4: Compute the eigen-decomposition of the covariance matrix C to find the eigenvector v_0 corresponding to the smallest eigenvalue.

Step 5: Update the position of the point p':

$$p' = p - (p - p', v_0)v_0, \tag{6}$$

where $(p - p', v_0)$ is the projection length of the point p onto the eigenvector v_0.

Step 6: Update the normal vector at the new position $p' \cdot n$:

$$p' \cdot n = \frac{p - p'}{\|p - p'\|} \cdot \text{sign}\left(\langle p - p', p' \cdot n \rangle\right). \tag{7}$$

where $\text{sign}\left(p - p', p' \cdot n\right)$ ensures the consistency of the normal vector direction.

These steps guarantee the stability and accuracy of the projection process. Even when the direction of the normal vector differs slightly from the continuous case, the smooth handling of the normal vector enhances the performance of the algorithm.

2.2.3 Scale-space implementation

The implementation is achieved by applying the mean curvature flow algorithm to the entire point set. In each iteration, a new point set is constructed, but it is unnecessary to retain the results of all iterations; only the original point set, the results of the last iteration, and the point set currently being constructed are stored. The point sets are organized within an octree, where each node contains three point sets: the original point set and two sets for storing intermediate results (Steps 1 and 2), which alternately hold the results of the previous and current scale-space iterations. The octree maintains an index of the current collection and is updated after each iteration. Spatial queries are performed by traversing the octree. The initial point set or the results of even-numbered iterations are stored in Step 1, while the results of odd-numbered iterations are stored in Step 2. Each point records its original source for back-projection purposes. To enhance efficiency, a depth-first traversal method is employed for the octree, allowing for the faster processing of leaf node points. The process of applying the scale-space iterative algorithm to the point set P is as follows:

The algorithm requires the input of a point set P, the number of iterations (N), and a radius r, ultimately outputting the modified point set P_N.

Step 1: Sort the point set P and store it in P_0, simultaneously constructing an octree structure to facilitate rapid neighborhood queries in subsequent steps.

Step 2: For each point P in the point set P_0, set its initial origin to itself $p.origin = p$.

Step 3: Initialize the index variable *idx* to 0, which will indicate the point set currently in use during the iteration process.

Step 4: Begin N iterations, with the primary objective of smoothing the point set in each iteration.

Step 5: Calculate a new index variable *nidx* to indicate the position of the point set where the results of the current iteration will be stored. The modulo operation is used to alternate the index between 0 and 1

Step 6: For each point p in the current point set P_{idx}, apply the MCM algorithm to compute the new point position p' and store it in P_{nidx}, while retaining the initial origin information of the new point.

Step 7: If the current iteration is not the first, delete the old point set P_{idx} to free up memory.

Step 8: Update the index variable *idx* to the new index variable *nidx* in preparation for the next iteration.

The final results must retain the initial point set, the outcomes of the scale-space iterations, and the correspondence between the point sets to facilitate the back-projection of the coarse-scale mesh to the fine scale. Throughout the entire process, the number of points remains consistent, although a small number of points (<0.1%) may be lost. The results of the scale-space iterations represent a denoised point set that approximates a smooth surface, achieved through interpolation methods constrained by the average distance $\overline{p}$ computed from the neighborhood, such as the ball rotation algorithm and the advancing front algorithm for mesh partitioning.[15]

2.3 Back-projection

After data input and mesh partitioning, the final step involves back-projection. Since each point in the smoothed point set records its original position, the back-projection process simply transfers the connectivity of the smoothed mesh back to the original points. This back-projection can generate an interpolated mesh of the original point set, although it cannot guarantee that the final mesh is free from self-intersections. However, with reasonable choices of radius and iteration count, this phenomenon typically does not adversely affect the visualization. The back-projection algorithm proceeds as follows:

The algorithm requires the input of the mesh M_N corresponding to the final smoothed point set P_N and outputs the mesh M_N on the initial point set P_0.

Step 1: Traverse each triangle t in the final smoothed point set P_N.

Step 2: Retrieve the three vertices of triangle t, denoted as v_0, v_1, and v_2.

Step 3: Using the initial positions of the vertices v_0, v_1, and v_2 (i.e., their origins $v_0.origin$, $v_1.origin$, and $v_2.origin$), create a new triangle t'.

Step 4: Add the new triangle t' to the initial mesh M_0.

By mapping each vertex of the smoothed mesh's triangles back to their original positions, the final interpolated mesh is generated.

3. Experiments and Analysis of Results

All experiments in this study were conducted on a computer equipped with a 3.60 GHz quad-core Intel Core i7-4790 CPU and 16 GB of DDR3 RAM. The point cloud 3D reconstruction system, along with the algorithms presented in this paper, was developed using Microsoft Visual Studio 2019. This system facilitates the import and export of point cloud data, as well as preprocessing operations such as point cloud segmentation, denoising, and simplification. Additionally, the OpenGL graphics library was employed to render the models, enabling their visualization. The system allows for interactive operations such as zooming, rotating, and scaling the models, which aids in dynamically observing the effects of the reconstruction. In the following section, we introduce the data required for the experiments.

3.1 Data description

The profile of the urban underground tunnel is an irregular polygon, with a length of 100 m, a maximum width of 10 m, and a height of 4 m. Pipes are stacked in layers along the walls, with

a maintenance route running through the center. The entire scanning task for the underground tunnel was conducted using nine scanning stations, resulting in a total of 1954204895 scanning points. The point cloud data used in this experiment consists of data from four scanning stations, totaling 87853148 scanning points, as shown in Fig. 4.

3.2 Experimental parameters and performance evaluation indexes

In this study, we primarily compare the original scale-space algorithm with the parallelized scale-space algorithm. The parameters are configured on the basis of the authors' recommendations, with the scale selection for all algorithms set to $N = 4$, which ensures consistency across different methods. The parameters of the proposed algorithm align with those of the parallelized scale-space algorithm, while the distance constraint and neighborhood radius r exhibit adaptability, indicating that these parameters are adjusted according to the number of point clouds sampled from different regions. In this study, the neighborhood radius is set as $N_r(p) = 24$, which is determined on the basis of the result of empirical analysis.

The performance of the algorithms is evaluated using two key metrics:

(1) Root Mean Square Error (*RMSE*): This metric measures the average distance between the sample points and the reconstructed surface, providing insight into the overall accuracy of the reconstruction.

(2) Hausdorff Distance (*HD*): This metric quantifies the maximum deviation between the reconstructed surface and the true reference surface from the dataset, highlighting the worst-case errors in the reconstruction process.

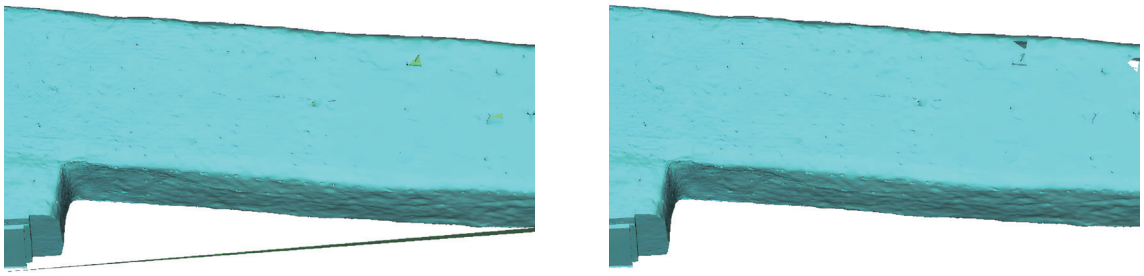
3.3 Results and comparisons

The algorithm presented in this paper was evaluated using a real urban underground tunnel dataset, comparing the accuracy of *RMSE* and *HD* as shown in Table 2. Each row of data under the two evaluation metrics was normalized by comparing it with the maximum value, with

Fig. 4. (Color online) Effect of each viewpoint of the underground tunnel data. (a) Top, (b) side, and (c) elevation views of the point cloud. (d) Internal.

Table 2
Quantitative comparison results under different dataset models.

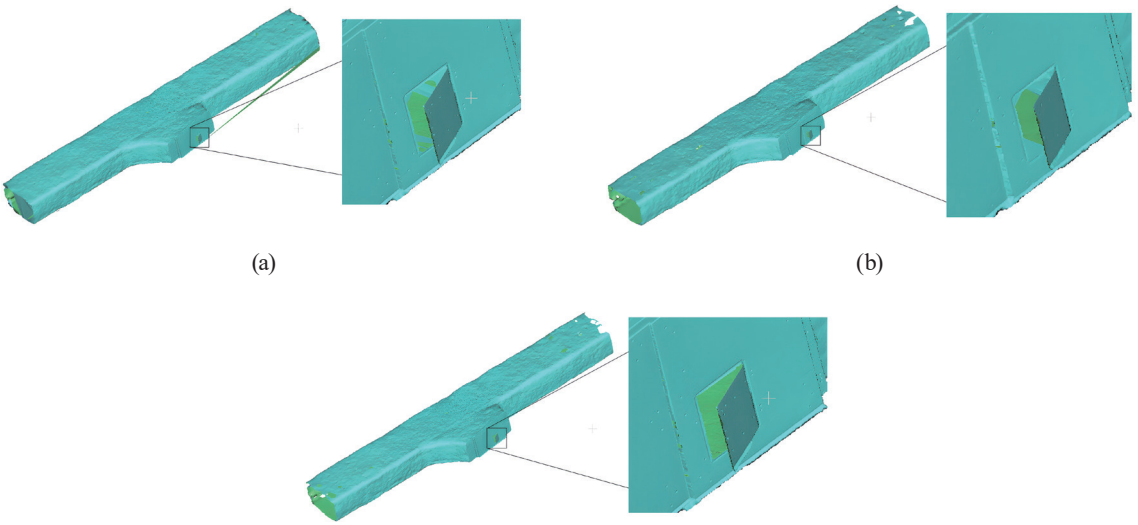
	RMSE			*HD*		
	Original scale	Parallelization scale	Constraints scale	Original scale	Parallelization scale	Constraints scale
HD0816_1	**0.920**	1.000	0.925	1.000	0.745	**0.701**
HD0816_2	1.000	**0.855**	0.861	0.936	1.000	**0.854**
HD0816_3	1.000	0.892	**0.859**	1.000	**0.782**	0.815
HD0816_4	1.000	0.905	**0.785**	1.000	0.945	**0.835**



(a) (b)

Fig. 5. (Color online) Comparison of the reconstruction results of our algorithm with those of the original algorithm on the HD0816_4 model. (a) Original scale-space algorithm and (b) our algorithm.

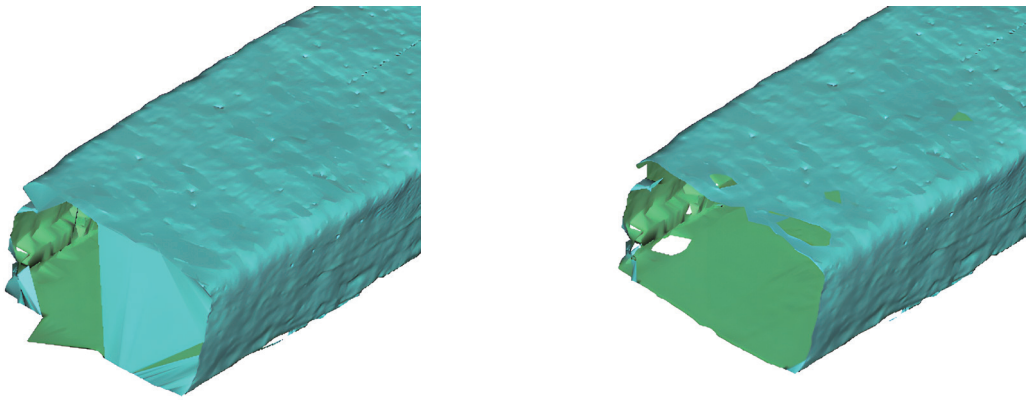
smaller values indicating greater accuracy. The best results are highlighted in bold in Table 2. From Table 2, it can be observed that the proposed algorithm demonstrates a degree of accuracy improvement across various segments of the urban underground tunnel dataset. Notably, the *HD* value achieved the optimal result in the HD0816_4 tunnel segment model, while the HD0816_1 and HD0816_2 models also exhibited similar levels of accuracy. Owing to the presence of noise, data loss, and mismatches, the *HD* values of the original scale-space algorithm are relatively high, as demonstrated by the local model of HD0816_4 shown in Fig. 5. However, the *RMSE* values of the proposed algorithm are not particularly outstanding across all models. This is primarily because the distance-constrained scale-space algorithm is designed to address the issues of abnormal triangular meshes generated by the original scale-space algorithm (as illustrated in Fig. 6), which allows point clouds in certain areas to better conform to the generated surface model. Conversely, the accuracy optimization for other normal regions is relatively minor. From Table 2, it is evident that the original scale-space algorithm performs the worst in the HD0816_4 model, primarily owing to significant data loss (see Fig. 7), leading to the generation of numerous abnormal triangular meshes. In contrast, the proposed algorithm implements targeted constraints, regulating the generation of triangular meshes based on the average radius of neighborhood points, thereby minimizing the impact of edge noise and data loss, resulting in lower *HD* values. Ultimately, the underground tunnel model results are depicted in Fig. 8. Owing to small areas of point cloud data loss, the reconstructed model inevitably exhibits holes, which can be filled as needed in subsequent processes.



(a) (b)

(c)

Fig. 6. (Color online) Comparison of the reconstruction results of our algorithm with those of the two original algorithms on the HD0816_4 model. (a) Original scale-space algorithm, (b) parallelized scale-space algorithm, and (c) our algorithm.



(a) (b)

Fig. 7. (Color online) Comparison of the reconstruction results of our algorithm with those of the original algorithm on the HD0816_4 model. (a) Original scale-space algorithm and (b) our algorithm.

(a)

Fig. 8. (Color online) (a) Final model HD0816_4 and (b) local details of the model HD0816_4.

(b)

Fig. 8. (Continued) (Color online) (a) Final model HD0816_4 and (b) local details of the model HD0816_4.

4. Conclusions

In the construction of smart cities, the development and utilization of urban underground spaces are gaining increasing attention. As a crucial component of these underground spaces, underground tunnels benefit significantly from the generation of more refined point cloud models, which enhance the efficiency and accuracy of smart city construction and management. In this paper, we proposed a distance-constrained scale-space surface reconstruction algorithm based on the scale-space surface reconstruction technique. The algorithm constrains the distances within point cloud neighborhoods across different regions, thereby reducing or eliminating the formation of anomalous triangular meshes within those areas. By making full use of the neighborhood information of the point cloud, experimental results in the urban underground tunnel dataset show that our proposed algorithm achieves more accurate reconstruction results in urban underground tunnels.

However, in this study, we focused solely on the surface modeling of underground tunnels and do not delve into the internal components (such as pipelines) within the tunnels. In our future work, we aim to conduct in-depth modeling research on the internal components of the tunnels.

Acknowledgments

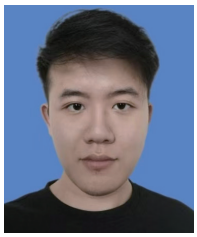
This work was supported by the National Natural Science Foundation of China (42171416), Huangshan University School-Level “Golden Course” Architectural Science (2021JK102), the BUCEA Doctor Graduate Scientific Research Ability Improvement Project (DG2024034), and the National Key Research and Development Program of China (Grant No. 2022YFF0904400).

References

1 F. Zhou, M. Li, C. Huang, H. Liang, Y. Liu, J. Zhang, B. Wang, and M. Hao: Front. Earth Sci. **10** (2022) 918285. https://doi.org/10.3389/feart.2022.918285
2 M. Yu, J. Wu, Y. Wang, W. Shi, and X. Wang: J. Disaster Prev. Mitig. Chn. **43** (2023) 588–595 + 636. https://doi.org/10.13409/j.cnki.jdpme.20210815002
3 M. Berger, A. Tagliasacchi, L. M. Seversky, P. Alliez, G. Guennebaud, J. A. Levine, A. Sharf, and C. T. Silva: Comput. Graph. Forum **36** (2017) 301. https://doi.org/10.1111/cgf.12802
4 Y. Yang, H. Fang, Y. Fang, and S. Shi: Measurement **151** (2020) 107220. https://doi.org/10.1016/j.measurement.2019.107220
5 S. B. Thayyil, S. K. Yadav, K. Polthier, and R. Muthuganapathy: Comput. Aided Geom. Des. **86** (2021) 101973. https://doi.org/10.1016/j.cagd.2021.101973
6 A. Crivellaro, S. Perotto, and S. Zonca: Appl. Numer. Math. **113** (2017) 93. https://doi.org/10.1016/j.apnum.2016.11.003
7 Y. F. Hamza, H. Lin, and Z. Li: Comput. Aided Geom. Des. **77** (2020) 101817. https://doi.org/10.1016/j.cagd.2020.101817
8 Z. Xu, C. Xu, J. Hu, and Z. Meng: Comput. Graphics. **97** (2021) 19. https://doi.org/10.1016/j.cag.2021.04.005
9 J. Digne, J.-M. Morel, C. Mehdi-Souzani, and C. Lartigue: Comput. Graph. Forum **30** (2011) 1630. https://doi.org/10.1111/j.1467-8659.2011.01848.x
10 J. Digne: Image Process. Online **5** (2015) 282. https://doi.org/10.5201/ipol.2015.102
11 M. A. Irfan and E. Magli: J. Visual Commun. Image Represent. **75** (2021) 103027. https://doi.org/10.1016/j.jvcir.2021.103027
12 J. Herráez, J. L. Denia, P. Navarro, S. Yudici, M. T. Martin, and J. Rodriguez: Measurement **93** (2016) 243. https://doi.org/10.1016/j.measurement.2016.06.039
13 J. Digne and J.-M. Morel: Numer. Math. **127** (2014) 255.http://dx.doi.org/10.1007/s00211-013-0584-y
14 J. Digne, J.-M. Morel, C.-M. Souzani, and C. Lartigue: Comput. Graph. Forum **30** (2011) 1630.http://dx.doi.org/10.1111/j.1467-8659.2011.01848.x
15 K. Yu, J. Chen, K. Fu, J. He, J. Zheng, and Y. Zheng: Comput.-Aided Des. **153** (2022) 103403. https://doi.org/10.1016/j.cad.2022.103403

About the Authors

Ming Huang is a professor and Ph.D. supervisor of Beijing University of Civil Engineering and Architecture. His research interests are in point cloud and image hyperfine 3D modeling and visualization, and mobile vehicle laser point cloud ground feature intelligent recognition.
(huangming@bucea.edu.cn)



Wenjie Xiong is a graduate student of Beijing University of Civil Engineering and Architecture. His research interests are in 3D laser scanning, 3D modeling, and visualization.
(xiongwj1123@163.com)

Rui Wu received his Master's degree from Hefei University of Technology, P.R. China. He now works in the School of Architecture and Civil Engineering, Huangshan University, and he studies in the School of Geomatics and Urban Spatial Informatics, Beijing University of Civil Engineering and Architecture. His research interests include architectural design and its theory, the conservation of historic buildings, green building, and city planning. (1108130122005@stu.bucea.edu.cn)

Lei Wang received his master’s degree from Guilin University of Technology in 2020, China. He is currently pursuing his Ph.D. degree in architecture at Beijing University of Civil Engineering and Architecture, China. His research interests include point cloud deep learning and architectural heritage protection. (leiw8448@gmail.com)