3D-PatchMatch: an Optimization Algorithm for Point Cloud Completion

Zhipeng Cai[1], Cheng Wang[2], Chenglu Wen[3], Jonathan Li[4]

Fujian Key Laboratory of Sensing and Computing for Smart City, School of Information Science and Engineering, Xiamen University
Xiamen, China

[1]czptc2h@gmail.com
[2]cwang@xmu.edu.cn
[3]clwen@xmu.edu.cn
[4]Junli@xmu.edu.cn

***Abstract*— Point cloud completion is an indispensible process for handling the occlusion problem occurring during the data acquisition. For completion, 3D point clouds generate a much larger searching space than 2D images while searching for the best match for boundary patches. To handle the searching speed bottleneck, this paper proposed a novel optimization algorithm which is called the 3D-PatchMatch algorithm for 3D point cloud completion. Inspired by the nature coherence of the point cloud data, the proposed algorithm use a random initialization process and a propagation process to reduce the convergence time. Furthermore, a random guess process is proposed to make sure the proposed algorithm has the ability of escaping from the local minima. Experimental results show significant time reduction rate for this method compared to brute-force search. Furthermore, the completion results with comparisons indicate that the proposed algorithm is rarely trapped in the local minima.**

***Keywords*— Optimization, LiDAR, Point Cloud, 3D, Completion**

I. INTRODUCTION

Flourishing LiDAR technologies have made the 3D point cloud a popular media data format. However, occlusion which occurs during data acquisition is a serious problem that affects the completeness of point clouds. Shadow-liked missing regions on the target are generated when the laser pulse is blocked by the frontal object. Completion is an important solution for occlusion problem. While completing the missing region using the complete region, searching for the best match is the core. Differ from 2D images, 3D space makes the searching speed the main bottleneck for point cloud completion. This paper proposed a 3D-PatchMatch algorithm for optimizing the searching process during point cloud completion.

Traditional context based point cloud completion methods tried to use hierarchical searching strategy [1] or course-to-fine process [2] to accelerate the searching speed. However, for large scale 3D point clouds, searching within one level can be a time-consuming task. Therefore, to handle this problem, a novel 3D-PatchMatch algorithm is proposed.

Connelly et al. [3] proposed a randomized correspondence algorithm for accelerating the searching speed for image editing. The key idea of this method is consisted of two parts: 1) some good matches is likely to be found using random sampling. 2) Good matches can be propagated quickly to the surrounding areas due to natural coherence. Inspired by the key idea of this method, we extend the algorithm to 3D space and made special modifications to improve the performance of the algorithm for point cloud completion.

The main contribution of this work is two-fold, as follows:

- A novel 3D-PatchMatch algorithm is proposed to accelerate the searching speed in 3D domain for point cloud completion.
- A general 3D point cloud completion framework is proposed to enhance the quality of point clouds.

The rest of this paper is organized as follows: Section Ⅱ introduced the related work of point cloud completion. Next, Section Ⅲ describes the detail of the 3D-PatchMatch Algorithm and proposed a framework for point cloud completion. Then, Section Ⅳ shows the performance of this algorithm for point cloud completion. Finally, Section Ⅴ makes the conclusion of this work.

II. RELATED WORK

The completeness of the point cloud would benefit many applications such as reconstruction and object recognition. 3D point cloud completion has been a popular research topic for over a decade. Most of them search for the best match in the complete region or in the database to fill in the missing region. Sharf et al. [2] proposed a context based method for completing small holes. Pauly et al. [4] first built a database of complete objects and then used a classification based method to complete the data. Park et al. [5] proposed a PDE modelling based method to complete closed surfaces. Xiao et al. [6] introduced a PDE and energy function based method which completes both the texture and geometry of the point cloud. However, the above methods are not focusing on real-world point cloud data. In other words, the size of the data is usually small. A few approaches aim at completing real-world point cloud were also published recently. Friedman and stamos [7] detected the scan line regularity and proposed an online completion method based on this regularity. Doria and Radke [8] transformed the point cloud into a depth image and then used a image in-painting method to complete the data. Zheng

978-1-4799-7749-9 /15/$31.00 ©2015 IEEE

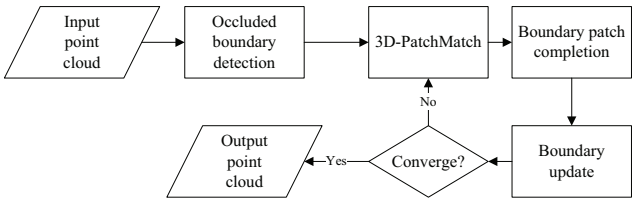

Fig. 1 The workflow of point cloud completion.

et al. [9] detected and clustered repeated parts to complete the point cloud. Chauve et al. [10] proposed an urban scene completion method based on a planar primitive. Nevertheless, lacking an appropriate optimization algorithm makes these methods suffer from the bottleneck of the searching speed.

III. METHODOLOGY

Define $P \subseteq R^3$ to be the input point cloud with occlusion and $Q \subseteq R^3$ to be the candidate point cloud. Let $B \subseteq P$ denote the occluded boundary point set of the input point cloud which represents the occluded region. For point cloud completion, the goal is to first detect the occluded boundary, B and then generate a completed point cloud, $P_{completed}$ based on the detected boundary. The workflow of our point cloud completion method is shown in Fig. 1. The proposed algorithm is used to quickly search for every boundary patch the best matched patch.

The workflow of the 3D-PatchMatch algorithm is shown in Fig. 2. Similar to the PatchMatch [3] algorithm in 2D space, It starts with an initialization process which randomly guesses the matches. Next, it iteratively propagates good matches to nearby area until convergence.

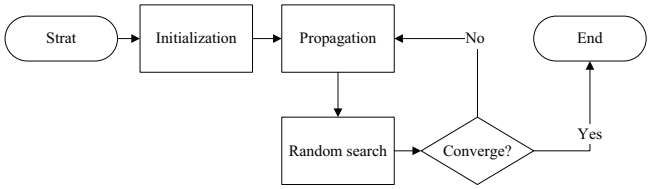

Fig. 2 The workflow of 3D-PatchMatch algorithm.

A. 3D-PatchMatch algorithm

In 3D point cloud, the natural coherence also exists as in 2D images. Specifically, a good guess is likely to be found randomly and then propagated to nearby patches. Therefore, we extend the PatchMatch algorithm to 3D space to accelerate the searching speed for point cloud completion.

For every boundary patch, let the nearest-neighbour field (NNF) $f : A \mapsto R^3$ denote map to the matched patch, for a certain distance function D of two patches. In a 3D point cloud P, given a patch coordinate a and its corresponding nearest neighbour b in the candidate point cloud Q, $f(a) = b - a$. For real-world point cloud, the patches in a nearby area tend to have the similar NNF. Therefore, our 3D-PatchMatch algorithm has a similar workflow as the PatchMatch algorithm. Specifically, in 2D space, the PatchMatch algorithm works as follows: 1) Randomly choose a match for every patch. 2) Propagate good guesses following the diagonal direction. 3) Randomly guess for every patch and accept the guess if it matches better. 4) Iterate step 2) and 3) until there is no update of the NNF for all patches between two iterations.

Specifically, for 3D point cloud, the random guess process is defined as follows. Let b^j denote the coordinate of a boundary patch center. We assume a better guess is more likely to be found near the current best match. Therefore, a sequence of guesses at an exponentially decreasing distance from the current best candidate patch center v_0. Because the point cloud is discrete, we select a point nearest to the guessed coordinate. For the ith candidate patch $p_i^{b^j}$:

$$p_i^{b^j} = \arg\min_{p^i \in P}(\| p^i - (v_0 + w\alpha^i R_i) \|). \quad (1)$$

where P is the searching space, R_i is a uniform random in $[-1,1]\times[-1,1]\times[-1,1]$, w is a large maximum searching radius, and α is a fixed ratio.

However, due to the much larger and discrete searching space in 3D point cloud, there are two problems that needs to be handled:

1) For completion, boundary patches take up only a small proportion of the whole point cloud. Therefore, for initialization, if the number of boundary patches is not large enough, a good guess is less likely to be found. This phenomenon will limit the converging speed of the algorithm.
2) The shape of the occluded boundary is usually irregular. Therefore, a different propagation order is imperative.

To solve the above problems, we propose two modifications:

1) To increase the probability of finding a good guess, we replace the point-based searching process with patch-based searching process. Concretely, assume b is the randomly chosen match of a by the PatchMatch algorithm. Let $S(a,b)$ be the similarity function of patch a and b. The match we choose is defined as:
$$b' = \arg\min_{q \in Q}(S(a,q)). \quad (2)$$
2) To further accelerate the converging speed and solve the second problem above, we proposed a novel best-guess-centered propagation order. Let b^j denote a boundary patch center, and q^j denote the current match for b^j. The propagation starts from the currently best matched boundary patch to the neighbouring patches. The starting patch is defined as:
$$s = \arg\min_{b^j}(S(b^j, q^j)). \quad (3)$$

The performance of this algorithm and the comparison of the algorithms before and after the modification are shown in the result section.

B. 3D-PatchMatch based Point cloud completion

Occluded boundary detection. To complete a point cloud, the occluded region must be first detected. Here, a last-echo based method is proposed to detect the occluded boundary points. Concretely, we use all the last echo points as the detection center, and then use a multi-clue filtering method proposed in [11] to detect all the boundary points within a certain radius from the detection center. The detection method defines three kinds of criteria to describe the feature of a boundary point: 1) the Angle Criterion, 2) the Halfdisk Criterion, and 3) the Shape Criterion.

After detecting the occluded boundary, an iterative completion process is proposed as shown in the right part of Fig. 1.

ICP based best match searching. After the occluded boundary is detected, a context based completion method is proposed to fill in the missing region. A best matched candidate patch is selected for every boundary patch based on the 3D-PatchMatch algorithm. For every pair of the candidate patch and the boundary patch, we first transform them into a local coordinate in which their centers are at the same location. Let S^{b^j} denote a boundary patch whose center is at b^j. Assume S^{q^k} is a candidate patch located at q^k. Then the transformed candidate patch $S_t^{q^k}$ is defined as:

$$S_t^{q^k} = \{s_t^i = s^i + b^j - q^k \mid s^i \in S_t^{q^k}\}. \quad (4)$$

After the transformation, we use the Iterative Closest Point (ICP) algorithm [12] to match S^{b^j} to $S_t^{q^k}$ and get the average distance d after matching as the similarity function S in the 3D-PatchMatch algorithm. Let T denote the transformation matrix of ICP, we use T^{-1} to transform the candidate patch to the boundary patch.

Pattern-preserving completion. If we simply fill the best matched patch to the boundary patch, the point distribution pattern will be destroyed. Therefore, we first analyse the influence radius (IR) of every boundary patch and its best matched patch and restore the point distribution pattern while completion. The IR for a boundary patch, S^{b^j} is defined as follows:

$$IR^{b^j} = \frac{\min(\| b^j - q^{\theta_1} \| + \| b^j - q^{\theta_2} \|)}{2}, \quad (5)$$

where $b^j, q^{\theta_1}, q^{\theta_2} \in S^{b^j}$ and $q^{\theta_1} \neq q^{\theta_2}$.

Let $S_{transformed}^{b^j}$ denote the transformed best match, a point in $S_{transformed}^{b^j}$ is filled into the boundary patch if it cannot find another point within IR. In this way, the point distribution pattern is preserved.

Boundary update. The boundary is updated after we complete a boundary patch, S^{b^j}. Because we use the IR to preserve the point distribution pattern, most of the newly created points are located in the missing region of the boundary patch. Therefore, to save the time cost of boundary detection, we update the new boundary patch center, b_{new}^j using the mean direction of all the newly created points to the boundary patch center b^j. The boundary update algorithm is shown as follows:

1) Let $S_{new}^{b^j}$ denote the complete boundary patch, we first find b_{new}^j as the new boundary point for planar patches, which is defined as follows:

$$b_{new}^j = \arg\min_{p \in S_c^{b^j}}(\| p - v_{mean} \|), \quad (6)$$

where, $S_c^{b^j} = S_{new}^{b^j} - S^{b^j}$ and v_{mean} is defined as:

$$v_{mean} = (b^j + \frac{\sum_{p \in S_c^{b^j}} \frac{(p - b^j)}{\| p - b^j \|}}{| S_c^{b^j} |} * r), \quad (7)$$

where, r is the patch radius. In practice, the patch radius is usually set to 15 cm by our experience.

2) Because b_{new}^j would be far away from the boundary when dealing with non-planar structures. Therefore, if

$$| b_{new}^j - b^j \| < \frac{2}{3} r, \quad (8)$$

find the new boundary point in the same direction but whose distance to the center point is closest to patch radius r. The boundary point, $b_{new}^{j'}$ for non-planar patches is defined as follows:

$$b_{new}^{j'} = \arg\min_{p \in S_c^{b^j}}(p - (b^j + \frac{b_{new}^j - b^j}{\| b_{new}^j - b^j \|} * r)). \quad (9)$$

Iteration and converging conditions. During the iteration, a completion priority similar to Criminisi et al. [13] is defined to make the algorithm complete patches with more points first. For every boundary patch, S^{b^j}, a weight function is defined as:

$$w(S^{b^j}) = \frac{| S^{b^j} |}{| S^{b^j} | + 1}. \quad (10)$$

The patch with highest weight is completed first.

After a boundary point is updated, the converging condition is checked for the completion process to halt. The converging condition is consisted of two parts:

1) After finding the best match, $S_{transformed}^{b^j}$ for a boundary patch S^{b^j}, if

$$\frac{| S^{b^j} |}{| S_{transformed}^{b^j} |} \leq 1, \quad (11)$$

we say the boundary patch is already complete because the candidate patch carries less information than the boundary patch.

2) After a boundary point is updated, if

$$\frac{| S^{b^j} |}{| S_{new}^{b^j} |} \leq 1, \quad (12)$$

we say the boundary patch is already complete because the new boundary patch is more complete than the original patch.

During the iteration, a boundary patch is removed from the boundary patch set if it meets either condition. And if there is no boundary patch left for completion or the iteration time exceeds a predefined threshold, we say the completion has converged. In the real-world point cloud, the missing region is not always closed. Therefore, a threshold is set depending on the size of the missing region to prevent the algorithm from misconvergence. In practice, the threshold is set to 10 times of the number of detected boundary points by our experience.

IV. Experimental Results

A. Implementation

The data of our experiments is acquired by a RIEGL VZ-1000 system and a RIEGL VMX-450 system. The proposed method is implemented using C++ and executed on a computer with Intel Core(TM) i3-2120 3.30GHz CPU and 4.0GB RAM.

B. Results

The completion results are shown in Fig. 3. As shown in the right part of Fig. 3. The occluded regions of all the demonstrated data are accurately restored. Meanwhile, there are little local minima generated by the 3D-PatchMatch algorithm. The time cost of the proposed algorithm is shown in Table 1. The algorithm reduced higher than 50% of the completion time cost comparing to searching with the naive brute force method when the searching space is large.

C. Evaluation & Comparison

The time cost using the point-based 3D-PatchMatch algorithm and patch-based 3D-PatchMatch algorithm is shown in Table 1. For point-based search, it is harder to find a good guess randomly when the searching space is large. Therefore, in the third data, the point-based search cost even more time than the brute force search while the patch-based search converged much faster. The time reduction rate tends to be larger when the searching space grows.

We compared our completion results with the method proposed in [8]. By transforming the 3D point cloud into a depth image, it uses an image in-painting method proposed in [13] to complete the depth image and then transforms the depth image back to point cloud. The comparative result is shown in Fig. 4. As shown in the left part of Fig. 4, the point distribution pattern of the occluded region was damaged after completion. The density of the occluded region is much larger than the complete region. By contrast, the completion results of our method restore the pattern well.

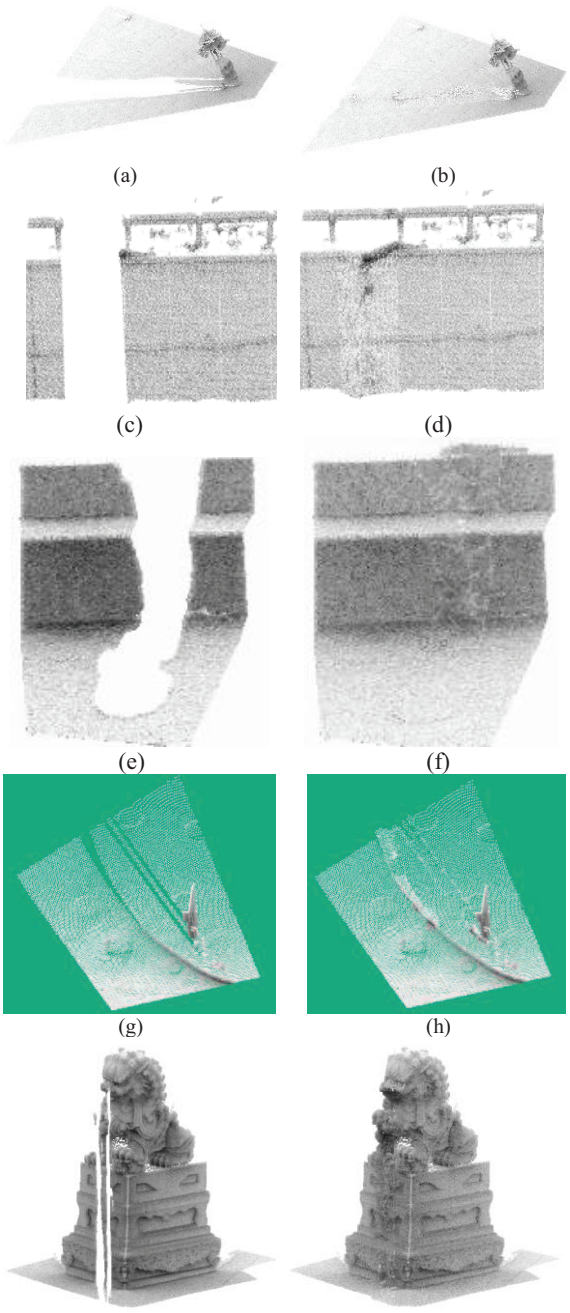

Fig. 3 The completion result. Left: input data with occlusion. Right: completion results.

TABLE I
COMPARISON OF POINT-BASED SERACH AND PATCH-BASED SEARCH

number of points (size of the searching space)	Time cost (s)		
	naive brute force search	Point-based search	Patch-based search
29539	42.861	41.99	38.11
79034	868.3	441.302	306.014
270231	12293.7	14556.6	4099.15

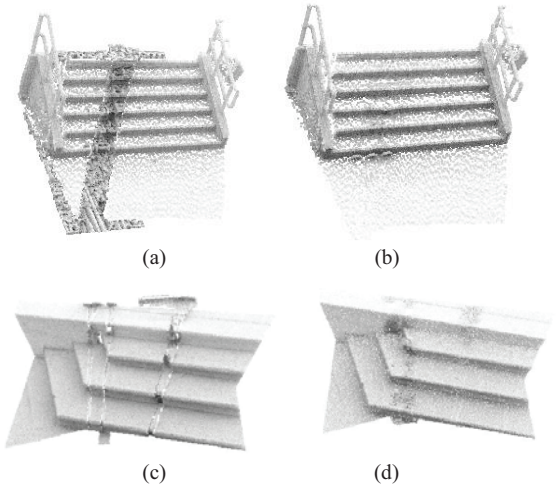

Fig. 4 Comparitive results. Left: Results of Doria et al.. Right: Results of the proposed method

V. Acknowledgement

This research is supported by the National Natural Science Foundation of China (Project No. 61371144 and 61401382).

VI. Conclusion

This paper proposed a novel 3D-PatchMatch algorithm for optimization during the searching process of point cloud completion. The algorithm utilizes the natural coherence of 3D point clouds to quickly find good matches for some of the boundary patches and then propagates the good matches to nearby area. The random guess process provides the algorithm with the ability to escape from local minima. We also proposed a general pattern-preserving point cloud completion framework to test the ability of the proposed algorithm. All the demonstrated results restored the occluded region well using the proposed framework. A fixed patch size might be a limitation of the completion framework since small patch size might not represent the feature of the neighbourhood well and large patch size might increase the time cost of matching process. Therefore, our future work will focus on adaptively choosing the patch size.

References

[1] G. H. Bendels, R. Schnabel, and R. Klein, "Detail-preserving surface inpainting," in *Proceedings of the 6th International conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage*, 2005, pp. 41-48.

[2] A. Sharf, M. Alexa, and D. Cohen-Or, "Context-based surface completion," in *ACM Transactions on Graphics (TOG)*, 2004, pp. 878-887.

[3] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics-TOG,* vol. 28, p. 24, 2009.

[4] M. Pauly, N. J. Mitra, J. Giesen, M. H. Gross, and L. J. Guibas, "Example-based 3D scan completion," in *Symposium on Geometry Processing*, 2005, pp. 23-32.

[5] S. Park, X. Guo, H. Shin, and H. Qin, "Shape and appearance repair for incomplete point surfaces," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, 2005, pp. 1260-1267.

[6] C. Xiao, W. Zheng, Y. Miao, Y. Zhao, and Q. Peng, "A unified method for appearance and geometry completion of point set surfaces," *The Visual Computer,* vol. 23, pp. 433-443, 2007.

[7] S. Friedman and I. Stamos, "Online facade reconstruction from dominant frequencies in structured point clouds," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012, pp. 1-8.

[8] D. Doria and R. J. Radke, "Filling large holes in LiDAR data by inpainting depth gradients," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, 2012, pp. 65-72.

[9] Q. Zheng, A. Sharf, G. Wan, Y. Li, N. J. Mitra, D. Cohen-Or*, et al.*, "Non-local scan consolidation for 3D urban scenes," *ACM Transactions on Graphics-TOG,* vol. 29, p. 94, 2010.

[10] A.-L. Chauve, P. Labatut, and J.-P. Pons, "Robust piecewise-planar 3D reconstruction and completion from large-scale unstructured point data," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 1261-1268.

[11] G. H. Bendels, "Detecting holes in point set surfaces," 2006.

[12] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision,* vol. 13, pp. 119-152, 1994.

[13] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2003, pp. II-721-II-728 vol. 2.