Dynamic Point Cloud Inpainting via Spatial-Temporal Graph Learning

Zeqing Fu, *Student Member, IEEE,* Wei Hu, *Member, IEEE*

***Abstract*—The maturity of depth sensors and laser scanning techniques has enabled the convenient acquisition of 3D dynamic point clouds—one natural representation of 3D objects/scenes in motion, leading to a wide range of applications such as immersive tele-presence, autonomous driving, augmented and virtual reality. Nevertheless, dynamic point clouds usually exhibit holes of missing data, thus inpainting is crucial to the subsequent rendering or downstream understanding tasks. Dynamic point cloud inpainting has been largely overlooked so far, which is also quite challenging due to the irregular sampling patterns both in the spatial domain and temporal domain. To this end, we propose an efficient dynamic point cloud inpainting method based on a learnable spatial-temporal graph representation, exploiting both the second-order inter-frame coherence and the intra-frame self-similarity. The key is the second-order inter-frame coherence that enforces the consistent flow in 3D motion over time, for which we search the temporal correspondence in consecutive frames for the same underlying surface by the point-to-plane distance and represent the correlation between them via temporal edge weights in the graph. Based on the second-order inter-frame coherence and intra-frame self-similarity, we formulate dynamic point cloud inpainting as a joint optimization problem of the desired point cloud and underlying spatial-temporal graph, which is regularized by consistency in the temporal edge weights and smoothness in the spatial domain. We analyze and reformulate the optimization, leading to an efficient alternating minimization algorithm. Experimental results show that the proposed approach outperforms several competing methods significantly, both on synthetic holes and real holes.**

***Index Terms*—3D dynamic point clouds, inpainting, spatial-temporal graph, inter-frame coherence.**

I. Introduction

3D dynamic point cloud has received increasing attention as an efficient representation of arbitrarily-shaped 3D objects or scenes in motion. It consists of a sequence of point clouds, each of which is a set of discrete points sampled from the continuous surface of objects or scenes. Each point in a point cloud corresponds to a measurement point and contains most original information of the point, including the 3D coordinates representing the geometric information and possibly attribute information such as color and normal. The development of depth sensing and laser scanning techniques[1] has enabled the convenient acquisition of dynamic point clouds, with a wide range of applications such as immersive tele-presence, autonomous driving, augmented and virtual reality [1].

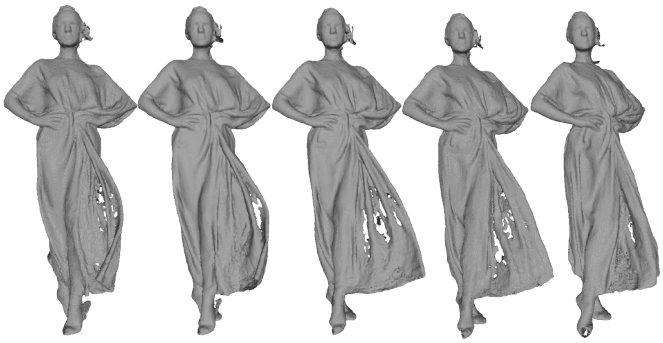


Fig. 1. Several consecutive frames of the 3D dynamic point cloud *Long-dress* [2] with holes.

Nevertheless, 3D dynamic point clouds often exhibit several holes of missing data inevitably, as shown in Fig. 1. This is mainly due to incomplete scanning views, fast object motion and inherent limitations of the acquisition equipments[2]. Besides, there may lack some regions in the data itself (*e.g.*, dilapidated heritage). Therefore, it is necessary to inpaint incomplete point clouds prior to the subsequent applications. Nevertheless, dynamic point cloud inpainting is quite challenging, because each point cloud frame is irregularly sampled and different frames have varying sampling patterns with possibly different numbers of points, which means there is no explicit temporal correspondence between points over time.

However, the direct inpainting of dynamic point clouds has been largely overlooked so far in the literature, while many approaches have been proposed for static point clouds [3]–[18]. If we apply them to point cloud sequences frame by frame, the inpainting process of each frame is independent to each other, which neglects the inter-frame correlation and thus is sub-optimal. Fu *et al.* proposed dynamic point cloud inpainting by exploiting the inter-frame consistency and the intra-frame self-similarity [19]. However, the inter-frame consistency is only enforced between each pair of adjacent point cloud frames, which may not capture the consistent *change* of 3D motion over several consecutive frames—consistent motion flow. Also, the graph representation in [19] is empirically constructed, and the search for temporally corresponding cubes of points could be further improved.

Z. Fu (e-mail: zeqing_fu@pku.edu.cn) and W. Hu (e-mail: forhuwei@pku.edu.cn) are with Wangxuan Institute of Computer Technology, Peking University, No. 128, Zhongguancun North Street, Beijing, China.

Corresponding author: Wei Hu (forhuwei@pku.edu.cn). This work was supported in part by the National Key R&D project of China under contract No. 2019YFF0302903, in part by Beijing Natural Science Foundation [19L2053], and in part by National Natural Science Foundation of China [61972009].

[1]Commercial products include Microsoft Kinect (2010-2014), Intel RealSense (2015-), Velodyne LiDAR (2007-2020), LiDAR scanner of Apple iPad Pro (2020), *etc.*

[2]For example, laser scanning is less sensitive to the objects in dark colors. This is because the darker it is, the more wavelengths of light the surface absorbs and the less light it reflects. Thus the laser scanning devices are unable to receive enough reflected light for dark objects to recognize.

To this end, we propose to inpaint the geometry of dynamic point clouds by exploiting the *second-order inter-frame coherence* and the intra-frame self-similarity, based on a *learnable* spatial-temporal graph representation to capture the underlying structure of irregular point cloud frames. The key is the second-order inter-frame coherence based on the observation that the flow in 3D motion is often consistent over time. It is meanwhile challenging, since there is no explicit point-to-point correspondence between adjacent point cloud frames as mentioned. To address this issue, given consecutive frames, we propose to search local cubes of points among the frames that correspond to the same underlying surface—referred to as *corresponding cubes* via a *point-to-plane* metric, which captures the temporal distance between the surfaces in cubes better than [19]. We then learn the correlation between them as the temporal edge weights in the spatial-temporal graph. Further, the temporal edge weights between each pair of corresponding cubes are constrained to be similar over different pairs so as to exploit the temporally consistent motion flow, *i.e.*, the second-order inter-frame coherence.

Based on the second-order inter-frame coherence and our previously proposed intra-frame self-similarity [17] that exploits regions with similar geometry, we formulate dynamic point cloud inpainting as a joint optimization problem of the desired point cloud and the underlying spatial-temporal graph. The optimization is regularized by 1) the second-order inter-frame coherence, formulated as the l_1 norm of the difference in the temporal edge weights among consecutive frames and 2) intra-frame self-similarity, formulated as a graph-signal smoothness prior [20] in the spatial domain. To solve the optimization, we analyze and reformulate the objective, and then optimize the desired point cloud, the temporal edge weights and the spatial edge weights alternately. Experimental results show that the proposed approach outperforms several competing methods significantly on both synthetic holes and real holes.

Our contributions can be summarized as follows:

- We propose dynamic point cloud inpainting by exploiting the *second-order inter-frame coherence* and the intra-frame self-similarity, based on a *learnable* spatial-temporal graph representation for adaptive structural description.
- We propose the second-order inter-frame coherence to capture the consistent motion flow between corresponding cubes among consecutive frames, and formulate it as the l_1 norm of the difference in the temporal edge weights of the underlying spatial-temporal graph.
- We cast dynamic point cloud inpainting as the joint optimization of the desired point cloud and the underlying spatial-temporal graph, regularized by the second-order inter-frame coherence and the intra-frame self-similarity.
- We analyze and reformulate the optimization problem, and present an algorithm to solve it efficiently. Experimental results demonstrate the superiority of the proposed method.

The outline of the paper is as follows. We first discuss previous methods in Section II. Then we review relevant concepts and tools in graph signal processing in Section III. Next, we elaborate on the problem formulation in Section IV, and present the algorithm development in Section V. Experimental results and conclusions are presented in Section VI and VII, respectively.

II. Related Work

We review previous works on point cloud inpainting and spatial-temporal graph learning, respectively.

A. Point Cloud Inpainting

While few methods [19] study the direct inpainting of dynamic point clouds, we focus on the review of previous inpainting works on static point clouds.

1) Static Point Cloud Inpainting: According to the cause of holes, we divide static point cloud inpainting into two classes: 1) inpainting holes in the object itself such as heritage and sculptures [3]–[9], which is often large, and 2) inpainting holes caused by the limitation of scanning devices [10]–[18], which is comparatively small and fragmented.

For the first class of methods, the main hole-filling data source is online database, as the holes are often large. Sahay *et al.* [5] propose a gradient map and dictionary learning-based method to harness the geometric prior. They project the point cloud to a depth map, search a similar one from an online depth database via dictionary learning, and then minimize the error in the known region to ensure the smoothness in the formulation. However, the projection process inevitably introduces geometric loss. Instead, Dinesh *et al.* [8], [9] fill this kind of holes with the data in the object itself. In particular, they determine the inpainting order by giving priority to the points along the hole boundary, and search best matching regions based on the smallest rotation difference in order to fill the missing area. The results still suffer from geometric distortion due to the simple data source.

The second class of methods focus on holes generated due to the limitations of scanning devices. This kind of holes is smaller than the aforementioned ones in general, thus the information of the data itself is often enough for inpainting. Wang *et al.* [10] and Quinsat *et al.* [11] create a triangle mesh from the input point cloud, identify the vicinity of the hole to build the mesh over the hole, and finally interpolate the missing area. These methods rely on the quality of mesh construction though. Lozes *et al.* [12], [13] deploy partial difference operators to solve an optimization problem on the point cloud, which only refers to the neighborhood of the hole to compute the geometric structure. Muraki *et al.* [15] generate a surface to fit the vicinity of the hole and interpolate the surface for inpainting. Due to the reference information from the local neighborhood only, the results of these methods tend to be more planar than the ground truth. Also, artifacts are likely to occur around the boundary when the geometric structure is complicated. Fu, Hu *et al.* [16], [17] exploit the non-local similarity in the point cloud, which searches the most similar region to the missing region based on the normals of points, and fills the hole by formulating an optimization problem based on the similar region and a graph-signal smoothness prior.

Besides, some works for static point clouds deal with particular point cloud data such as geometrically regular point clouds of buildings in [21], flattened bar-shaped holes in the human body data in [22], and dynamic holes with static objects in [23], which are unsuitable for general cases though.

2) Dynamic Point Cloud Inpainting: Fu *et al.* [19] propose the geometry inpainting of dynamic point clouds, exploiting the inter-frame consistency and the intra-frame self-similarity based on a spatial-temporal graph representation. We extend this work from the following three aspects: 1) instead of considering only the first-order inter-frame consistency enforced between each pair of adjacent point cloud frames, we further exploit the second-order temporal coherence, which captures the consistency in changes/flows of 3D motion over several consecutive frames; 2) instead of empirical graph construction, we propose to learn the underlying spatial-temporal graph adaptively from the data and the regularization of the second-order temporal coherence; 3) we improve the search for temporally corresponding cubes of points by leveraging a point-to-plane metric instead of a point-to-point metric, which measures the distance between the underlying surfaces more accurately.

B. Spatial-Temporal Graph Learning

While many efforts have been made for graph learning [24]–[26], few attempts explore spatial-temporal graph learning.

Hallac *et al.* [27] address spatial-temporal graph learning by time-varying graphical Lasso, which combines graphical Lasso with a temporal regularization and acquires the solution using alternating direction method of multipliers. The graphs estimated by this approach often have negative edge weights, whereas edge weights in our problem formulation are constrained to be non-negative in the context of point clouds.

Kalofolias *et al.* [28] and Yamada *et al.* [29] learn spatial-temporal graphs by penalizing fast changes of the adjacency matrices to enforce smoothly varying edge weights over time. Different from their prior that graph edges at each time change smoothly over time, we assume the *change* in graph edges between adjacent point cloud frames varies consistently over time, *i.e.*, the second-order temporal coherence, which characterizes dynamic point clouds better. Also, we learn the edge weights between adjacent point clouds to capture the temporal correlation in addition to edge weights at each time.

Besides, [30]–[34] learn spatial-temporal graphs in graph convolutional neural networks from multiple observations, while our method learns the graph from the input single observation. In addition, Baingana *et al.* [35] aim to learn the time-varying graphs to capture causal relationships in the network. This model focuses on the learning of directed graphs by considering the spread process on networks along the time dimension, while our method focuses on learning undirected graphs for point clouds. Compared with deep learning based spatial-temporal graph learning, the proposed method is advantageous in the following three aspects: 1) The proposed method learns the graph from the input *single* observation in an unsupervised manner; in contrast, deep learning based methods often require a large amount of training data and the supervision of ground truth graphs. 2) The proposed method is more generalizable to various point clouds than deep learning based methods. 3) The proposed method is interpretable based on the second-order inter-frame coherence and the intra-frame self-similarity, while deep learning based methods often lack interpretability.

III. Spectral Graph Theory

We first provide a review on basic concepts in spectral graph theory [36] and graph signal processing [37]–[39], including graph, graph Laplacian and graph-signal smoothness prior, which will be leveraged in the proposed dynamic point cloud inpainting.

A. Graph and Graph Laplacian

We consider an undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ composed of a vertex set $\mathcal{V}$ of cardinality $|\mathcal{V}| = N$, an edge set $\mathcal{E}$ connecting vertices, and a weighted *adjacency matrix* $\mathbf{W}$. $\mathbf{W}$ is a real symmetric $N \times N$ matrix, where $w_{i,j}$ is the weight assigned to the edge (i, j) connecting vertices i and j. We assume non-negative weights, *i.e.*, $w_{i,j} \geqslant 0$. For example, the K-Nearest Neighbor (K-NN) graph is a commonly used undirected graph, which is constructed by connecting each point with its nearest K neighbors.

The Laplacian matrix is defined from the adjacency matrix [36]. Among different variants of Laplacian matrices, the commonly used *combinatorial graph Laplacian* [40]–[42] is defined as $\mathcal{L} := \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is the *degree matrix*—a diagonal matrix where $d_{i,i} = \sum_{j=1}^{N} w_{i,j}$.

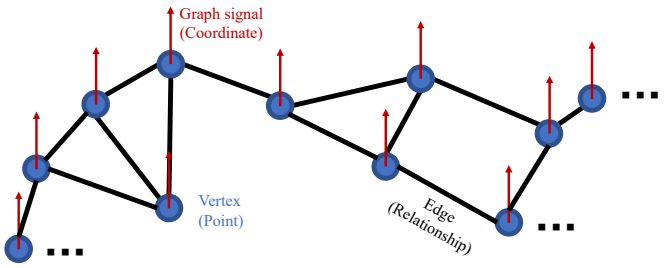


Fig. 2. A K-NN graph constructed when $K = 3$ with graph signal (red arrows). The connections of boundary vertices are omitted.

B. Graph-Signal Smoothness Prior

Graph signal refers to data residing on the vertices of a graph. For example, if we construct a K-NN graph on the point cloud, then the normal or the coordinate of each point can be treated as graph signal defined on the K-NN graph, as shown in Fig. 2. In our context, the graph signal is the normal in the intra-source cube searching approach in Section IV-B, while it refers to the coordinates of points in the problem formulation in Section IV-E.

A graph signal $\mathbf{z}$ defined on a graph $\mathcal{G}$ is smooth with respect to the topology of $\mathcal{G}$ if

$$\sum_{i \sim j} w_{i,j}(z_i - z_j)^2 < \epsilon, \quad \forall i, j, \tag{1}$$

where ϵ is a small positive scalar, and $i \sim j$ denotes two vertices i and j are one-hop neighbors in the graph. In order to satisfy Eq. (1), z_i and z_j have to be similar for a large edge weight $w_{i,j}$, and could be quite different for a small $w_{i,j}$. Hence, Eq. (1) enforces $\mathbf{z}$ to adapt to the topology of $\mathcal{G}$, which is thus called *graph-signal smoothness prior*. This prior also possesses an interpretation in the frequency domain as low-pass filtering, as well as a continuous interpretation as a smoothness functional defined on the underlying Riemannian manifold [43].

As $\mathbf{z}^\top \mathcal{L}\mathbf{z} = \sum_{i\sim j} w_{i,j}(z_i - z_j)^2$ [44], Eq. (1) is concisely written as $\mathbf{z}^\top \mathcal{L}\mathbf{z} < \epsilon$ in the sequel. This prior will be deployed in our problem formulation of point cloud inpainting as a regularization term for spatial-temporal smoothness, as detailed in Section IV-E and Section V-A.

IV. Problem Formulation

We now introduce the proposed point cloud inpainting method, leveraging on the spectral graph theory in Section III. The input data is a point cloud sequence denoted by $\mathbf{S} = \{\mathbf{P}_1, \mathbf{P}_2, ..., \mathbf{P}_q\}$, where $\mathbf{P}_f, f = 1, ..., q$ denotes each frame of point cloud in the sequence. As shown in Fig. 3, we process each target frame $\mathbf{P}_f$ with holes in order with the following steps:

1) We split $\mathbf{P}_f$ into cubes of fixed size as units to be processed in the subsequent steps.
2) We choose the target cube with missing area manually.
3) We search for the most similar cube to the target cube in $\mathbf{P}_f$, which is referred to as the *intra-source cube*, based on the variation in normals of points as in our previous work [17].
4) We search for the corresponding cubes to the target cube both in $\mathbf{P}_{f-1}$ and $\mathbf{P}_{f+1}$, which is referred to as the *inter-source cubes*. The idea is to search the cube with the most nearest neighbors of the points in the target cube, in which the distance measure between two points is point-to-plane for capturing surface structures.
5) We formulate the inpainting problem into an optimization problem, which leverages the intra- and inter-source cubes, the graph-signal smoothness prior of the spatial-temporal graph as well as a second-order temporal coherence term in the temporal edge weights. We analyze and reformulate the objective, and then solve the optimization problem iteratively, leading to the resulting cube.
6) We replace the target cube with the resulting cube as the output.

A. Preprocessing

Given a target frame of point cloud $\mathbf{P}_f = \{\mathbf{p}_1, \mathbf{p}_2, ...\}$ with $\mathbf{p}_i \in \mathbb{R}^3$ meaning the coordinates of the i-th point in the point cloud, we first split $\mathbf{P}_f$ into overlapping cubes $\{\mathbf{c}_1, \mathbf{c}_2, ...\}$ with $\mathbf{c}_i \in \mathbb{R}^{M^3 \times 3}$ (M is the size of the cube), as the processing unit of the proposed inpainting algorithm. M is empirically set according to the coordinate range of $\mathbf{P}_f$ ($M = 20$ in our experiments), while the overlapping step is empirically set as $\frac{M}{4}$. This is a trade-off between the computational complexity and ensuring enough geometry information available to search for source cubes.

Having obtained the cubes, we choose the cube with missing data as the target cube $\mathbf{c}_t$ manually. In order to ensure there is enough known information available in $\mathbf{c}_t$ for similar cube search, we constrain that the percentage of the hole in a cube

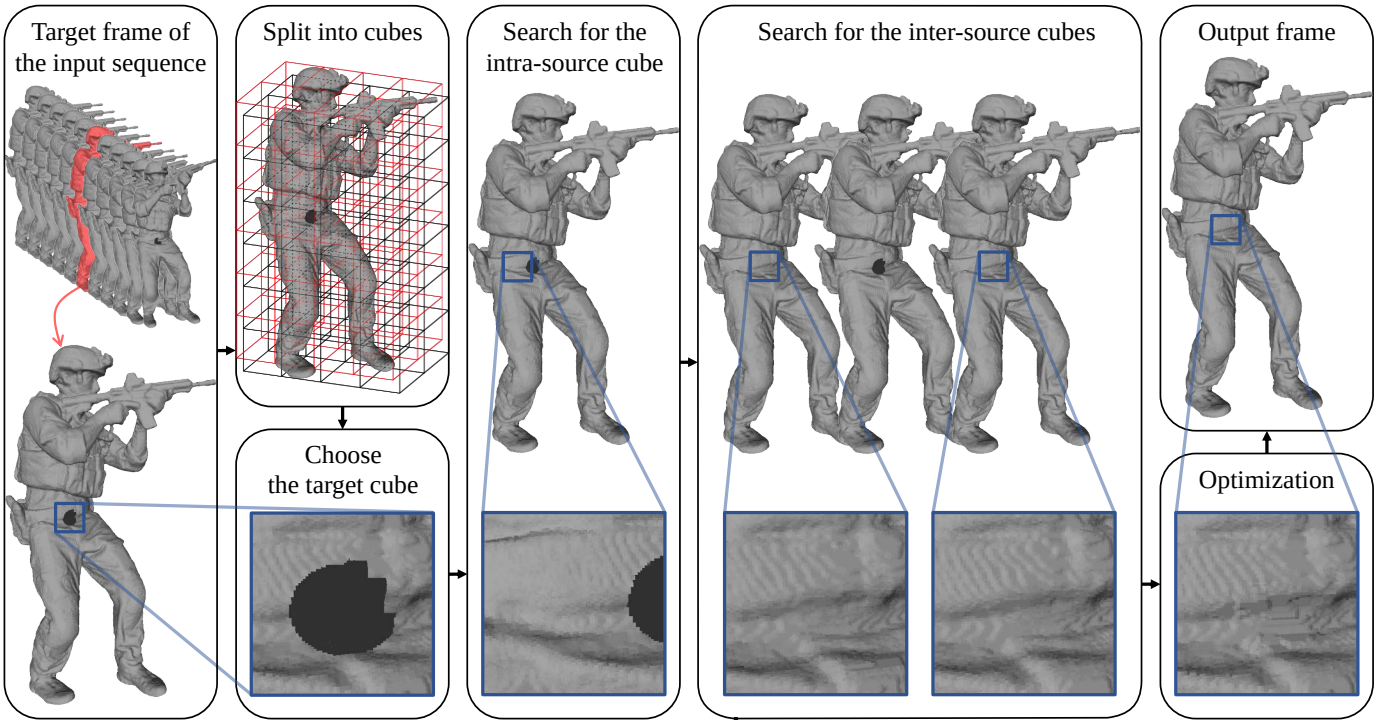


Fig. 3. The framework of the proposed 3D dynamic point cloud inpainting method.

should cover less than 50% of the total points when projected to a 2D depth map. Besides, in the presence of a hole larger than the cube size, we will divide the hole into several small holes, and then inpaint the small holes in the inward order.

In addition, in order to save the computation complexity and increase the accuracy of the subsequent cube matching, we choose candidate cubes $\mathbf{c}_c$ for intra-source cube searching by filtering out cubes with the number of points less than 80% of that of $\mathbf{c}_t$. We test that, around 67.96% of cubes are filtered out on average, which reduces the computation complexity of the intra-source cube matching significantly, given that the running time for each cube matching takes about 0.12 second. Also, we *augment* the candidates by mirroring these cubes with respect to the *x*-*y* plane, which has been validated to be useful in the previous work [17] and will be employed in the next step.

B. Intra-frame Self-Similar Cube Matching

In order to search for the most similar cube to $\mathbf{c}_t$ in $\mathbf{P}_f$, we first define the geometric similarity metric $\delta(\mathbf{c}_t, \mathbf{c}_c)$ between the target cube $\mathbf{c}_t$ and each candidate cube $\mathbf{c}_c$ as in [16], [17], by the difference in Direct Component (DC) and Anisotropic Graph Total Variation (AGTV) of normals between $\mathbf{c}_t$ and $\mathbf{c}_c$:

$$\delta(\mathbf{c}_t, \mathbf{c}_c) = \exp\{-[\,|\langle \mathbf{d}(\mathbf{c}_t),\ \mathbf{d}(\mathbf{c}_c)\rangle| + |v(\mathbf{c}_t) - v(\mathbf{c}_c)|\,]\}, \quad (2)$$

where $\mathbf{d}(\mathbf{c}_t)$ and $\mathbf{d}(\mathbf{c}_c)$ are the DC of cube $\mathbf{c}_t$ and $\mathbf{c}_c$, while $v(\mathbf{c}_t)$ and $v(\mathbf{c}_c)$ are the AGTV of $\mathbf{c}_t$ and $\mathbf{c}_c$.

Specifically, as defined in [17], the DC is essentially an average of the normals, which presents the *prominent geometry direction* of the cube. The AGTV is a variant of graph total variation, which describes the variation of normals in the cube with respect to the underlying graph structure.

Having computed the similarity metric in Eq. (2) between the target cube and all candidate cubes in $\mathbf{P}_f$, we choose the candidate cube with the largest similarity as the intra-source cube $\mathbf{c}_s$. However, $\mathbf{c}_s$ cannot be directly adopted for inpainting, because it is just the most similar to $\mathbf{c}_t$ in the geometric structure, but not in the relative location in the cube. Hence, we further perform structure matching (*i.e.*, coarse registration) for $\mathbf{c}_s$ and $\mathbf{c}_t$ so as to match the relative locations as in [17], which includes both translation and rotation as a simplified Iterative Closest Points (ICP) algorithm [45], [46]. This leads to the final intra-source cube, denoted as $\hat{\mathbf{c}}_s$, which will be adopted in the final inpainting step.

C. Inter-frame Corresponding Cube Matching

Considering that the inpainted results of dynamic point clouds should be coherent among consecutive frames, it is necessary to search the temporal corresponding cubes between neighboring frames in a point cloud sequence, which correspond to the same local underlying surface in the 3D object or scene. Unlike videos, dynamic point clouds are irregular, thus the temporal correspondence is challenging to search. Few methods explore this problem for the compression of dynamic point clouds [47]–[49], which compare the difference of the octree data structure [47], the local features of each point based on spectral graph wavelets [48], or 2D patches projected from 3D dynamic point clouds [49]. However, there exist the limitations of the octree structure, the expensive computation complexity, or the projection loss.

In order to efficiently explore the temporal coherence in dynamic point clouds, we propose to find *corresponding cubes* for $\mathbf{c}_t$ both in $\mathbf{P}_{f-1}$ and $\mathbf{P}_{f+1}$, which describe the same underlying surface as $\mathbf{c}_t$ at different time and are denoted by $\mathbf{c}_t^{f-1}$ and $\mathbf{c}_t^{f+1}$ respectively as the inter-source cubes. This is realized by searching the nearest neighbor of each point in terms of a *point-to-plane* distance metric, which is simple yet effective. Note that, the inter-frame coherence can be generalized to several previous and subsequent frames, instead of one forward and one backward as in our method.

Specifically, inspired by the observation that a set of points representing the same region have little variation in the consecutive frames, we find the temporal coherence via searching the nearest neighbor of each point in the target cube in terms of a *point-to-plane* distance metric. As shown in Fig. 4, we first find the cube $\mathbf{c}_t' \in \mathbb{R}^{M^3 \times 3}$ in $\mathbf{P}_{f-1}$ (one green cube in Fig. 4) as

$$\mathbf{s}(\mathbf{c}_t') = \mathbf{s}(\mathbf{c}_t), \quad (3)$$

where $\mathbf{s}(\mathbf{c}_t')$ denotes the coordinate of the centering point of $\mathbf{c}_t'$, and $\mathbf{s}(\mathbf{c}_t)$ denotes the coordinate of the center of $\mathbf{c}_t$. This means $\mathbf{c}_t'$ is collocated as $\mathbf{c}_t$ in the relative location.

Then we create a bounding box $\mathbf{b}_t^{f-1} \in \mathbb{R}^{H^3 \times 3}$ (H is the size of the bounding box) around $\mathbf{c}_t'$ as

$$\mathbf{s}(\mathbf{b}_t^{f-1}) = \mathbf{s}(\mathbf{c}_t'), \quad (4)$$

where $\mathbf{s}(\mathbf{b}_t^{f-1})$ is the coordinate of the centering point of $\mathbf{b}_t^{f-1}$. Then, in $\mathbf{b}_t^{f-1}$, we search the nearest neighbor of each point in $\mathbf{c}_t$ in terms of a point-to-plane metric, as illustrated with yellow lines in Fig. 4. In particular, the point-to-plane distance $\mathrm{dis}_{k,l}$ between the point $\mathbf{q}_{t,k}$ in $\mathbf{c}_t$ and the point $\mathbf{q}_{t,l}^{f-1}$ in $\hat{\mathbf{c}}_t^{f-1}$ is defined as in [50]:

$$\mathrm{dis}_{k,l} = \|(\mathbf{q}_{t,k} - \mathbf{q}_{t,l}^{f-1}) \cdot \mathbf{n}_{t,k}\|_2^2, \quad (5)$$

where the distance vector between two points is calculated as the difference of their coordinates, and the unit normal vector $\mathbf{n}_{t,k}$ is the normal vector of the point $\mathbf{q}_{t,k}$ in $\mathbf{c}_t$. Thus the point-to-plane distance is the projection of the point-to-point

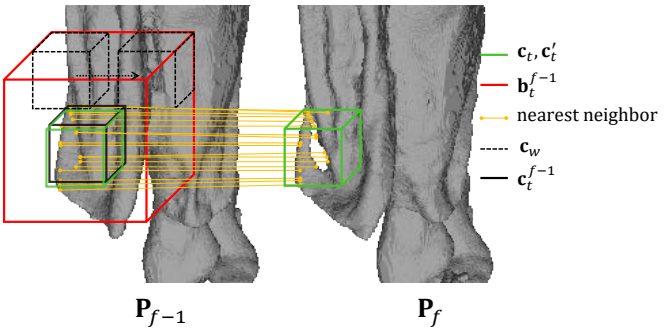


Fig. 4. The inter-source cube searching. The green cubes in $\mathbf{P}_f$ and $\mathbf{P}_{f-1}$ are $\mathbf{c}_t$ and $\mathbf{c}_t'$, respectively. The red cube is the searching box $\mathbf{b}_t^{f-1}$, the yellow lines connect the nearest neighbors in relative location to each point in $\mathbf{c}_t$, and the black dotted cube is the sliding cubic window $\mathbf{c}_w$ to find the inter-source cube $\mathbf{c}_t^{f-1}$ in black.

distance along the normal of the point in $\mathbf{c}_t$, which reflects the distance to the tangent plane for structural characterization.

Next, we create a sliding cubic window $\mathbf{c}_w \in \mathbb{R}^{M^3\times 3}$ in the bounding box $\mathbf{b}_t^{f-1}$. The inter-source cube $\mathbf{c}_t^{f-1} \in \mathbb{R}^{M^3\times 3}$ in $\mathbf{P}_{f-1}$ is found by

$$\mathbf{c}_t^{f-1} = \arg\max_{\mathbf{c}_w} V(\mathbf{c}_w), \tag{6}$$

where $V(\mathbf{c}_w)$ is the number of the nearest neighbors of $\mathbf{c}_t$ in $\mathbf{c}_w$ in terms of the point-to-plane distance. That is, $\mathbf{c}_t^{f-1}$ contains the most structurally nearest neighbors of $\mathbf{c}_t$.

However, $\mathbf{c}_t^{f-1}$ is just the most relevant cube in $\mathbf{P}_{f-1}$ to $\mathbf{c}_t$ in the temporal correlation. There may exist a rigid transformation between $\mathbf{c}_t^{f-1}$ and $\mathbf{c}_t$ due to the motion. Thus, we perform the same structure matching on $\mathbf{c}_t^{f-1}$ as the way we deal with $\mathbf{c}_s$ in Section IV-B, which leads to the final inter-source cube in $\mathbf{P}_{f-1}$, denoted as $\hat{\mathbf{c}}_t^{f-1}$. The final inter-source cube in $\mathbf{P}_{f+1}$, denoted by $\hat{\mathbf{c}}_t^{f+1}$, is searched in the same way as in $\mathbf{P}_{f-1}$. Thus we obtain two source cubes as the temporal reference, which will be adopted in the graph initialization and the problem formulation as follows.

D. Spatial-Temporal Graph Initialization

In order to exploit the spatial-temporal correlation, we propose to build a *triple-cube* $\mathbf{g} \in \mathbb{R}^{3M^3\times 3}$ from the target cube $\mathbf{c}_t$ and its inter-frame corresponding cubes as

$$\mathbf{g} = \begin{bmatrix} \hat{\mathbf{c}}_t^{f-1} \\ \mathbf{c}_t \\ \hat{\mathbf{c}}_t^{f+1} \end{bmatrix}, \tag{7}$$

and construct a spatial-temporal graph $\mathcal{G}_g = \{\mathcal{V}_g, \mathcal{E}_g, \mathbf{W}_g\}$ over $\mathbf{g}$ as an initial representation of the desired cube $\mathbf{c}_r$, which will be updated during the optimization. We discuss the construction of spatial connectivities and temporal connectivities in order.

1) Spatial Graph Initialization: Due to the hole in $\mathbf{c}_t$, the spatial correlation within $\mathbf{c}_t$ is incomplete from the observation. Instead, we approximate such spatial correlation with that of its intra-frame self-similar cube $\hat{\mathbf{c}}_s$.

As in [19], we choose to build a K-NN graph as mentioned in Section III-A, based on the affinity of geometric distance among points in $\hat{\mathbf{c}}_s$. The edge weight $w_{k,l}$ between nodes k and l in $\hat{\mathbf{c}}_s$ is assigned as a thresholded Gaussian function of the distance between points:

$$w_{k,l} = \begin{cases} \exp\{-\frac{\|\mathbf{p}_k - \mathbf{p}_l\|_2^2}{2\sigma^2}\}, & k \sim l \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where σ is a weighting parameter (empirically $\sigma = 1$ in our experiments). This is based on the assumption that geometrically closer points are more similar in general.

2) Temporal Graph Initialization: We connect points in each pair of corresponding cubes as the temporal edges. Due to the hole in $\mathbf{c}_t$, we approximate the connectivity associated with the unknown region of $\mathbf{c}_t$ by the corresponding region in $\hat{\mathbf{c}}_s$. Whereas there is no explicit point-to-point correspondence in the temporal domain, we connect points that are close in terms of the projection distance. In particular, inspired by the method in [51], we compare the distance from each point in the corresponding cubes to the tangent plane at one (pseudo) cube center, which captures the similarity in geometric curvature of cubes.

Taking the connectivity between $\mathbf{c}_t$ and $\hat{\mathbf{c}}_t^{f-1}$ as an example, we first construct a reference plane at the pseudo center of $\mathbf{c}_t$—the nearest point $\mathbf{q}_{t,c}$ to the center of $\mathbf{c}_t$. The reference plane is tangent to $\mathbf{q}_{t,c}$ and perpendicular to the normal $\mathbf{n}_{t,c}$ at $\mathbf{q}_{t,c}$. Secondly, we project each point in $\mathbf{c}_t$ and $\hat{\mathbf{c}}_t^{f-1}$ onto the reference plane. For each point $\mathbf{q}_{t,k}$ in $\mathbf{c}_t$, we search a point $\mathbf{q}_{t,l}^{f-1}$ in $\hat{\mathbf{c}}_t^{f-1}$, whose projection on the tangent plane is closest to that of $\mathbf{q}_{t,k}$. If the nearest neighbor of $\mathbf{q}_{t,k}$ in terms of projection is too far, we take its three nearest neighbors in $\hat{\mathbf{c}}_t^{f-1}$ to compute a nearest plane for $\mathbf{q}_{t,k}$, and take the intersection point of the plane and the projection perpendicular at $\mathbf{q}_{t,k}$ as the nearest point $\mathbf{q}_{t,l}^{f-1}$.

Having connected points between $\mathbf{c}_t$ and $\hat{\mathbf{c}}_t^{f-1}$, we set the weight of each temporal edge as 1 or 0 for simplicity:

$$w_{k,l} = \begin{cases} 1, & \mathbf{q}_{t,k} \sim \mathbf{q}_{t,l}^{f-1} \\ 0, & \text{otherwise} \end{cases}. \tag{9}$$

It is the same for the connectivity between $\mathbf{c}_t$ and $\hat{\mathbf{c}}_t^{f+1}$.

Thus, we initialize the spatial-temporal graph $\mathcal{G}_g$ over $\mathbf{c}_t$ as an initial graph representation of the desired cube $\mathbf{c}_r$, which incorporates information from both intra-frame and inter-frame source cubes. Next, we discuss the problem formulation based on $\mathbf{g}$ and $\mathcal{G}_g$.

E. Problem Formulation

Based on the triple-cube $\mathbf{g}$ and the intra-frame source cube $\hat{\mathbf{c}}_s$, we cast the inpainting problem as an optimization problem, which is regularized by the graph-signal smoothness prior mentioned in Section III-B with respect to $\mathcal{G}_g$ and the l_1 norm of the difference in the temporal edge weights for keeping the second-order temporal coherence. This problem is formulated as

$$\begin{aligned} \min_{\mathbf{c}_r, \mathbf{W}_{f-1,f}, \mathbf{W}_{f+1,f}, \mathcal{L}_g} \quad & \|\overline{\mathbf{\Omega}}\mathbf{c}_r - \overline{\mathbf{\Omega}}\mathbf{c}_t\|_F^2 + \alpha\|\mathbf{\Omega}\mathbf{c}_r - \mathbf{\Omega}\hat{\mathbf{c}}_s\|_F^2 \\ & + \beta \mathrm{tr}(\mathbf{g}^\top \mathcal{L}_g \mathbf{g}) + \gamma\|\mathbf{W}_{f-1,f} - \mathbf{W}_{f+1,f}\|_1, \end{aligned} \tag{10}$$

where:

- $\mathbf{c}_r \in \mathbb{R}^{M^3\times 3}$ is the desired resulting cube.
- $\mathbf{\Omega}$ is a $M^3 \times M^3$ diagonal matrix with $\Omega_{i,i} \in \{0, 1\}$, where 0 indicates known points and 1 indicates missing points. Thus $\mathbf{\Omega}\mathbf{c}_r$ and $\mathbf{\Omega}\hat{\mathbf{c}}_s$ represent the missing region in $\mathbf{c}_r$ and $\hat{\mathbf{c}}_s$ respectively. $\overline{\mathbf{\Omega}}$ is complementary to $\mathbf{\Omega}$, which extracts the known region.
- $\mathbf{W}_{f-1,f}$ is the weight matrix of the temporal graph edges we construct between $\hat{\mathbf{c}}_t^{f-1}$ and $\mathbf{c}_r$, and $\mathbf{W}_{f+1,f}$ is the weight matrix between $\hat{\mathbf{c}}_t^{f+1}$ and $\mathbf{c}_r$. These two matrices will be discussed in detail later.
- $\mathcal{L}_g \in \mathbb{R}^{3M^3\times 3M^3}$ is the graph Laplacian matrix of the spatial-temporal graph constructed over $\mathbf{g}$. The graph-signal smoothness prior $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$ will be analyzed in Section V-A in detail.

- α, β and γ are three positive weighting parameters (we empirically set $\alpha = 1$, $\beta = 0.5$ and $\gamma = 5$ in the experiments).

The first term in Eq. (10) is a data fidelity term, which ensures the desired cube to be close to the observed $\mathbf{c}_t$ in the known region by minimizing the Frobenius norm of the difference between them. The second term constraints the missing region of $\mathbf{c}_r$ to be similar to that of its self-similar cube $\hat{\mathbf{c}}_s$ by minimizing the Frobenius norm of the difference between them in the missing region. The third term is the graph-signal smoothness prior with respect to the spatial-temporal graph, which enforces the spatial-temporal smoothness. Note that, we take the trace to compute the sum of the prior in the x-, y-, and z-coordinate. Further, the last term is the regularization for the second-order inter-frame coherence, which keeps the motion flow among consecutive frames temporally consistent. That is, by enforcing the temporal edges between $\{\hat{\mathbf{c}}_t^{f-1}, \mathbf{c}_r\}$ and those between $\{\mathbf{c}_r, \hat{\mathbf{c}}_t^{f+1}\}$ to be similar, we constrain the *change* in 3D motion to be smooth over time.

To solve Eq. (10) efficiently, we unfold the third term with some analysis, and reformulate Eq. (10) to develop an algorithm in the next section.

V. Formulation Analysis and Algorithm Development

In this section, we first expand and analyze the spatial-temporal graph-signal smoothness prior $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$. Based on the analysis, we simplify and reformulate the optimization in Eq. (10). Finally, we present an efficient algorithm to optimize the desired point cloud, the temporal edge weights and the spatial edge weights alternately.

A. Analysis of the Spatial-Temporal Smoothness Prior

The spatial-temporal smoothness prior $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$ in Eq. (10) enforces the structure of the triple-cube $\mathbf{g}$ to be smooth with respect to the spatial-temporal graph $\mathcal{G}_g$. We unfold it and analyze its function for simplification as follows.

As defined in IV-D, $\mathbf{g}$ consists of three cubes: $\hat{\mathbf{c}}_t^{f-1}, \mathbf{c}_r, \hat{\mathbf{c}}_t^{f+1}$, so the weighted adjacency matrix $\mathbf{W}_g$ of $\mathcal{G}_g$ can be written as:

$$\mathbf{W}_g = \begin{bmatrix} \mathbf{W}_{f-1,f-1} & \mathbf{W}_{f-1,f} & \mathbf{W}_{f-1,f+1} \\ \mathbf{W}_{f,f-1} & \mathbf{W}_{f,f} & \mathbf{W}_{f,f+1} \\ \mathbf{W}_{f+1,f-1} & \mathbf{W}_{f+1,f} & \mathbf{W}_{f+1,f+1} \end{bmatrix}, \tag{11}$$

where the submatrix $\mathbf{W}_{f-1,f-1}$ is composed of edge weights between points in $\hat{\mathbf{c}}_t^{f-1}$, $\mathbf{W}_{f-1,f}$ denotes edge weights between the points of $\hat{\mathbf{c}}_t^{f-1}$ and $\mathbf{c}_r$, and the other submatrices are composed similarly. Note that $\mathbf{W}_g$ is symmetric, so we have $\mathbf{W}_{f-1,f} = \mathbf{W}_{f,f-1}^\top$, $\mathbf{W}_{f-1,f+1} = \mathbf{W}_{f+1,f-1}^\top$ and $\mathbf{W}_{f,f+1} = \mathbf{W}_{f+1,f}^\top$.

As mentioned in Section III-A, the degree matrix $\mathbf{D}_g$ of $\mathcal{G}_g$ is a diagonal matrix. We rewrite it as:

$$\mathbf{D}_g = \begin{bmatrix} \mathbf{D}_{f-1} & 0 & 0 \\ 0 & \mathbf{D}_f & 0 \\ 0 & 0 & \mathbf{D}_{f+1} \end{bmatrix}. \tag{12}$$

Then we can write the graph Laplacian matrix $\mathcal{L}_g$ of $\mathcal{G}_g$ according to the definition as

$$\mathcal{L}_g = \begin{bmatrix} \mathbf{D}_{f-1} - \mathbf{W}_{f-1,f-1} & -\mathbf{W}_{f-1,f} & -\mathbf{W}_{f-1,f+1} \\ -\mathbf{W}_{f,f-1} & \mathbf{D}_f - \mathbf{W}_{f,f} & -\mathbf{W}_{f,f+1} \\ -\mathbf{W}_{f+1,f-1} & -\mathbf{W}_{f+1,f} & \mathbf{D}_{f+1} - \mathbf{W}_{f+1,f+1} \end{bmatrix}, \tag{13}$$

Therefore, the spatial-temporal smoothness term $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$ can be expanded as

$$\begin{aligned} \mathbf{g}^\top \mathcal{L}_g \mathbf{g} = & \begin{bmatrix} (\hat{\mathbf{c}}_t^{f-1})^\top & (\mathbf{c}_r)^\top & (\hat{\mathbf{c}}_t^{f+1})^\top \end{bmatrix} \mathcal{L}_g \begin{bmatrix} \hat{\mathbf{c}}_t^{f-1} \\ \mathbf{c}_r \\ \hat{\mathbf{c}}_t^{f+1} \end{bmatrix} \\ = & (\hat{\mathbf{c}}_t^{f-1})^\top (\mathbf{D}_{f-1} - \mathbf{W}_{f-1,f-1}) \hat{\mathbf{c}}_t^{f-1} \\ & - (\mathbf{c}_r)^\top \mathbf{W}_{f,f-1} \hat{\mathbf{c}}_t^{f-1} \\ & - (\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f-1} \hat{\mathbf{c}}_t^{f-1} \\ & - (\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f} \mathbf{c}_r \\ & + (\mathbf{c}_r)^\top (\mathbf{D}_f - \mathbf{W}_{f,f}) \mathbf{c}_r \\ & - (\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f} \mathbf{c}_r \\ & - (\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f+1} \hat{\mathbf{c}}_t^{f+1} \\ & - (\mathbf{c}_r)^\top \mathbf{W}_{f,f+1} \hat{\mathbf{c}}_t^{f+1} \\ & + (\hat{\mathbf{c}}_t^{f+1})^\top (\mathbf{D}_{f+1} - \mathbf{W}_{f+1,f+1}) \hat{\mathbf{c}}_t^{f+1}. \end{aligned} \tag{14}$$

In Eq. (14), we observe that some terms of $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$ are independent of the inpainting task, such as the 1st, 3rd, 7th and 9th term. These terms are only related to $\hat{\mathbf{c}}_t^{f-1}$ and $\hat{\mathbf{c}}_t^{f+1}$, but irrelevant to the unknown variable $\mathbf{c}_r$. So we can treat them as constants in the optimization objective. Besides, some pairs of terms are equivalent. For example, the 2nd and 4th terms are transposed to each other, so their trace are equal. It is the same for the 6th and 8th terms, which can be combined into one term. Hence, $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$ is simplified as

$$\begin{aligned} \mathbf{g}^\top \mathcal{L}_g \mathbf{g} = C + (\mathbf{c}_r)^\top \mathcal{L}' \mathbf{c}_r - 2(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f} \mathbf{c}_r \\ - 2(\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f} \mathbf{c}_r, \end{aligned} \tag{15}$$

where C is a constant. $\mathcal{L}' = \mathbf{D}_f - \mathbf{W}_{f,f}$ is a *generalized Laplacian*, because the degree $\mathbf{D}_f$ in Eq. (12) includes not only the sum of the edge weights in the current cube $\mathbf{c}_r$ but also the sum of the temporal edge weights with the previous and subsequent corresponding cubes. As generally defined in [52], a generalized Laplacian is a symmetric matrix with non-positive off-diagonal entries, which can be constructed as $\mathcal{L}' = \mathcal{L} + \mathbf{P}$, where $\mathcal{L}$ is the combinatorial graph Laplacian matrix and $\mathbf{P}$ is a diagonal matrix. Here $\mathbf{P}$ provides the additional degree contributed by temporal edge weights associated with $\hat{\mathbf{c}}_t^{f-1}$ and $\hat{\mathbf{c}}_t^{f+1}$. Hence, the second term in Eq. (15) is the graph-signal smoothness prior for the desired cube $\mathbf{c}_r$, which enforces its internal structure to be smooth when merging information from both intra- and inter-source cubes.

The third term of Eq. (15) is actually a dot product between $\hat{\mathbf{c}}_t^{f-1}$ and $\mathbf{W}_{f-1,f}\mathbf{c}_r$. As $\mathbf{W}_{f-1,f}$ contains temporal edge weights between corresponding cubes in the current frame and the previous frame, $\widetilde{\mathbf{c}}_r = \mathbf{W}_{f-1,f}\mathbf{c}_r$ can be interpreted as the diffusion from the desired cube $\mathbf{c}_r$ to its temporal correspondence in the previous frame, which makes the distribution of

the points in $\widetilde{\mathbf{c}}_r$ approach that in $\hat{\mathbf{c}}_t^{f-1}$. Therefore, when we minimize the third term of Eq. (15), the operation follows as

$$\min \quad -2\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \widetilde{\mathbf{c}}_r] = \max \quad 2\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \widetilde{\mathbf{c}}_r]. \tag{16}$$

Thus, the smaller the third term of Eq. (15) is, the larger the dot product between $\hat{\mathbf{c}}_t^{f-1}$ and $\widetilde{\mathbf{c}}_r$ is, and the more similar they are. Hence, this term enforces the structure of $\widetilde{\mathbf{c}}_r$ to be similar with $\hat{\mathbf{c}}_t^{f-1}$, thus ensuring the temporal consistency with $\hat{\mathbf{c}}_t^{f-1}$. The fourth term is a similar function, which ensures the temporal consistency with $\hat{\mathbf{c}}_t^{f+1}$.

Now we have simplified the spatial-temporal smoothness term $\mathbf{g}^\top \mathcal{L}_g \mathbf{g}$, and have analyzed the functionality of each simplified term. Next, we simplify and reformulate the optimization in Eq. (10).

B. Reformulation

According to Eq. (15), Eq. (10) is simplified and reformulated as

$$\begin{aligned}
\min_{\mathbf{c}_r, \mathbf{W}_{f-1,f}, \mathbf{W}_{f+1,f}, \mathcal{L}'} \quad & \|\overline{\mathbf{\Omega}}\mathbf{c}_r - \overline{\mathbf{\Omega}}\mathbf{c}_t\|_F^2 + \alpha\|\mathbf{\Omega}\mathbf{c}_r - \mathbf{\Omega}\hat{\mathbf{c}}_s\|_F^2 \\
& + \beta\mathrm{tr}[(\mathbf{c}_r)^\top \mathcal{L}'\mathbf{c}_r] \\
& - 2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f}\mathbf{c}_r] \\
& - 2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f}\mathbf{c}_r] \\
& + \gamma\|\mathbf{W}_{f-1,f} - \mathbf{W}_{f+1,f}\|_1. \\
\text{s.t.} \quad & \mathbf{0} < \mathbf{W}_{f-1,f} \cdot \mathbf{1} \leqslant \mathbf{1}, \\
& \mathbf{0} < \mathbf{W}_{f+1,f} \cdot \mathbf{1} \leqslant \mathbf{1}, \\
& \mathcal{L}'_{i,j} = \mathcal{L}_{j,i} \leqslant 0, i \neq j, \\
& \mathcal{L}' \cdot \mathbf{1} \geqslant \mathbf{0}, \\
& \mathrm{tr}(\mathcal{L}') = m(K+2),
\end{aligned} \tag{17}$$

where $\mathbf{1}$ is a vector with all the elements as 1.

Here we add some constraints for the variables. The first and second constraints enforce the sum of each row in $\mathbf{W}_{f-1,f}$ and $\mathbf{W}_{f+1,f}$ to be within the range $(0, 1]$ so as to constrain the sum of temporal weights for each point. As $\beta > 0$, this minimization problem will not lead to a pathological solution of $\mathbf{W}_{f-1,f}$ or $\mathbf{W}_{f+1,f}$ as a zero matrix.

The last three constraints are set for a valid generalized Laplacian $\mathcal{L}'$. According to the definition, the constraint $\mathcal{L}'_{i,j} = \mathcal{L}_{j,i} \leqslant 0$ enforces $\mathcal{L}'$ to be symmetric with non-positive off-diagonal entries (corresponding to non-negative edge weights in the graph). Also, the sum of each row in $\mathcal{L}'$ should be larger than 0 as a generalized Laplacian. Besides, as discussed in V-A , $\mathcal{L}'$ has extra degrees from its temporal connectivities as a generalized graph Laplacian. The degree of each point should be $K + 2$, since a point is expected to be connected to K neighbors in the spatial domain and 2 neighbors in the previous and subsequent frames in the temporal domain. m denotes the number of the points in $\mathbf{c}_r$, thus the trace of $\mathcal{L}'$ should be $m(K + 2)$, which prevents a pathological solution of $\mathcal{L}'$—a zero matrix that corresponds to a fully disconnected graph.

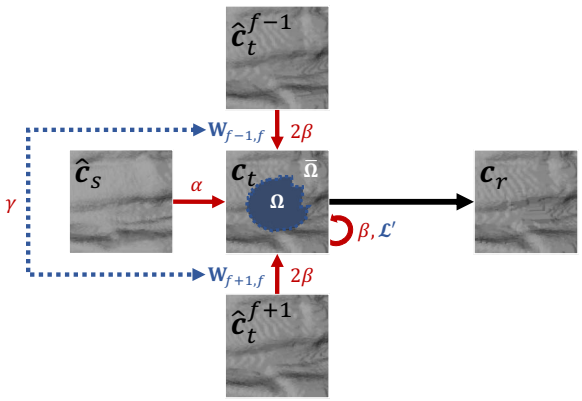


Fig. 5. The schematic figure for the reformulation.

C. The Proposed Algorithm

As there are multiple variables in the formulation, we optimize each variable alternately as follows.

1) The optimization of $\mathbf{c}_r$: We first initialize the spatial-temporal graph encoded in $\mathbf{W}_{f-1,f}$, $\mathbf{W}_{f+1,f}$ and $\mathcal{L}'$ as introduced in Section IV-D, which leads to the following optimization for $\mathbf{c}_r$:

$$\begin{aligned}
\min_{\mathbf{c}_r} \quad & \|\overline{\mathbf{\Omega}}\mathbf{c}_r - \overline{\mathbf{\Omega}}\mathbf{c}_t\|_F^2 + \alpha\|\mathbf{\Omega}\mathbf{c}_r - \mathbf{\Omega}\hat{\mathbf{c}}_s\|_F^2 + \beta\mathrm{tr}[(\mathbf{c}_r)^\top \mathcal{L}'\mathbf{c}_r] \\
& - 2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f}\mathbf{c}_r] - 2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f}\mathbf{c}_r].
\end{aligned} \tag{18}$$

This is a quadratic programming problem. Taking derivative of Eq. (18) with respect to $\mathbf{c}_r$ and setting the derivative to 0, we have the closed-form solution of $\mathbf{c}_r$:

$$\begin{aligned}
\mathbf{c}_r^{\text{opt}} = & (\overline{\mathbf{\Omega}}^2 + \alpha\mathbf{\Omega}^2 + \beta\mathcal{L}')^{-1} \\
& (\overline{\mathbf{\Omega}}^2\mathbf{c}_t + \alpha\mathbf{\Omega}^2\hat{\mathbf{c}}_s + \beta\mathbf{W}_{f,f-1}\hat{\mathbf{c}}_t^{f-1} + \beta\mathbf{W}_{f,f+1}\hat{\mathbf{c}}_t^{f+1}).
\end{aligned} \tag{19}$$

2) The optimization of $\mathcal{L}'$: With the initialized $\mathbf{W}_{f-1,f}$, $\mathbf{W}_{f+1,f}$ and the updated $\mathbf{c}_r$, we optimize the generalized graph Laplacian $\mathcal{L}'$:

$$\begin{aligned}
\min_{\mathcal{L}'} \quad & \mathrm{tr}[(\mathbf{c}_r)^\top \mathcal{L}'\mathbf{c}_r], \\
\text{s.t.} \quad & \mathcal{L}_{i,j} = \mathcal{L}_{j,i} \leqslant 0, i \neq j, \\
& \mathcal{L}' \cdot \mathbf{1} \geqslant \mathbf{0}, \\
& \mathrm{tr}(\mathcal{L}') = m(K+2).
\end{aligned} \tag{20}$$

This is a convex optimization problem, which can be efficiently solved via off-the-shelf tools such as the convex optimization toolbox of MATLAB [53].

3) The optimization of $\mathbf{W}_{f-1,f}$ and $\mathbf{W}_{f+1,f}$: With the updated $\mathbf{c}_r$ and $\mathcal{L}'$, we then further optimize the temporal edge weights in $\mathbf{W}_{f-1,f}$ and $\mathbf{W}_{f+1,f}$. When we fix $\mathbf{W}_{f+1,f}$ and optimize $\mathbf{W}_{f-1,f}$, the formulation follows as

$$\begin{aligned}
\min_{\mathbf{W}_{f-1,f}} \quad & -2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f}\mathbf{c}_r] + \gamma\|\mathbf{W}_{f-1,f} - \mathbf{W}_{f+1,f}\|_1, \\
\text{s.t.} \quad & \mathbf{0} < \mathbf{W}_{f-1,f} \cdot \mathbf{1} \leqslant \mathbf{1}.
\end{aligned} \tag{21}$$

Similarly, with the updated $\mathbf{W}_{f-1,f}$, we can optimize $\mathbf{W}_{f+1,f}$ via

$$\begin{aligned}
\min_{\mathbf{W}_{f+1,f}} \quad & -2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f+1})^\top \mathbf{W}_{f+1,f}\mathbf{c}_r] + \gamma\|\mathbf{W}_{f-1,f} - \mathbf{W}_{f+1,f}\|_1, \\
\text{s.t.} \quad & \mathbf{0} \leqslant \mathbf{W}_{f+1,f} \cdot \mathbf{1} \leqslant \mathbf{1}.
\end{aligned} \tag{22}$$

Both Eq. (21) and Eq. (22) consist of a convex and differentiable term as well as an l_1-norm. As the Orthant-Wise Limited-memory Quasi-Newton (OWL-QN) method has been demonstrated to be very effective in solving such l_1-regularized optimization problem [54], we employ OWL-QN to our formulation with a matrix variable, in which the quadrant of the objective function is limited to guarantee the continuity and differentiability of the l_1-norm. Specifically, taking the objective function $F(\mathbf{W}_{f,f-1}) = -2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f}\mathbf{c}_r] + \gamma\|\mathbf{W}_{f-1,f} - \mathbf{W}_{f+1,f}\|_1$ in Eq. (21) as an example, the Newton iterative formula is defined as:

$$\mathbf{W}_{f,f-1}^{i+1} = \mathbf{W}_{f,f-1}^{i} - \frac{F(\mathbf{W}_{f,f-1}^{i})}{F'(\mathbf{W}_{f,f-1}^{i})}, \tag{23}$$

where $\mathbf{W}_{f,f-1}^{i}$ and $\mathbf{W}_{f,f-1}^{i+1}$ are $\mathbf{W}_{f,f-1}$ in the i-th and $(i+1)$-th iteration, respectively. $F'(\mathbf{W}_{f,f-1}^{i})$ is the first derivative of the objective function with respect to $\mathbf{W}_{f,f-1}$ in the i-th iteration.

$F(\mathbf{W}_{f,f-1})$ is not continuously differentiable, so we limit the quadrant of $\mathbf{W}_{f,f-1}$ to compute the *pseudo-gradient* as $F'(\mathbf{W}_{f,f-1}^{i})$ in the iterations. Firstly, for each $w_{k,l}$ in $\mathbf{W}_{f,f-1}$, we constrain the iteration direction by

$$w_{k,l}^{i+1} = \begin{cases} w_{k,l}^{i+1}, & \sigma(w_{k,l}^{i+1}) = \sigma(w_{k,l}^{i}) \\ 0, & \text{otherwise} \end{cases}, \tag{24}$$

where the sign function σ takes values in $\{-1, 0, 1\}$ according to whether a real value is negative, zero, or positive. Then we define the pseudo-gradient of $F(\mathbf{W}_{f,f-1})$ at $\mathbf{W}_{f,f-1}^{i}$ as in OWL-QN [54]:

$$F'(\mathbf{W}_{f,f-1}^{i}) = \begin{cases} \partial^- F(\mathbf{W}_{f,f-1}^{i}), & \partial^- F(\mathbf{W}_{f,f-1}^{i}) > 0, \\ \partial^+ F(\mathbf{W}_{f,f-1}^{i}), & \partial^+ F(\mathbf{W}_{f,f-1}^{i}) < 0 \\ \infty, & \text{otherwise} \end{cases}, \tag{25}$$

where $\partial^- F(\mathbf{W}_{f,f-1}^{i})$ and $\partial^+ F(\mathbf{W}_{f,f-1}^{i})$ are the left and right partial derivatives of $F(\mathbf{W}_{f,f-1})$ respectively, which are given by

$$\partial^{\pm} F(\mathbf{W}_{f,f-1}^{i}) = \frac{\partial\, l(\mathbf{W}_{f,f-1}^{i})}{\partial \mathbf{W}_{f,f-1}^{i}} + \partial^{\pm}\sigma(\mathbf{W}_{f,f-1}^{i}), \tag{26}$$

where $l(\mathbf{W}_{f,f-1}^{i}) = -2\beta\mathrm{tr}[(\hat{\mathbf{c}}_t^{f-1})^\top \mathbf{W}_{f-1,f}\mathbf{c}_r]$ is the linear term of the objective function in Eq. (21), and $\partial^{\pm}\sigma(\mathbf{W}_{f,f-1}^{i})$ is the left and right partial derivatives of the l_1-norm of the objective function in Eq. (21). Each element $\partial^{\pm}\sigma(w_{k,l})$ in $\partial^{\pm}\sigma(\mathbf{W}_{f,f-1}^{i})$ is a function of $w_{k,l}$:

$$\partial^{\pm}\sigma(w_{k,l}) = \begin{cases} \gamma\sigma(w_{k,l}), & w_{k,l} \neq 0 \\ \pm\gamma, & \text{otherwise} \end{cases}. \tag{27}$$

Thus we can solve Eq. (21) by the enhanced Newton-Raphson method via pseudo-gradient. Eq. (22) is solved in the same way.

Hence, Eq. (17) is solved optimally by iteratively solving each variable. Finally, we replace the target cube with the resulting cube in the target frame $\mathbf{P}_f$, which serves as the output. After each hole is inpainted, the candidate cubes and the target frame will be updated, which will be considered in the inpainting of the subsequent holes and subsequent frames. This provides more opportunities to find more similar cubes for the subsequent holes as well as more accurate temporal correspondence for the subsequent frames, which is thus beneficial to performance improvement.

VI. Experimental Results

A. Experimental Setup

We evaluate the proposed method by testing on several 3D dynamic point cloud datasets from MPEG [2], [56] and Microsoft [57], including *Longdress, Loot, Maria, Phili, Queen, Redandblack, Sarah, Skiing, Soldier*, and *UlliWegner*, each with 40 frames. We test on two types of holes: 1) real holes generated during the capturing process, which have no ground truth; 2) synthetic holes on point clouds so as to compare with the ground truth. In particular, the number of nearest neighbors K is considered to be related to m, the number of existing points in the cube. Empirically, $K = \sqrt{m}$ in our experiments. Besides, the size H of the searching box in Section IV-C is considered to be related to M, the size of the unit cube. Empirically, $H = 2.5M$ in our experiments.

Further, we compare our method with 1) three competing algorithms for static 3D geometry inpainting, including Meshlab [55], Lozes *et al.* [13] and Hu *et al.* [17]. To test on dynamic point clouds, we perform them frame by frame independently. As Meshlab is based on meshes, we convert

TABLE I
Performance Comparison in GPSNR (dB)

	Meshlab [55]	Lozes [13]	Hu [17]	Fu [19]	Spatial-ours	Proposed
Longdress	11.7926	30.3883	41.5686	43.1301	43.8497	**44.5519**
Loot	16.4451	27.3715	40.1546	47.5648	48.0212	**48.7913**
Maria	12.2943	28.4910	34.9747	40.5228	40.9791	**41.1820**
Phili	9.9593	19.4861	35.4972	38.8930	40.4547	**41.6200**
Queen	14.0248	22.4650	29.6086	37.0174	38.4851	**39.1679**
Redandblack	13.1772	24.4810	33.9921	39.0103	40.4571	**41.8005**
Sarah	14.5581	26.4108	30.2719	34.9186	38.1165	**39.4186**
Skiing	20.1247	29.9233	36.0379	42.4754	44.4520	**45.9341**
Soldier	17.4697	23.1571	34.5062	42.2980	43.4823	**44.9148**
UlliWegner	24.9424	31.5455	41.3037	45.8411	46.3179	**48.1035**

TABLE II
PERFORMANCE COMPARISON IN NSHD ($\times 10^{-7}$)

	Meshlab [55]	Lozes [13]	Hu [17]	Fu [19]	Spatial-ours	Proposed
Longdress	24.8631	7.1362	2.9174	0.9131	1.0962	**0.8450**
Loot	14.1925	8.9410	3.1102	**0.3549**	1.5650	0.7944
Maria	21.3481	11.0980	4.2009	2.5187	2.6485	**2.0812**
Phili	25.3014	16.4119	7.9063	3.4920	2.7526	**2.1810**
Queen	23.1001	13.2545	6.0290	2.8429	2.6613	**1.9334**
Redandblack	22.8300	9.6233	5.9856	1.9324	1.8681	**1.0352**
Sarah	17.3732	11.4299	2.0854	1.2461	1.0590	**0.9941**
Skiing	19.5847	10.6713	3.5294	1.9145	1.6002	**1.3903**
Soldier	17.1376	10.0044	5.2145	1.2057	1.1623	**1.0851**
UlliWegner	11.2509	6.7370	1.9658	0.6362	0.6588	**0.5696**

point clouds to meshes via the Meshlab software [55] prior to testing the method, and then convert the inpainted meshes back to point clouds as the final output; 2) our previous work on dynamic point cloud inpainting [19], which is our baseline method since we extend it by learning the spatial correlation as well as the temporal correlation via the second-order inter-frame coherence as mentioned in Section II; 3) an ablation study—"Spatial-ours", where we consider only the spatial graph learning of $\mathcal{L}'$ while removing the temporal graph learning so as to evaluate its effectiveness.

B. Results on Point Cloud Inpainting

Objective results. It is nontrivial to measure the geometric difference of point clouds objectively. We apply the geometric distortion metrics in [58] and [8], referred to as GPSNR and NSHD respectively. NSHD is a point-to-point distance metric that directly measures the original error between points in the reconstructed point cloud and the ground truth, while GPSNR is a point-to-plane distance metric that measures projected error vectors along normal directions for surface structural description. The higher GPSNR is and the lower NSHD is, the smaller the difference between two point clouds is.

Table I and Table II present the average objective results of the frames for each sequence with synthetic holes in GPSNR and NSHD respectively. We see that our scheme outperforms all the competing methods in GPSNR and NSHD significantly. Specifically, in comparison with methods for static point cloud inpainting, we achieve 28.07 dB gain in GPSNR on average over Meshlab, 17.18 dB over [13], and 7.76 dB over [17], as well as much lower NSHD than the other methods. Note that, the NSHD measurement of the proposed method on *Loot* is a bit higher than that of [19], because of the limitation of the point-to-point NSHD metric and the sparsity of the point cloud *Loot*. Nevertheless, our performance improves in terms of the GPSNR metric, which captures the geometric structure better.

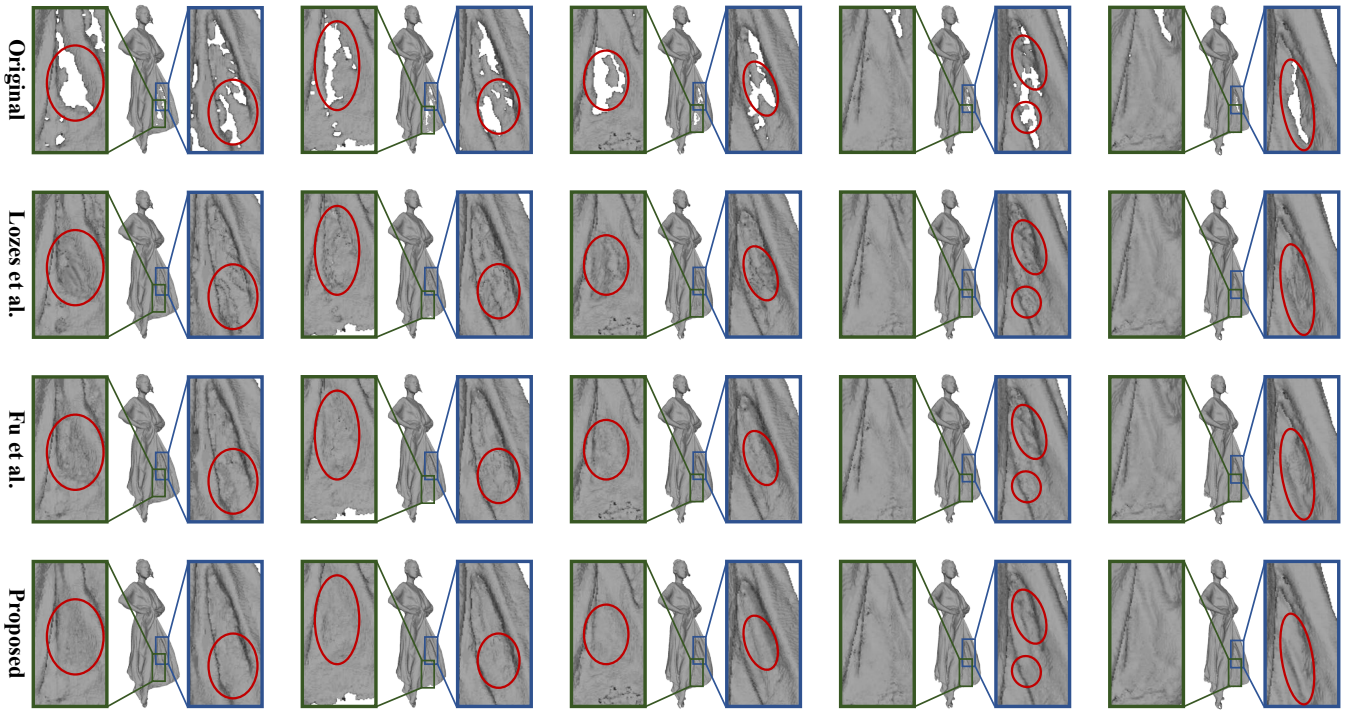


Fig. 6. Some consecutive frames of the inpainting results from different methods for *Longdress* with the real holes magnified.

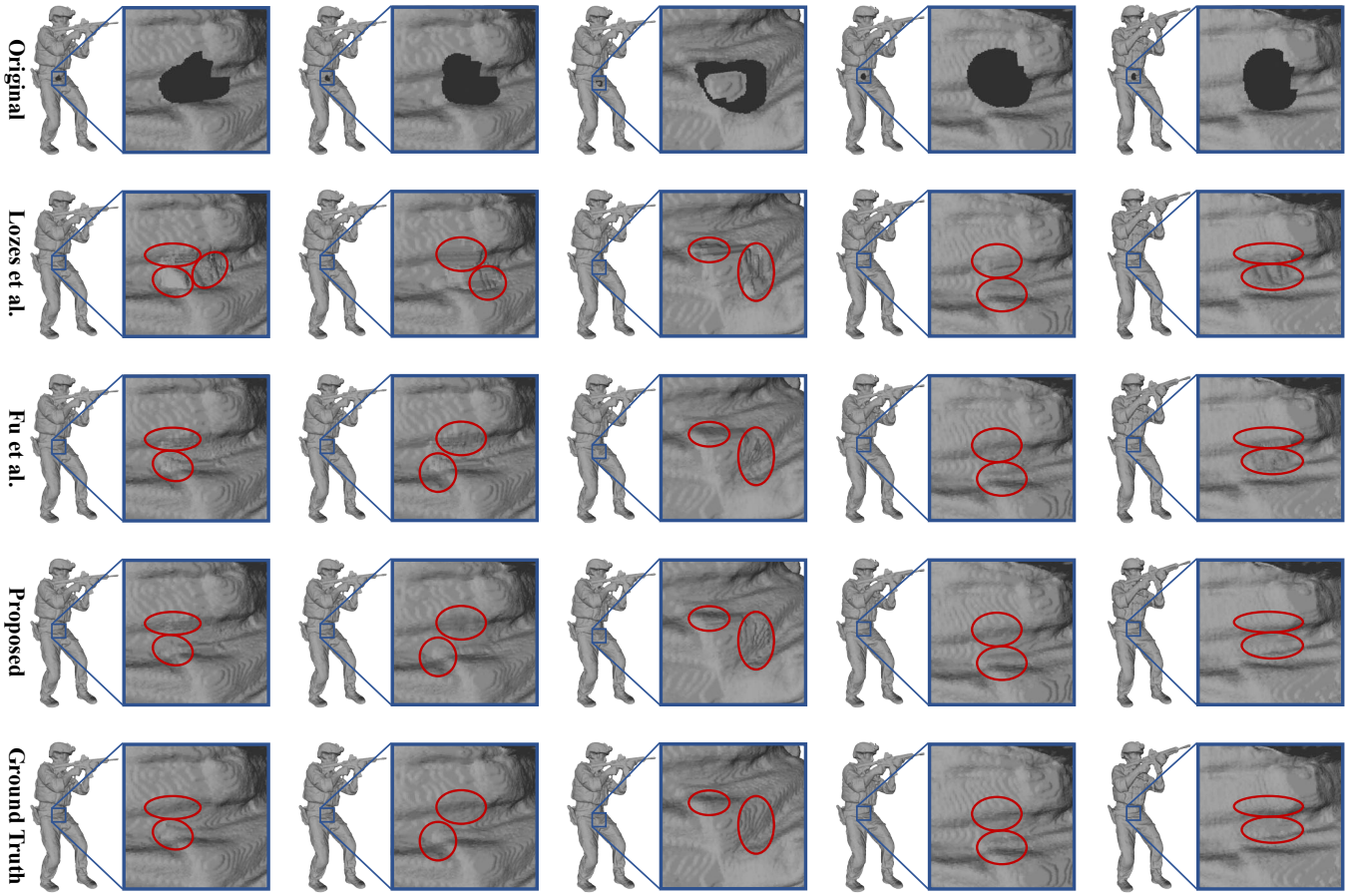


Fig. 7. Some frames of the inpainting results from different methods for *Soldier* with the synthetic holes magnified.

In comparison with our previous method [19] for dynamic point cloud inpainting *without* spatial-temporal graph learning, we achieve 2.38 dB gain in GPSNR on average and reduce 0.41×10^{-7} in NSHD on average, thus validating the importance of the spatial-temporal graph learning. Further, when compared to *Spatial-ours* with the temporal graph learning removed, *i.e.*, when we only learn the spatial graph and keep the initialized temporal graph fixed, we achieve 1.09 dB gain in GPSNR on average and reduce 0.42×10^{-7} in NSHD on average. This validates the effectiveness of the temporal graph learning for dynamic point cloud inpainting.

Subjective results. Further, Fig. 6 and Fig. 7 demonstrate the subjective inpainting results for real holes and synthetic holes respectively. Due to the page limit, we show several representative frames compared with Lozes' method [13] and our previous method [19]. For the real holes in consecutive frames, as shown in the first row of Fig. 6, which are fragmentary, the results of [13] in the second row show artificial contours of the original holes, since it attempts to connect the boundary of the hole region with planar structures without smoothing. Also, their inpainted results are not consistent among consecutive frames. The results of our previous method [19] in the third row are not very coherent among consecutive frames, because only the first-order temporal coherence is considered and the spatial-temporal graph is empirically constructed. In comparison, our results shown in the last row of Fig. 6 demonstrate that the proposed method is able to inpaint holes with reasonable geometry structure and smoothness over the hole region. Besides, since we leverage the second-order inter-frame coherence, our inpainted regions show good consistency among consecutive frames.

In Fig. 7, we synthesize holes in the point cloud sequence *Soldier*, with more complex and larger holes than the real holes in Fig. 6. We observe that [13] covers the missing area with curt ribbed geometry, which introduces wrong geometry around the holes compared to the ground truth. Also, the contents look incoherent among the consecutive frames. The results of [19] are a little bumpy, and exhibit some geometric distortion as in the red circles. In comparison, our results shown in the fourth row of Fig. 7 are almost the same as the ground truth, and exhibit coherence among neighboring frames. This gives credits to the spatial-temporal graph learning and the second-order temporal coherence.

Analysis. Further, we discuss and analyze the robustness of the proposed method to the complexity of holes. The visual results in Fig. 6 and Fig. 7 demonstrate holes with high frequencies in both the spatial domain and the temporal domain. We see that, our results reconstruct the complex geometric structure well with consistency in the temporal domain, while the results from comparison methods exhibit blurred geometric details or artifacts. This shows that the proposed method is able to address challenging holes with

high frequencies and is insensitive to the placement of the inpainting volume. This is because we exploit the second-order inter-frame coherence and the intra-frame self-similarity along with spatial-temporal graph learning, leading to abundant references in both the temporal and spatial domain for inpainting. Further, the graph-signal smoothness prior in the formulation enforces the signal to adapt to the topology of the graph. If the signal contains high frequencies, *e.g.*, with a sharp boundary inside, such high-frequency structure will be preserved with the graph-signal smoothness regularization, as it is encoded in the graph topology learned from self-similar regions in the spatial domain or the corresponding regions in the temporal domain.

VII. Conclusion

We propose to address 3D dynamic point cloud inpainting via spatial-temporal graph learning, exploiting the second-order inter-frame coherence and the intra-frame self-similarity. The key idea is to characterize the consistent motion flow among corresponding local cubes in consecutive frames—the second-order inter-frame coherence, where the temporal correspondence is searched based on the point-to-plane distance for structural description. We formulate dynamic point cloud inpainting as the joint optimization of the desired complete point cloud and the underlying spatial-temporal graph, regularized by the difference in temporal edge weights of the underlying spatial-temporal graph (the second-order inter-frame coherence) and smoothness in the spatial graph (the intra-frame self-similarity). We further provide analysis and reformulation of the optimization problem, and present an efficient algorithm. Experimental results demonstrate the superiority of the proposed method on both synthetic and real holes. Future works include the extension to inpainting the color attribute of dynamic point clouds.

References

[1] C. Tulvan, R. Mekuria, and Z. Li, "Use cases for point cloud compression (pcc)," in *ISO/IEC JTC1/SC29/WG11 (MPEG) output document N16331*, June 2016.

[2] T. Ebner, I. Feldmann, O. Schreer, P. Kauff, and T. Unger, "Hhi point cloud dataset of a boxing trainer," in *ISO/IEC JTC1/SC29/WG11 (MPEG2018) input document M42921*, July 2018.

[3] P. Chalmovianský and B. Jüttler, "Filling holes in point clouds," in *Mathematics of Surfaces*. Springer Berlin Heidelberg, 2003, pp. 196–212.

[4] P. Sahay and A. N. Rajagopalan, "Harnessing self-similarity for reconstruction of large missing regions in 3D models," in *International Conference on Pattern Recognition*, 2012, pp. 101–104.

[5] ——, "Geometric inpainting of 3D structures," in *Computer Vision and Pattern Recognition Workshops*, 2015, pp. 1–7.

[6] S. Shankar, S. A. Ganihar, and U. Mudenagudi, "Framework for 3D object hole filling," in *Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics*, 2015, pp. 1–4.

[7] S. Shankar and U. Mudenagudi, "Example-based 3d inpainting of point clouds using metric tensor and christoffel symbols," *Machine Vision and Applications*, vol. 29, pp. 329–343, 2018.

[8] C. Dinesh, I. V. Bajic, and G. Cheung, "Exemplar-based framework for 3D point cloud hole filling," in *IEEE International Conference on Visual Communications and Image Processing*, May 2017.

[9] ——, "Adaptive nonrigid inpainting of three-dimensional point cloud geometry," *IEEE Signal Processing Letters*, vol. 25, no. 6, pp. 878–882, 2018.

[10] J. Wang, M. M. Oliveira, M. Garr, and M. Levoy, "Filling holes on locally smooth surfaces reconstructed from point clouds," *Image & Vision Computing*, vol. 25, no. 1, pp. 103–113, 2007.

[11] Y. Quinsat and C. Lartigue, "Filling holes in digitized point cloud using a morphing-based approach to preserve volume characteristics," *International Journal of Advanced Manufacturing Technology*, vol. 81, no. 1-4, pp. 411–421, 2015.

[12] F. Lozes, A. Elmoataz, and O. Lézoray, "Partial difference operators on weighted graphs for image processing on surfaces and point clouds," *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3896–909, 2014.

[13] ——, "PDE-based graph signal processing for 3-D color point clouds : opportunities for cultural heritage," *IEEE Signal Processing Magazine*, vol. 32, no. 4, pp. 103–111, 2015.

[14] H. Lin and W. Wang, "Feature preserving holes filling of scattered point cloud based on tensor voting," in *IEEE International Conference on Signal and Image Processing*, 2017, pp. 402–406.

[15] Y. Muraki, K. Nishio, and T. Kanaya, "An automatic hole filling method of point cloud for 3d scanning," 2017.

[16] Z. Fu, W. Hu, and Z. Guo, "Point cloud inpainting on graphs from non-local self-similarity," in *IEEE International Conference on Image Processing*, Athens, Greece, 2018.

[17] W. Hu, Z. Fu, and Z. Guo, "Local frequency interpretation and non-local self-similarity on graph for point cloud inpainting," *IEEE Transactions on Image Processing*, vol. 28, no. 8, pp. 4087–4100, 2019.

[18] J. He, Z. Fu, W. Hu, and Z. Guo, "Point cloud attribute inpainting in graph spectral domain," in *IEEE International Conference on Image Processing*, 2019.

[19] Z. Fu, W. Hu, and Z. Guo, "3D dynamic point cloud inpainting via temporal consistency on graphs," in *IEEE International Conference on Multimedia & Expo*, London, United Kingdom, 2020.

[20] J. Pang and G. Cheung, "Graph Laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Transactions on Image Processing*, vol. 26, no. 6, pp. 1770–1785, 2017.

[21] S. Friedman and I. Stamos, "Online facade reconstruction from dominant frequencies in structured point clouds," in *Computer Vision and Pattern Recognition Workshops*, 2012, pp. 1–8.

[22] X. Wu and W. Chen, "A scattered point set hole-filling method based on boundary extension and convergence," in *Intelligent Control and Automation*, 2015, pp. 5329–5334.

[23] C. Chen and B. Yang, "Dynamic occlusion detection and inpainting of in situ captured terrestrial laser scanning point clouds sequence," *Isprs Journal of Photogrammetry Remote Sensing*, vol. 119, pp. 90–107, 2016.

[24] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representation perspective," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 44–63, 2019.

[25] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Processing Magazine*, vol. 36, no. 3, pp. 16–43, 2019.

[26] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2841–2856, 2020.

[27] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 205–213.

[28] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2826–2830.

[29] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning with constraints on graph temporal variation," *https://arxiv.org/abs/2001.03346*, 2020.

[30] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," *https://arxiv.org/abs/1801.07455*, 2018.

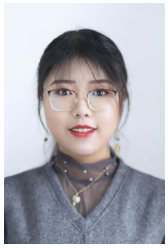
[31] C. C. de Amorim, D. Macêdo, A. G. Marques, and C. Zanchettin, "Spatial-temporal graph convolutional networks for sign language recognition," in *International Conference on Artificial Neural Networks*, 2019, pp. 646–657.

[32] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, 2019, pp. 922–929.

[33] X. Gao, W. Hu, J. Tang, J. Liu, and Z. Guo, "Optimized skeleton-based action recognition via sparsified graph regression," in *Proceedings of the 27th ACM International Conference on Multimedia*, 2019, pp. 601–610.

[34] C. Song, Y. Lin, S. Guo, N. Feng, and H. Wan, "Spatial-temporal synchronous graph convolutional networks: A new framework for

spatial-temporal network data forecasting," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 1, 2020, pp. 914–921.
[35] B. Baingana and G. B. Giannakis, "Tracking switched dynamic network topologies from information cascades," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 985–997, 2016.
[36] F. K. Chung, "Spectral graph theory," vol. 92, no. 6, p. 212, 1996.
[37] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.
[38] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.
[39] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 771–773, 2017.
[40] G. Shen, W.-S. Kim, S. K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, Nagoya, Japan, December 2010, pp. 566–569.
[41] W. Hu, G. Cheung, X. Li, and O. C. Au, "Depth map compression using multi-resolution graph-based transform for depth-image-based rendering," in *IEEE International Conference on Image Processing*, Orlando, FL, September 2012, pp. 1297–1300.
[42] W. Hu, G. Cheung, A. Ortega, and O. C. Au, "Multi-resolution graph fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, vol. 24, no. 1, January 2015, pp. 419–33.
[43] W. Hu, J. Pang, X. Liu, D. Tian, C. Lin, and A. Vetro, "Graph signal processing for geometric data and beyond: Theory and applications," *https://arxiv.org/abs/2008.01918*, 2020.
[44] D. A. Spielman, "Lecture 2 of spectral graph theory and its applications," September 2004.
[45] P. J. Besl and N. D. Mckay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 2002.
[46] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, "The trimmed iterative closest point algorithm," in *International Conference on Pattern Recognition, 2002. Proceedings*, vol. 3, 2002, pp. 545–548.
[47] K. Julius, B. Nico, R. R. Bogdan, B. Michael, S. Eckehard, and G. Suat, "Real-time compression of point cloud streams," in *IEEE International Conference on Robotics & Automation*, 2012.
[48] D. Thanou, P. A. Chou, and P. Frossard, "Graph-based compression of dynamic 3d point cloud sequences," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1765–1778, 2016.
[49] L. Li, Z. Li, S. Liu, and H. Li, "Efficient projected frame padding for video-based point cloud compression," *IEEE Transactions on Multimedia*, pp. 1–1, 2020.
[50] A. Javaheri, C. Brites, S. Guo, F. Pereira, and J. Ascenso, "Subjective and objective quality evaluation of 3d point cloud denoising algorithms," in *IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, 2017.
[51] J. Zeng, G. Cheung, M. Ng, J. Pang, and Y. Cheng, "3D point cloud denoising using graph Laplacian regularization of a low dimensional manifold model," *IEEE Transactions on Image Processing*, vol. 29, pp. 3474–3489, December 2019.
[52] T. Biyikoğu, J. Leydold, and P. F. Stadler, "Laplacian eigenvectors of graphs," *Lecture Notes in Mathematics*, vol. 1915, 2007.
[53] M. Grant and S. Boyd, "Cvx users' guide," *http://www.stanford.edu/ boyd/cvx*, 2011.
[54] G. Andrew and J. Gao, "Scalable training of l1-regularized log-linear models," in *Proceedings of the 24th International Conference on Machine Learning*, no. 8, 2007, pp. 33–40.
[55] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool," in *Eurographics Italian Chapter Conference*, S. Vittorio, D. C. Rosario, and F. Ugo, Eds. The Eurographics Association, 2008.
[56] E. d'Eon, B. Harrison, T. Myers, and P. A. Chou, "8i voxelized full bodies, version 2 - a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29/WG11 (MPEG2017) input document M40059*, January 2017.
[57] Q. Cai and P. A. Chou, "Microsoft voxelized upper bodies a voxelized point cloud dataset," in *ISO/IEC JTC1/SC29 WG11 ISO/IEC JTC1/SC29/WG1 input document m38673/M72012*, May 2016.
[58] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *IEEE International Conference on Image Processing*, Beijing, China, September 2017.



Zeqing Fu (Student Member, IEEE) received the B.S. degree in computer science from Beijing Normal University, Beijing, China, in 2018. She is currently pursuing the Master degree in Wangxuan Institute of Computer Technology, Peking University. Her current research interests include graph signal processing, 3D data representation and processing. She has published in top journals and applied for several patents. She is the first author of the Best Student Paper Runner Up Award in IEEE ICME 2020.

Wei Hu (Member, IEEE) received the B.S. degree in Electrical Engineering from the University of Science and Technology of China in 2010, and the Ph.D. degree in Electronic and Computer Engineering from the Hong Kong University of Science and Technology in 2015. She was a Researcher with Technicolor, Rennes, France, from 2015 to 2017. She is currently an Assistant Professor and Peking University Boya Young Fellow with Wangxuan Institute of Computer Technology, Peking University. Her research interests include graph signal processing, graph-based machine learning and 3D visual computing. She has authored over 40 international journal and conference publications, including top IEEE journal publications. She served as an area chair of ACM MM 2020 and ICME 2020, and will serve as an Open Source Chair for ICME 2021. She is the recipient of the Best Student Paper Runner Up Award in IEEE ICME 2020. She is a TC member of IEEE MSA, and has served as a regular reviewer for IEEE Trans. on Image Processing, IEEE Trans. on Signal Processing, IEEE Trans. on Circuits and Systems for Video Technology, etc.