Morphological PDEs on Graphs for Filtering and Inpainting of Point Clouds

François Lozes, Abderrahim Elmoataz, Olivier Lézoray
Normandie Univ., UNICAEN, ENSICAEN, GREYC UMR CNRS 6072, Caen, France
{francois.lozes,abderrahim.elmoataz-billah,olivier.lezoray}@unicaen.fr

***Abstract*—In this paper, we propose an adaptation of morphological Partial Differential Equations (PDEs) on graphs using the framework of Partial difference Equations (PdEs). This enables to define adaptive morphological operators on graphs. We then show how these operators can be used for interpolation and filtering of raw point clouds. To enable a patch-based processing of point clouds, we also show how a weighted graph based on patches can be associated with a point cloud. Finally, we present applications in cultural heritage[1].**

I. Introduction

The recent proliferation of commercial three-dimensional digital scanning devices has enabled to establish 3D scanning as a practical reality in the field of cultural heritage preservation [SBG11]. The interest in 3D scanning for cultural heritage is evident: it provides a high precision digital reference of a cultural object, and makes possible its easy mass distribution and consultation. The cyark project (www.cyark.org) is such an example that aims at digitally sharing and preserving the world's cultural heritage.

3D scanners create point clouds or meshes. Point clouds consist of sets of unorganized and non-oriented points given by their x, y, z coordinates. Additionally, a color can be associated with each point. On the contrary, a mesh is a collection of vertices, edges and faces that defines the shape of a polyhedral object. The faces usually consist of triangles and some texture can be associated with these faces for visualization purposes. The study of polygon meshes has a long history in the fields of computer graphics and geometric modeling [BKP$^+$10].

In fact, 3D scanners first acquire 3D data as a raw point cloud and the latter is post-processed to be converted in a (usually) triangular mesh. Indeed, triangular meshes can be very efficiently visualized on most 3D video cards. However, with the advent of 3D scanners that have a very low acquisition error, there is a growing interest in the processing of raw point clouds. The goal is different from having the data in the form of triangular meshes: one wants to keep an optimal accuracy that enables to put in evidence all the defects of the scanning process as well as recovering all textures and details. This interest in the disposal of raw point clouds goes beyond 3D scanners, for example LIDAR (Light Detection and Ranging) scanners also generate raw point clouds that are for interest in cultural heritage.

All these elements have led to the developments of techniques that enable to directly process a raw point cloud [Dig12] (see e.g., the Point Cloud library at www.pointclouds.org). Our work is in line with these recent developments and aims at proposing a general formalism for the morphological processing of point clouds with the adaptation of PDEs on Graphs.

In this paper, we present a way to restore 3D raw point clouds using the framework of PdEs to perform graph-based morphological processing [ELB08]. All the presented PdEs in this paper are based on morphological operators. Unlike level sets or parametric surfaces, with weighted graphs we work directly with raw discrete data. Here, the handled data are associated with each point of a point cloud: coordinates or colors.

The paper is organized as follows. In the first section, we present PdEs on graphs [ELB08], [EDL12] and derive morphological operators that are used to filter or inpaint point clouds. Second section presents the construction of a patch-based weighted graph from a point cloud. Last section presents examples of filtering and inpainting on real 3D scanned objects.

II. Weighted Graphs

In this section, we recall definitions and operators on graphs. This constitutes the basis of the framework of PdEs on a graph [ELB08] that enables to transpose PDEs on graphs.

A. Notations and Preliminaries

A weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ consists of a finite set $\mathcal{V} = \{v_1, \ldots, v_N\}$ of N vertices and a finite set $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ of weighted edges. We assume $\mathcal{G}$ to be undirected, with no self-loops and no multiple edges. Let (v_i, v_j) be the edge of $\mathcal{E}$ that connects two vertices v_i and v_j of $\mathcal{V}$. Its weight, denoted by $w(v_i, v_j)$, represents the similarity between its vertices. Similarities are usually computed by using a positive symmetric function $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ satisfying $w(v_i, v_j) = 0$ if $(v_i, v_j) \notin \mathcal{E}$. The notation $v_i \sim v_j$ is also used to denote two adjacent vertices. The degree of a vertex v_i is defined as $\delta_w(v_i) = \sum_{v_j \sim v_i} w(v_i, v_j)$. Let $\mathcal{H}(\mathcal{V})$ be the Hilbert space of real-valued functions defined on the vertices of a graph. A function $f : \mathcal{V} \to \mathbb{R}$ of $\mathcal{H}(\mathcal{V})$ assigns a real value $f(v_i)$ to each vertex $v_i \in \mathcal{V}$. $\mathcal{H}(\mathcal{V})$ space is endowed with the usual inner products.

B. Difference Operators on Weighted Graphs

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be a weighted graph, $f : \mathcal{V} \to \mathbb{R}$ be a function of $\mathcal{H}(\mathcal{V})$ and $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$, a weight function

[1]This work was funded under a Ph.D. grant of the regional council of Lower-Normandy.

that depends on the interactions between the vertices. The *difference operator* [ELB08] of f, noted $d_w : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{E})$, is defined on an edge $(v_i, v_j) \in \mathcal{E}$ by:

$$(d_w f)(v_i, v_j) = \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i)). \tag{1}$$

The *directional derivative* (or *edge derivative* of f, at a vertex $v_i \in \mathcal{V}$, along an edge $e = (v_i, v_j)$, is defined as:

$$\partial_{v_j} f(v_i) = (d_w f)(v_i, v_j). \tag{2}$$

The external and internal *morphological directional partial derivative* operators are respectively defined as [TEL11]:

$$\partial^+_{v_j} f(v_i) \;=\; \big(\partial_{v_j} f(v_i)\big)^+, \tag{3}$$

$$\partial^-_{v_j} f(v_i) \;=\; \big(\partial_{v_j} f(v_i)\big)^-. \tag{4}$$

where $(x)^+ = \max(x, 0)$ and $(x)^- = -\min(x, 0)$. *Discrete upwind non-local weighted gradients* are defined as:

$$(\nabla^\pm_w f)(v_i) = \Big((\partial^\pm_{v_j} f)(v_i)\Big)^T_{v_j \in \mathcal{V}}. \tag{5}$$

The $\mathcal{L}_p$ norms and the $\mathcal{L}_\infty$ of these gradients are defined by:

$$||(\nabla^\pm_w f)(v_i)||_p \;=\; \left[\sum_{v_j \sim v_i} w^p(v_i, v_j)\left[(f(v_j) - f(v_i))^\pm\right]^p\right]^{\frac{1}{p}}, \tag{6}$$

$$||(\nabla^\pm_w f)(v_i)||_\infty \;=\; \max_{v_j \sim v_i}\Big(w(v_i, v_j)|(f(v_j) - f(v_i))^\pm|\Big). \tag{7}$$

In the sequel we will consider only the $\mathcal{L}_\infty$ norm. The $\infty-$Laplacian can be defined from these norms by [EDL12]:

$$(\Delta_{w,\infty} f)(v_i) = \frac{1}{2}\left[||(\nabla^+_w f)(v_i)||_\infty - ||(\nabla^-_w f)(v_i)||_\infty||\right]. \tag{8}$$

C. Morphological Operators on Graphs

Continuous-scale morphology [BM94] defines the flat dilation δ and erosion ϵ of a function $f^0 : \mathbb{R}^m \to \mathbb{R}$ by using structuring sets $B = \{x : ||x||_p \leq 1\}$ with the following general PDEs:

$$\frac{\partial f}{\partial t} = +||\nabla f||_p \text{ and } \frac{\partial f}{\partial t} = -||\nabla f||_p, \tag{9}$$

where f is a modified function of f^0, ∇ is the gradient operator, $||\cdot||_p$ corresponds to the $\mathcal{L}_p$-norm, and one has the initial condition $f = f^0$ at $t = 0$. With different values of p, one obtains different structuring elements: a rhombus for $p = \infty$, a disc for $p = 2$, and a square for $p = 1$ [BM94]. The solution at time n provides a dilation (with the plus sign) or an erosion (with the minus sign) with a structuring element of size $n\Delta t$. We have proposed in [TEL11] the discrete PdEs analogue of these PDEs-based dilation and erosion formulations. This provides the following expression over graphs for a given initial function $f : \mathcal{V} \to \mathbb{R}$, $\forall v_i \in V$:

$$\begin{aligned} \frac{\partial f}{\partial t}(v_i, t) &= +||(\nabla^+_w f)(v_i)||_p \text{ , and} \\ \frac{\partial f}{\partial t}(v_i, t) &= -||(\nabla^-_w f)(v_i)||_p\,. \end{aligned} \tag{10}$$

For the case of $p = \infty$, these morphological processes can be expressed by iterative schemas. For instance, the dilation process can be expressed by:

$$f^{n+1}(v_i) = f^n(v_i) + \Delta t \max_{v_j \sim v_i}\left(\sqrt{w(v_i, v_j)}(f^n(v_j) - f^n(v_i))^+\right). \tag{11}$$

For $\Delta t = 1$ and $w = 1$, one recovers the classical algebraic formulation of mathematical morphology (see [TEL11] for details). In this paper we restrict ourselves to $\Delta t = 1$ but with $w \neq 1$. This enables to introduce adaptivity and provides a formulation of adaptive mathematical morphology on graphs. In the processing, the structuring element B (supposed to be symmetric) is provided by the local neighborhood configurations and expressed by $B(v_i) = \{v_j \sim v_i\} \cup \{v_i\}$. In the special case where $\Delta t = 1$, the dilation PdE can be interpreted as an iterative non-local dilation (NLD) process, and as a non-local erosion (NLE) for the erosion PdE. These processes can be expressed as

$$\begin{aligned} f^{n+1}(v_i) &= NLD(f^n)(v_i) \\ &= f^n(v_i) + ||(\nabla^+_w f^n)(v_i)||_\infty \\ &= f^n(v_i) + \max_{v_j \sim v_i}\big(w(v_i, v_j)(f^n(v_j) - f^n(v_i))^+\big) \end{aligned} \tag{12}$$

for the dilation, and

$$\begin{aligned} f^{n+1}(v_i) &= NLE(f^n)(v_i) \\ &= f^n(v_i) - ||(\nabla^-_w f^n)(v_i)||_\infty \\ &= f^n(v_i) + \min_{v_j \sim v_i}\big(w(v_i, v_j)(f^n(v_i) - f^n(v_j))^-\big) \end{aligned} \tag{13}$$

for the erosion. Additionnaly, it was shown in [EDL12] that the non-local $\infty-$Laplacian (Eq. 8) depends on these NLD and NLE operators:

$$(\Delta_{w,\infty} f)(v_i) = NLA(f)(v_i) - f(v_i), \tag{14}$$

where:

$$NLA(f)(v_i) = \frac{1}{2}[NLD(f)(v_i) + NLE(f)(v_i)]. \tag{15}$$

This approach can be used to define other morphological operators based on erosion ϵ or dilation δ operators, such as openings $\gamma = (\delta\epsilon)$, closings $\phi = (\epsilon\delta)$, or morphological gradients $(\delta - \epsilon)$. For instance we propose a formulation of the non-local closing (NLC) operation that can be defined as:

$$\begin{aligned} \frac{\partial f}{\partial t}(v_i, t) = &- sign^+(t - s + 1)||(\nabla^-_w f)(v_i)||_\infty \\ &+ sign^+(s - t)||(\nabla^+_w f)(v_i)||_\infty, \end{aligned} \tag{16}$$

with $t \in [0, 2s[$ and

$$sign^+(x) = \begin{cases} 1 \text{ if } x > 0, \\ 0 \text{ otherwise.} \end{cases} \tag{17}$$

This PdE has a time-dependent switching coefficient that makes it act as a dilation δ for $t \in [0, s[$ and an erosion ϵ for $t \in [s, 2s[$. This formulation is different from the classical PDEs one [AGLM93] and does not produce discontinuities at the switching time. This can be interpreted as the following non-local iterative process:

$$\begin{aligned} f^{(n+1)}(v_i) &= NLC(f^{(n)})(v_i) \\ &= sign^+(t - s + 1)NLE(f^{(n)})(v_i) \\ &\quad + sign^+(s - t)NLD(f^{(n)})(v_i). \end{aligned} \tag{18}$$

Similarly, one can express the non-local opening NLO as an iterative process.

D. Interpolation and Filtering on Weighted Graphs

Many image processing problems can be formalized as inverse problems. In this section, we show how our formulation of the non-local $\infty-$Laplacian (Eq.14) can be used for two inverse problems: interpolation and filtering.

1) Interpolation: Many tasks in image processing and computer vision can be formulated as interpolation problems. Interpolating data consists in constructing new values for missing data in coherence with a set of known data. Our motivation for using the non-local ∞-Laplacian on graphs for interpolation stems from the fact that flexible data processing tools that can be adapted easily to general domains modeled by graphs are needed. Recent works on inpainting tend to unify local and non-local approaches under a variational formulation (see [AFCS11] and references therein for more details). We presented a unifying approach of local geometric methods and non-local exemplar-based ones for inpainting [GCE09] using the framework of discrete non-local regularization on graphs introduced in [ELB08]. We consider that data are defined on a general domain represented on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. Let $f^0 : \mathcal{V} \to \mathbb{R}$ be a function. Let $A \subset \mathcal{V}$ be the subset of vertices with unknown values and ∂A the subset of vertices with known values. The purpose of interpolation is to find a function f^* approximating f^0 in $\mathcal{V}$ minimizing the energy:

$$\begin{cases} (\Delta_{w,\infty} f)(v_i) = 0 & \forall u \in A, \\ f(u) = f^0(v_i) & \forall u \in \partial A. \end{cases} \tag{19}$$

The solution f^* is said to be infinity-harmonic [GEL11].

2) Filtering: Being a typical inverse problem, filtering is a challenging task and basically addresses the problem of estimating the true signal from its noisy measured version. We introduce the use of the $\infty-$Laplacian on weighted graphs for data restoration. Let us consider a function $f^0 : \mathcal{V} \to \mathbb{R}$ of a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$. f^0 is an observation of an original function f corrupted by a noise n: $f^0 = f^* + n$. The purpose of restoration is to recover f^* from its noisy representation f^0. Thus, the discrete regularization of f^0 using the non-local $\infty-$Laplacian operator consists in seeking a function f^* that is smooth enough on $\mathcal{G}$. This corresponds to consider

$$\begin{cases} \dfrac{\partial f}{\partial t} & = \Delta_{w,\infty} f, \\ f(.,0) & = f^0. \end{cases} \tag{20}$$

3) Interpolation and filtering algorithms: Whatever the problems (interpolation or filtering), works of [EDL12] have shown that both problems have a unique solution that can be obtained using the following iterative algorithm:

$$\begin{cases} f^{(0)}(v_i) = f^0(v_i), \\ f^{(n+1)}(v_i) = NLA(f^{(n)})(v_i). \end{cases} \tag{21}$$

The solution is obtained with this simple iterative algorithm based on the NLA operator introduced in Eq. 15. The iterative algorithm presented in Eq. 21 converges to the unique solution [EDL12]. This general equation describes a family of discrete diffusion processes, parameterized by the structure of the graph (topology and weight function w). Modifying both graph topology and graph weights enables to perform both local and non-local inpainting / filtering within the same framework of PdEs. For the filtering case, all vertices are updated in parallel: $\forall u \in \mathcal{V}$ one applies iteratively Eq. 15, as shown in Eq. 21. For the special case of inpainting, at each iteration, only the internal boundary $\partial^- A = \{v_i \in A | \exists v_j \sim v_i, v_j \in \partial A\}$ is inpainted:

$$\begin{cases} NLA(f)(v_i) = \frac{1}{2}[NLD(f)(v_i) + NLE(E)(v_i)] & \forall v_i \in \partial^- A, \\ NLA(f)(v_i) = f^0(v_i) & \forall v_i \in \partial A. \end{cases} \tag{22}$$

At the end of each iteration the set ∂A is updated by $\partial A^{(n+1)} = \partial A^{(n)} \cup \partial^- A^{(n)}$ and $\partial^- A^{(n+1)}$ is updated from $\partial A^{(n+1)}$. The algorithm stops when the set of vertices to inpaint is empty.

III. Construction of a weighted graph from a point cloud

In this section, we explain how a weighted graph based on patches can be associated with a point cloud. This relies on three steps that we detail in the sequel.

A. Graph Creation from Data Points

First step consists in defining the sets $\mathcal{V}$ and $\mathcal{E}$ from a given point cloud. Let us consider a point cloud P as a set of data points $\{\mathbf{p}_1, \ldots, \mathbf{p}_n\} \in \mathbb{R}^3$. There are many ways of associating a graph, that encodes proximity between points, to such a data set. To each data point we first associate a vertex of a proximity graph $\mathcal{G}$ to define a set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. Then, determining the edge set $\mathcal{E}$ of the proximity graph $\mathcal{G}$ requires defining the neighbors of each vertex v_i according to its embedding $\mathbf{p}_i$ using the Euclidean distance. We will denote as $\mathcal{D}(v_i, v_j) = \|\mathbf{p}_i - \mathbf{p}_j\|_2$ the Euclidean distance between vertices. We consider the k Nearest Neighbors Graph (k-NNG): $v_j \sim v_i$ if the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ is among the k-th smallest distances from $\mathbf{p}_i$ to all the other data points. To conclude, the first step consists in associating a k-NNG to the 3D point cloud. The value of k will be denoted k_G for the graph $\mathcal{G}$ associated with the point cloud. To speed up the knn algorithm, a kD-tree is used [AMN$^+$98].

Once the graph has been created, it has to be weighted. If one does not want to take care of the vertices similarities, the weight function w can be set to $w = 1$. A better one can be obtained using patches [BCM10]. For images, a patch $\boldsymbol{\mathcal{P}}(v_i)$ centered at a vertex $v_i \in \mathcal{V}$ is a vector of values (e.g., coordinates, intensities, ...) defined by $\boldsymbol{\mathcal{P}}(v_i) = \big(f^0(v_j) : v_j \in B(v_i, n)\big)^T$ where $B(v_i, n)$ is a square of size n^2 centered at v_i. Using patches, $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is defined by:

$$w(v_i, v_j) = exp\left(-\frac{||\boldsymbol{\mathcal{P}}(v_i) - \boldsymbol{\mathcal{P}}(v_j)||_2^2}{\sigma^2}\right). \tag{23}$$

B. Patch Orientation

An important feature for the inpainting and the restoration of images relies on the use of patches [BCM10], [AFCS11]. Unfortunetely there is no straightforward extension of the definition of patches for point clouds, as well for the content of the patch than for its orientation. Therefore the second step consists in estimating the orientation of each patch. Indeed, since two patches can have very different orientations in the point cloud, we need to estimate this orientation to be able to compare the patches.

In our previous works [LEL12], we have proposed the following strategy to estimate the patch orientation. Let us recall it briefly. Point clouds are first smoothed, using a local filtering. From this filtered version, a PCA is locally applied on the k_n nearest neighbors of each point $\mathbf{p}_i$. This enables to define the normal to each point $\mathbf{p}_i$ (associated with each vertex v_i) as $\mathbf{n}(v_i)$ (see [LEL12]). Next, patches are oriented from principal directions computed on this smoothed point cloud. This means that directions of first and second axis of the patch basis will coincide respectively with major and minor principal directions. To compute these principal directions at point $\mathbf{p}_i$, we used the arguments of [Dig12]: principal directions can be estimated as the eigenvectors of a PCA of the covariance matrix of the normals of the neighbors of $\mathbf{p}_i$.

However this strategy is not always efficient. Indeed, because the orientation of patches are computed from principal directions, these orientations highly depend on the repartition of points in the 3D space. So, similar parts of a point cloud will produce similar orientations of the patches. Unfortunetly, because the obtained orientation depends highly on the most predominant axis, one can find different patches orientations for similars points repartitions, and conversely. Therefore we propose another way to obtain the patch orientation.

Another way to orient patches is to deduce an orientation from normal vectors. On the contrary to the principal direction method presented above, which depends on the repartition of neighboring points, it is better to deduce the orientation from the normals. Indeed, this will produce the same orientations for points that have similar normals. The proposed algorithm is therefore to first deduce a tangent vector $\mathbf{t}(v_i)$ from the normal vector $\mathbf{n}(v_i)$. Let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be the three axis of a 3D space, the tangent vector $\mathbf{t}(v_i)$ is computed with:

$$\begin{cases} \mathbf{t}(v_i) = \mathbf{z} \times \mathbf{n}(v_i) \text{ if } (\mathbf{x} \cdot \mathbf{n}(v_i)) > (\mathbf{z} \cdot \mathbf{n}(v_i)), \\ \mathbf{t}(v_i) = \mathbf{z} \times \mathbf{n}(v_i) \text{ if } (\mathbf{y} \cdot \mathbf{n}(v_i)) > (\mathbf{z} \cdot \mathbf{n}(v_i)), \\ \mathbf{t}(v_i) = \mathbf{x} \times \mathbf{n}(v_i) \text{ otherwise,} \end{cases} \tag{24}$$

with $\times$ the cross product operator, and $\cdot$ the dot product operator. Then a bitangent vector $\mathbf{b}(v_i)$ is computed by $\mathbf{b}(v_i) = \mathbf{n}(v_i) \times \mathbf{t}(v_i)$. The orientations vectors $\mathbf{o}_0(v_i), \mathbf{o}_1(v_i), \mathbf{o}_2(v_i)$ are then respectively assigned to $\mathbf{t}(v_i), \mathbf{b}(v_i), \mathbf{n}(v_i)$.

C. Patch Construction

Final step consists in constructing the patches. Given a point $\mathbf{p}_i$, defining a patch for this point comes to construct a square grid around $\mathbf{p}_i$ on its tangent plane. We fix the patch length l manually. Let n be the number of cells on a row of the patch. A square lattice of n^2 cells is constructed around $\mathbf{p}_i$ with respect to the basis obtained from orientation computation. Each cell has a side length of l/n. A local graph is then considered that connects the vertex v_i to all the vertices v_j contained in a sphere of diameter $l\sqrt{2}$. Then, all the neighbors v_j of v_i are projected on the tangent plane of $\mathbf{p}_i$ giving rise to projected points $\mathbf{p}'_i$. To fill the patch with values, these projected points $\mathbf{p}'_i$ are affected to the cells the center of which is the closest. The value of the cell is then deduced from a weighted average of the values $f^0(v_j)$ associated with the vertices v_j that where affected to the patch cell. This value is a spectral value (the points' colors). The set of values inside the patch of the vertex v_i are denoted as $\mathcal{P}(v_i)$. Let $C_k(v_i)$ denotes the kth cell of the constructed patch around v_i with $k \in [1, n^2]$. With the proposed patch construction process, one can define the set $V_k(v_i) = \{v_j \mid \mathbf{p}'_j \in C_k(v_i)\}$ as the set of vertices v_j that were affected to the kth patch cell of v_i. Then, the patch vector is defined as $\mathcal{P}(v_i) = \left(\frac{\sum_{v_j \in V_k(v_i)} w(\mathbf{c}_k, \mathbf{p}_j) f^0(v_j)}{|V_k(v_i)|} \right)^T_{k \in [1,n^2]}$, where $w(\mathbf{c}_k, \mathbf{p}_i) = exp(-\frac{||\mathbf{c}_k - \mathbf{p}'_i||_2^2}{\sigma^2})$ where $\mathbf{c_k}$ is the coordinates' vector of the kth patch cell center. This weighting enables to take into account the repartition of the points in the cells' patch to compute there mean feature vectors.

IV. Applications in cultural heritage

This section shows some examples of restoration and inpainting with the proposed morphological filters on graphs constructed from 3D point clouds. It is important to note that the presented results show very dense point clouds and not meshes.

A. Morphological Operators of Colored Point Clouds

Let P be a point cloud, that associates an intensity to each point $p \in P$. From the latter point cloud, a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ is first created using the method presented in Sec. III (i.e., a k_G nearest neighbor graph). Then this graph is filtered using some morphological operators, as explained in Sec. II-C. Fig. 3 shows some morphological processings of a tower of the bishop castle point cloud[2] obtained with an Optech Lidar scanner. Non-local filtering results correspond to the case where w is computed from patches as presented in Sec. III-C. Cells of each patches are built from the averaging of colors of neighbors points. Local filtering corresponds to the case where $w = 1$ with a small neighborhood whereas non-local filtering uses patches with a very large neighborhood. Results in Fig. 3 show that, with non-local filtering, the boundaries are better preserved.

Let us now consider a noisy point cloud. A filtered version of this point cloud can be obtained using the iterative algorithm presented in Eq. 21. This latter algorithm is based on the use of the $\infty-$Laplacian, which is a combinaison of erosion and dilation processes. Fig. 4 shows the use of $\infty-$Laplacian to filter the whole bishop castle point cloud. As it can be seen, the non-local $\infty-$Laplacian produces a very good visually simplified version of the point cloud: details corresponding to high frequencies are wiped out, whereas boundaries corresponding to low frequencies are preserved.

B. Color Inpainting on Point Cloud

Let P be a point cloud that has some colors to restore. The set of points to restore is denoted by A and is given by the user. From this point cloud P, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ is built using the method presented in Sec. III. The similarity function $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is computed from patches, and the patch

[2]Gracefully given by www.cloudcasterlite.com

at a vertex $v_i \in \mathcal{V}$ contains the average colors of projected neighbors of v_i. Fig. 1 shows the inpainting on a scanned real person using the algorithm presented in Eq.21. A part of the T-shirt has been removed by the user and successfully inpainted.

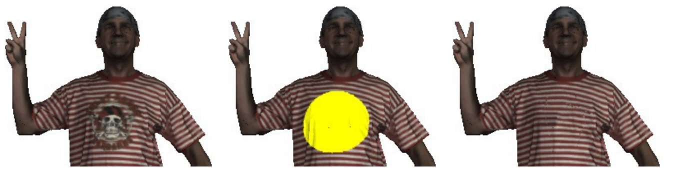


Fig. 1: Real object (with 127,039 points) inpainting with $k_G = 30000$ and $n^2 = 81$. From left to right respectively: the original man (obtained from www.cyberware.com), the part to be inpainted shown in yellow, the inpainted man.

C. Geometry and Color Inpainting on Point Cloud

Many real scanned objects in cultural heritage have real defects, a typical one being missing parts of the object. The challenge of the inpainting is then not only to restore the color but also to restore the geometry of the missing part. We propose thus a method to restore these missing parts. Let $f : \mathcal{V} \to \mathbb{R}^3$ be a function that associates to each node $v_i \in \mathcal{V}$ his 3D coordinates $\mathbf{p}_i$ in a point cloud P_1. A user manually delineates the part of the object that has to be virtually repaired. The algorithm is composed of two steps. First, the tangent plane to the missing part is determined. The boundary of the missing part is projected on this plane. Its convex hull is computed and points are added in such a way that the sampling allows to cover a dilated version of the convex hull. Let P_2 denote this set of generated points. From the union P of P_2 and the initial point cloud P_1, a nearest neighbors graph is constructed. The graph is weighted with patches constructed in a different way than for color inpainting. Indeed, to enable the generated points to fit correctly real parts of the object, patches are constructed from P and filled with distances $||\mathbf{p}_i - \mathbf{p}'_i||_2$. Then the algorithm presented in Eq.21 is used on points of P_2 but the patches (and also the weights) are updated after each iteration to enable the iterative deformation of the points of P_2. Fig. 2 shows the full process of virtual restoration of a antique broken vase. First the hole is filled with points to cover the missing part, next the color of the geometricaly recovered part is inpainted as in Sec. IV-B.

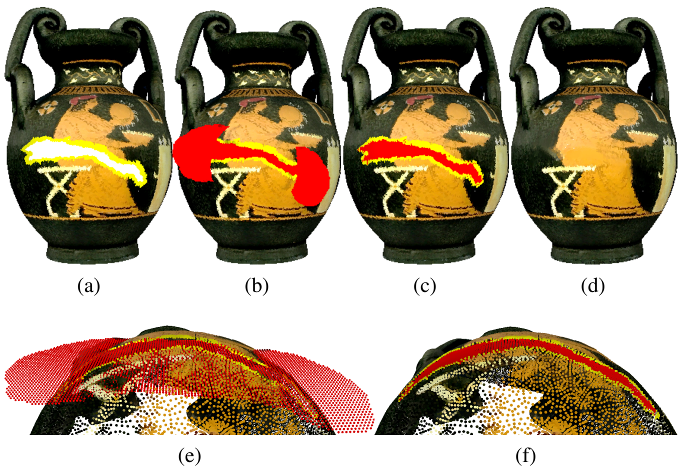


Fig. 2: Virtual restoration of a broken antique vase (with 220,994 points) with $k_G = 4000$ and $n^2 = 81$. (a) broken vase with labelized part to restore, (b) points' sampling on the tangent plane, (c) geometric filtering result, (d) color inpainting result, (e) zoom of a part of (b), (f) zoom of a part of (c).

V. Conclusion

In this paper, we have shown how morphological PDEs can be adapted on raw point clouds. Our proposal relies on several key components: the use of the framework of PdEs to adapt PDEs on graphs, a derived formulation of adaptive morphological operators on graphs, a unified formulation of interpolation and filtering with the $\infty-$Laplacian, and a specfic way to construct patches for point clouds. We have presented results that show the benefits of the approach for applications in cultural heritage.

References

[AFCS11] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro. A variational framework for exemplar-based image inpainting. *Int. J. Comput. Vision*, 93(3):319–347, 2011.

[AGLM93] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel. Axioms and Fundamental Equations of Image Processing. *Arch. Ration. Mech. An.*, 123(3):199–257, 1993.

[AMN$^+$98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, November 1998.

[BCM10] A. Buades, B. Coll, and J.-M. Morel. Image denoising methods. a new nonlocal principle. *SIAM Review*, 52(1):113–147, 2010.

[BKP$^+$10] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. *Polygon Mesh Processing*. AK Peters, 2010.

[BM94] R. W. Brockett and P. Maragos. Evolution equations for continuous-scale morphological filtering. *IEEE T. Signal. Proces.*, 42(12):3377–3386, 1994.

[Dig12] J. Digne. Similarity based filtering of point clouds. In *CVPR Workshops*, pages 73 –79, june 2012.

[EDL12] A. Elmoataz, X. Desquesnes, and O. Lézoray. Non-local morphological pdes and p-laplacian equation on graphs with applications in image processing and machine learning. *IEEE J. Sel. Top. Signa.*, 6:764–779, 2012.

[ELB08] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing. *IEEE T. Image. Process.*, 17(7):1047–1060, 2008.

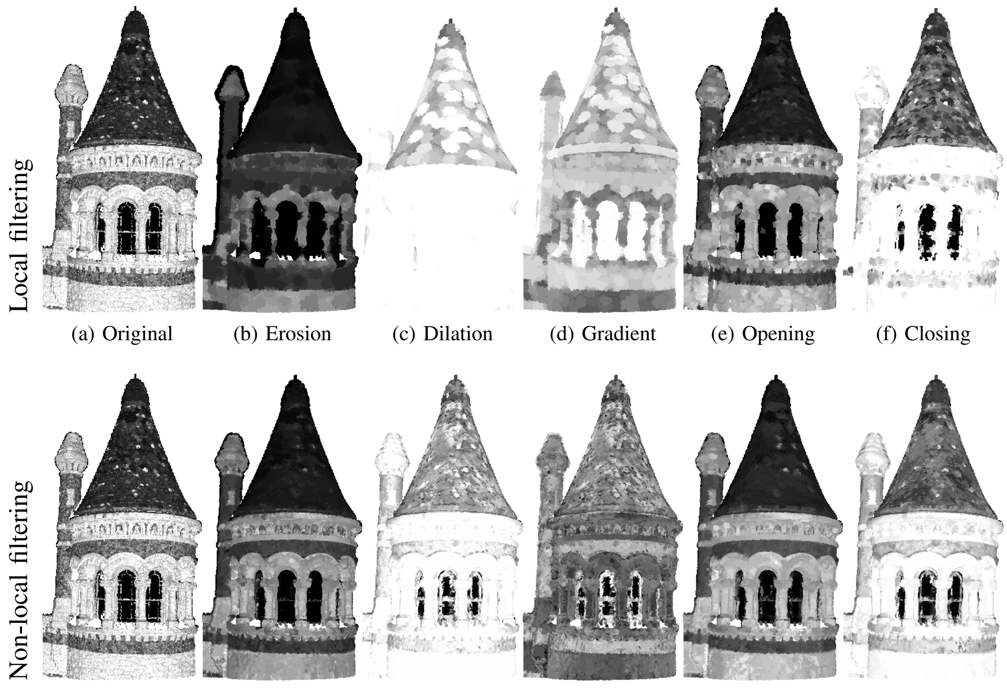
[GCE09] M. Ghoniem, Y. Chahir, and A. Elmoataz. Geometric and texture inpainting based on discrete regularization on graphs. In *ICIP*, pages 1349–1352, 2009.

[GEL11] M. Ghoniem, A. Elmoataz, and O. Lézoray. Discrete infinity harmonic functions: towards a unified interpolation framework on graphs. In *ICIP*, pages 1361–1364, 2011.

[LEL12] F. Lozes, A. Elmoataz, and O. Lézoray. Nonlocal processing of 3d colored point clouds. In *ICPR*, pages 1968–1971, 2012.

[SBG11] F. Stanco, S. Battiato, and G. Gallo. *Digital Imaging for Cultural Heritage Preservation: Analysis, Restoration, and Reconstruction of Ancient Artworks*. Digital Imaging and Computer Vision Series. Taylor and Francis, 2011.

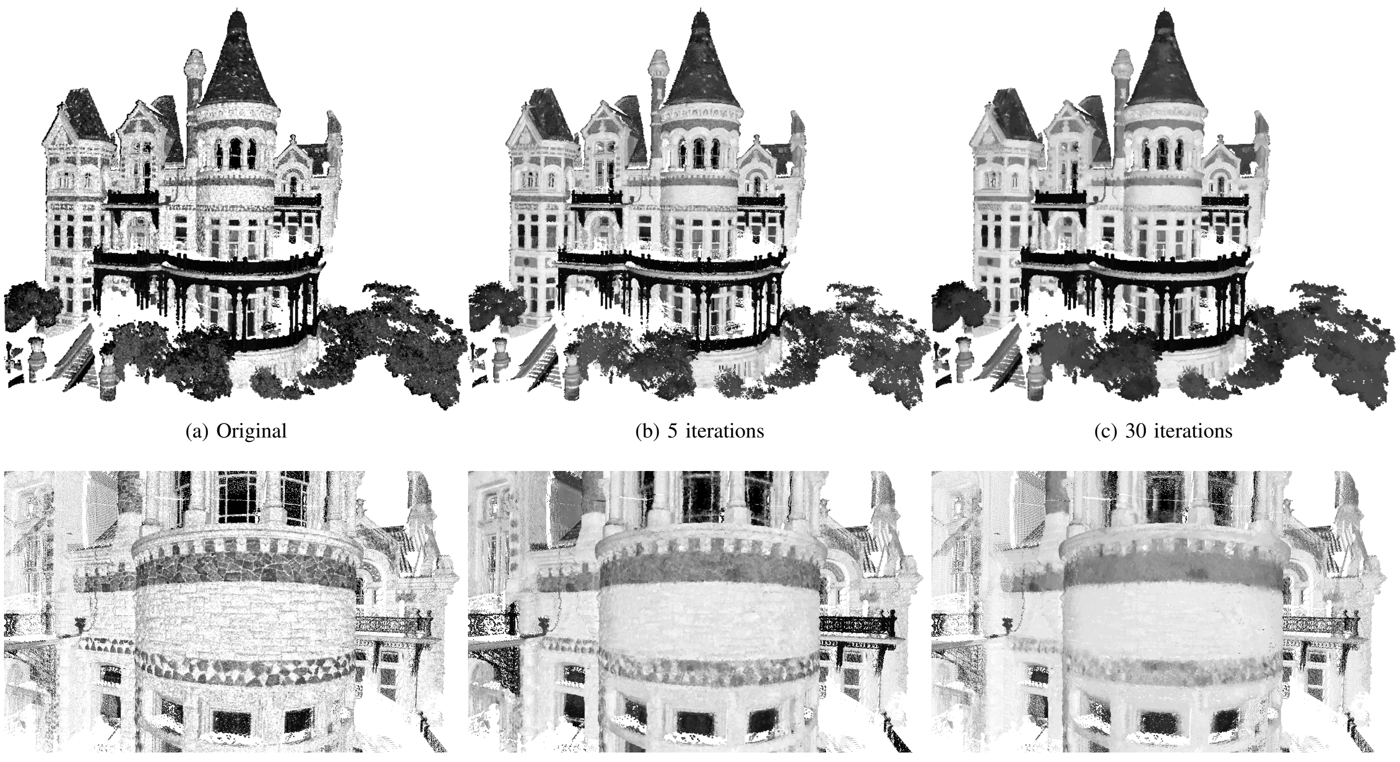
[TEL11] V.-T. Ta, A. Elmoataz, and O. Lézoray. Nonlocal pdes-based morphology on weighted graphs for image and data processing. *IEEE T. Image. Process.*, 26(2):1504–1516, june 2011.



(a) Original (b) Erosion (c) Dilation (d) Gradient (e) Opening (f) Closing

(g) Original (h) Erosion (i) Dilation (j) Gradient (k) Opening (l) Closing

Fig. 3: Morphological operators on a colored point cloud (with 219,699 points) after 10 iterations with $k_G = 1000$ and $n^2 = 25$. Local filtering: $w = 1$ and $k_G = 8$. Non-local filtering: w is a similarity measure between patches.



(a) Original (b) 5 iterations (c) 30 iterations

(d) Original (e) 5 iterations (f) 30 iterations

Fig. 4: Non-local $\infty-$Laplacian on a bishop castle (with 1,815,044 points) after several iterations with $k_G = 1000$ and $n^2 = 25$. From upper to bottom, view of the full castle, zoom at a corner of a tower of the castle.