# Partial difference operators on weighted graphs for image processing on surfaces and point clouds.

François Lozes, Abderrahim Elmoataz, Olivier Lézoray

***Abstract*—PDEs and variational methods for image processing on Euclidean domains spaces are very well established because they permit to solve a large range of real computer vision problems. With the recent advent of many 3D sensors, there is a growing interest in transposing and solving PDEs on surfaces and point clouds. In this paper, we propose a simple method to solve such PDEs using the framework of Partial difference Equations (PdEs) on graphs. This latter approach enables us to transcribe, for surfaces and point clouds, many models and algorithms designed for image processing. To illustrate our proposal, three problems are considered: $p$-Laplacian restoration and inpainting, PDEs Mathematical Morphology, and active contours segmentation.**

## I. Introduction

PDEs and variational methods are one of the most important tools widely used for modeling and solving inverse problems, e.g., in image processing and computer vision. They have been applied with success in many applications tasks such as image or video denoising, image inpainting, image segmentation, etc. Recently, many of these methods were extended to non-local forms [1, 2] that have shown their ability to preserve textures and repetitive structures. However, most of the research works on local or non-local processing focus only on image processing on Euclidean spaces. With 3D sensors becoming cheaper, there is a huge need in the processing of 3D surfaces or 3D point clouds. Indeed, many application fields such as medical imaging, industrial vision, terrestrial imaging now directly consider such 3D data. Recently, researchers have been interested in transposing and solving PDEs and variational problems on general surfaces or manifolds. PDEs on surfaces are traditionally handled by techniques that can be roughly classified in three categories whether they make use of either explicit, implicit or intrinsic representations.

The first category of techniques is using explicit representations of surfaces represented by triangular meshes [3, 4]. By relying on a specific parametrization to the given surface, differential operators can be defined and computed analytically. However the computation of a parametrization is a difficult task for arbitrary given surfaces and topological changes can be hard to handle.

The authors are with the Université de Caen Basse-Normandie and the ENSICAEN in the GREYC UMR CNRS 6072 Laboratory, Image Team, 6 Boulevard Maréchal Juin, F-14050 CAEN cedex FRANCE.
Emails: {francois.lozes, abderrahim.elmoataz-billah, olivier.lezoray }@unicaen.fr
Phone: +33(0)231452706, Fax: +00(0)231452698

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

The second category of techniques is using implicit representations of surfaces represented by a zero level-set function of a signed distance function in Euclidean domains. The differential operators are then approximated by combining the Euclidean differential operators with a projection along the normal direction [5, 6]. For instance, in [7] the coordinates of the closest point for each point of the surface is used and fast algorithms can be obtained in Euclidean domains. Implicit representation can deal easily with topological changes but all the data has to be extended to the definition domain of the implicit function.

The third category of techniques is using intrinsic geometry to study variational problems directly on the surface represented as a triangular mesh. Lai and Chan have recently proposed [8] a framework for intrinsic image processing on surfaces. They approximate surface differential operators such as surface gradient and divergence by specific intrinsic differential geometry definition. Intrinsic methods do not need any pre-processing but they require a specific discretization scheme on triangles.

As we just detailed it, the three categories of methods all suffer from some restrictions. Some explicit and intrinsic methods consider surfaces as a set of triangles called a triangular mesh. Indeed, for triangulated surfaces, the definition of differential operators can be defined in a rigorous way whereas this is much more difficult for arbitrary manifolds. For point clouds, the processing is even more difficult since we lack any structuring information. Indeed, contrary to triangular meshes, 3D points clouds are not associated with any connectivity. Moreover, if one considers 3D point clouds and not triangular meshes, it is very difficult to express any variational algorithm since the definition of the basic differential operators have specific connectivity requirements (triangles are needed) available only for triangular meshes. Finally, implicit methods can be time-consuming since one has to cope with very large volumes.

Some authors have recently considered the problem of solving PDEs on point clouds [9, 10]. However both these approaches use intermediate representations to approximate differential operators on point clouds. [10] use a local triangulation that necessitates a pre-processing. This pre-processing is needed for the authors to estimate their differential operators on triangular meshes and their method can be categorized as an intrinsic method. [9] compute a local approximation of the manifold using moving least squares from the $k$-nearest neighbors. From this local coordinate system, a local metric tensor is computed at each point to be able to differentiate on the manifold. This method can therefore be categorized as an explicit method. As we can see, even if these methods consider point clouds, they rely on intermediate representations to locally approximate manifolds.

Regarding all these difficulties, a different approach is needed that can cope with both 3D meshes and point clouds of the arbitrary connectivities. To achieve this goal, we propose in this paper a new approach that can be used to overcome all the problems or drawbacks that explicit / implicit / intrinsic methods do suffer: no parametrization is needed, no pre-processing is required, no assumption on the surfaces' graphs topologies is necessary. Since 3D point clouds and triangular meshes can have very different topologies, we propose to rely on a new class of methods that directly works in any discrete domain.

In this approach, surfaces (composed of a collection of any polygons and not necessarily triangular meshes) and 3D point clouds are represented under the same structure of weighted arbitrary graphs. Then we use the framework of Partial difference Equations (PdEs) on weighted graphs [11] to transcribe PDEs and variational methods in the discrete settings of graphs. Conceptually PdEs mimic the PDEs on a general domain by replacing the differential operators by non-local difference operators such as: difference, gradient, divergence, $p$-laplacian, etc. This graph version of differential operators will enable us to study image processing problems on general graphs representing surfaces and 3D point clouds. By the way, the framework we propose can extend classical PDEs on Euclidean domains to arbitrary graphs and give rise to many new problems transposed as PdEs on graphs. Regarding the other approaches of the literature, our approach to transpose PDEs on surfaces or point clouds (represented as graphs) has many advantages: no parametrization is needed, no pre-processing is required, no assumption on the surface graph topology is necessary, the use of graph unifies local and non-local processing, and a large range of PDEs can be easily adapted on arbitrary weighted graphs.

Our approach allows to transpose and extend many processes in classical image processing to surfaces and points cloud. In this paper, to illustrate our approaches, we consider four representative problems for image processing on surfaces and point clouds: $p$-Laplacian regularization denoising, morphological processing, inpainting and global minimization for segmentation. As far as we known, such problems on raw point clouds have been few investigated in literature.

Before entering into the details of our approach, we present our new contributions and also highlight the differences with our previous works. In our previous works [11], we have considered many image processing problems formulated on graphs using the framework of partial difference operators that we proposed. In this framework the derivatives on graph of a function $f$ at a given vertex are deconnected from the geometrical organization of the data (i.e., the graph topology). To cope with this, weights are introduced for the derivatives to incorporate the geometry. This is very important to enable for example non-local processing. In all our previous works, we have considered only the cases of images, meshes, and high dimensional manifolds [12, 11, 13]. In this paper we focus on a very challenging type of data: raw point clouds. To describe the geometry of the data, we propose a specific patch definition and construction that enables to account for similar geometric configurations. This framework is the first to propose a unified way to process a function on any arbitrary graph representing a surface or a point cloud, and there is no equivalent in the literature. One strong benefit of our approach is that it enables to process the color associated to 3D point clouds whereas usual mesh filtering consider only vertex coordinates filtering. However, we want to stress that even if we consider point clouds in this paper, all the processing that we will proposed can be applied to any type of graph signal.

In [11, 13], we have proposed $p$-Laplacian regularization of graph signals. In this paper, we propose to unify under a same formulation, in a divergence form, both the isotropic and anisotropic formulations of the $p$-Laplacian. Since this new formulation unifies many works of the litterature (including our previous works), we also show how with different configurations (graph topology and weights, regularization parameters), we can recover classical approaches for regularizing graph signals such as images. In addition, since we consider point clouds, we also show how we can recover state-of-the-art filtering approaches for meshes and point clouds. This shows the genericity of our new proposal.

In [12], we have proposed an adaptation of PDEs based mathematical morphology with convex structuring elements. We extend this work by considering new adaptive structuring elements that provide more adaptation in the processing, especially in non-local configurations. Given this new proposal we show how this is related to our previous works. Then, we consider the problem of graph signal inpainting as a morphological process based on non-local infinity Laplacian interpolation. We show how this can be used to perform both patch-based non-local geometric and color inpainting of point clouds described as graphs. To the best of our knowledge, there is no equivalent in the litterature.

In [14], we have drawn the bases for the adaptation of active contours on graphs. In this paper, we propose to use the Chambolle and Pock algorithm [15] on weighted graphs to perform the optimization and we apply it for point clouds clustering. To the best of our knowledge this is the first method to proposed non-local patch-based segmentation of colored point clouds on graphs.

The rest of the paper is organized as follows. In Section II, we review the principle of PdEs on graphs and provide all the necessary notations and definitions of our graph-based difference operators. After that, in Section III, we first generalize the $p$-Laplacian on graphs for solving the concept of $p$-Laplacian regularization on surfaces and point clouds. In section IV and V, we present our generalization of Mathematical Morphology and Active Contours to surfaces and point clouds. Section VI details how to build graphs from surfaces and point clouds, as well as how to weight these graphs accounting local neighborhood information. Section VII provides experimental results and last section concludes.

## II. PdEs on weighted graphs

In this section, we recall definitions and present difference and $p$-Laplacian operators on graphs. This constitutes the basis of the framework of PdEs on a graph that enables to transpose PDEs on graphs. Most of these definitions are borrowed from [11].

### A. Definitions

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 $f \in \mathcal{H}(\mathcal{V})$ assigns a real value $f(v_i)$ to each vertex $v_i \in \mathcal{V}$.

Similarly, we define the space $\mathcal{H}(\mathcal{E})$ of functions defined on the set $\mathcal{E}$ of edges. Given a function $f : \mathcal{V} \to \mathbb{R}$, the integral of $f$ is defined as:

$$\int_{\mathcal{V}} f = \sum_{v_i \in \mathcal{V}} f(v_i),$$

and its $\ell_p$ and $\ell_\infty$ norms are given by:

$$||f||_p = \left( \sum_{v_i \in \mathcal{V}} |f(v_i)|^p \right)^{1/p}, \quad \text{with } 1 \leq p \leq \infty, \tag{1}$$

$$||f||_\infty = \max_{v_i \in \mathcal{V}} |f(v_i)|, \quad \text{for } p = \infty. \tag{2}$$

### B. Difference operators on weighted graphs

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be a weighted graph and $w : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$ a weight function that depends on the interactions between the vertices. The *difference operator* [11], denoted $d_w : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{E})$, is defined for all $f \in \mathcal{H}(\mathcal{V})$ and $(v_i, v_j) \in \mathcal{E}$ by:

$$(d_w f)(v_i, v_j) = \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i)). \tag{3}$$

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:

$$\partial_{v_j} f(v_i) = (d_w f)(v_i, v_j). \tag{4}$$

The *adjoint* of the difference operator, denoted $d_w^* : \mathcal{H}(\mathcal{E}) \to \mathcal{H}(\mathcal{V})$, is the unique linear operator satisfying

$$\langle d_w f, H \rangle_{\mathcal{H}(\mathcal{E})} = \langle f, d_w^* H \rangle_{\mathcal{H}(\mathcal{V})}, \tag{5}$$

for all $f \in \mathcal{H}(\mathcal{V})$ and all $H \in \mathcal{H}(\mathcal{E})$. Its expression is given by:

$$(d_w^* H)(v_i) = \sum_{v_j \sim v_i} \sqrt{w(v_i, v_j)}(H(v_j, v_i) - H(v_i, v_j)). \tag{6}$$

The *divergence operator*, defined by $-d_w^*$, measures the net outflow of a function of $\mathcal{H}(\mathcal{E})$ at each vertex of the graph. The *weighted gradient operator* of a function $f \in \mathcal{H}(\mathcal{V})$, at a vertex $v_i \in \mathcal{V}$, is the vector defined by:

$$(\nabla_w f)(v_i) = ((d_w f)(v_i, v_j))^T_{v_j \in \mathcal{V}}. \tag{7}$$

The $\ell_p$ norm of this vector is defined, for $p \geq 1$, by:

$$\|(\nabla_w f)(v_i)\|_p = \left( \sum_{v_j \sim v_i} w(v_i, v_j)^{p/2} \big| f(v_j) - f(v_i) \big|^p \right)^{1/p}. \tag{8}$$

The external and internal *morphological directional partial derivative* operators are respectively defined as [12]:

$$\begin{cases} \partial^+_{v_j} f(v_i) = (\partial_{v_j} f(v_i))^+ \\ \partial^-_{v_j} f(v_i) = (\partial_{v_j} f(v_i))^- \end{cases}, \tag{9}$$

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

$$(\nabla^\pm_w f)(v_i) = \left( (\partial^\pm_{v_j} f)(v_i) \right)^T_{v_j \in \mathcal{V}}. \tag{10}$$

The $\ell_p$ and the $\ell_\infty$ norms of these gradients are defined by:

$$\|(\nabla^\pm_w f)(v_i)\|_p = \left[ \sum_{v_j \sim v_i} w(v_i, v_j)^{\frac{p}{2}} \left( (f(v_j) - f(v_i))^\pm \right)^p \right]^{\frac{1}{p}}, \tag{11}$$

$$\|(\nabla^\pm_w f)(v_i)\|_\infty = \max_{v_j \sim v_i} \sqrt{w(v_i, v_j)}(f(v_j) - f(v_i))^\pm). \tag{12}$$

### C. p-Laplace operators on weighted graphs

The *isotropic weighted $p$-Laplace operator*, with $p \in [1, +\infty[$, at a vertex $v_i \in \mathcal{V}$ is defined on $\mathcal{H}(\mathcal{V})$ by [11]:

$$(\Delta^i_{w,p} f)(v_i) = \frac{1}{2} d_w^* (||\nabla_w f||_2^{p-2} d_w f)(v_i), \tag{13}$$

$$= \sum_{v_j \sim v_i} \psi^i_{w,p}(v_i, v_j)(f(v_i) - f(v_j)), \tag{14}$$

where

$$\psi^i_{w,p}(v_i, v_j) = \frac{1}{2} w(v_i, v_j)(||\nabla_w f(v_i)||_2^{p-2} + ||\nabla_w f(v_j)||_2^{p-2}). \tag{15}$$

The *anisotropic weighted $p$-Laplace operator*, with $p \in [1, +\infty[$, at a vertex $v_i \in \mathcal{V}$ is defined on $\mathcal{H}(\mathcal{V})$ by [13]:

$$(\Delta^a_{w,p} f)(v_i) = \frac{1}{2} d_w^* (|d_w f|^{p-2} d_w f)(v_i), \tag{16}$$

$$= \sum_{v_j \sim v_i} \psi^a_{w,p}(v_i, v_j)(f(v_i) - f(v_j)), \tag{17}$$

where

$$\psi^a_{w,p}(v_i, v_j) = w(v_i, v_j)^{\frac{p}{2}} |f(v_j) - f(v_i)|^{p-2}. \tag{18}$$

When $p = 2$, we recover the combinatorial Laplacian operator [11]:

$$\Delta^i_{w,2} = \Delta^a_{w,2} = \Delta. \tag{19}$$

The $\infty$-*Laplacian* is defined by [16]:

$$(\Delta_{w,\infty}f)(v_i) = \frac{1}{2}\left[||(\nabla_w^+ f)(v_i)||_\infty - ||(\nabla_w^- f)(v_i)||_\infty||\right]. \tag{20}$$

From all the definitions of these discrete difference operators on graphs, we are now in position to transpose any PDEs that involves gradient, $p$-Laplacian or $\infty$-Laplacian in their continuous formulation on Euclidean domains. In the following sections, we present several transcriptions of non-local PDEs considered in image processing. We adapt them to arbitrary weighted graphs (for representing surfaces and point clouds) to perform $p$-Laplacian regularization, adaptive Mathematical Morphology, and convex relaxation for segmentation.

## III. Unified $p$-Laplacian regularization on graphs

In this section we consider a first transposition of PDEs on graphs and propose a methodology to regularize functions defined on the vertices of graphs. To do so, we consider a new family of $p$-Laplace operators based on a divergence formulation, that unifies both the isotropic and anisotropic $p$-Laplacian on graphs proposed earlier [11, 13]. This section will provide only the underlying theory, experiments of local and non-local regularization of functions on surfaces and point clouds will be presented in section VII. To start this section, let us first recall the problem we consider.

### A. Problem formulation

Let $f^0 \in \mathcal{H}(\mathcal{V})$ be a given function defined on the vertices of a weighted graph $G = (\mathcal{V}, \mathcal{E}, w)$. In a given context this function represents an observation of a clean function $h \in \mathcal{H}(\mathcal{V})$ corrupted by an additive noise $n \in \mathcal{H}(\mathcal{V})$ such that $f^0 = h + n$. Regularizing functions on graphs using either isotropic or anisotropic $p$-laplacian, was proposed in [11, 13]. In this section we proposed to unify and to extend these both $p$-laplacian in a same operator.

To recover the uncorrupted function $h$, the processing task is to remove the noise $n$ from $f^0$. A commonly used method is to seek for a function $f \in \mathcal{H}(\mathcal{V})$, which is regular enough on $G$, and also close enough to $f^0$. This can be formalized by the minimization of an energy functional which involves a regularization term (or penalty term) plus an approximation one (or fitting term). In this paper, we consider the following model:

$$h \approx \underset{f:\mathcal{V}\to\mathbb{R}}{\arg\min}\ J(f) + \tfrac{\lambda}{2}||f - f^0||_2^2 \tag{21}$$

$$\text{where} \quad J^\phi_{w,p}(f) = \textstyle\sum_{v_i\in\mathcal{V}} \phi(||(\nabla_w f)(v_i)||_p) \tag{22}$$

is a gradient-based functional, and $\lambda \in \mathbb{R}$ is a regularization parameter, called Lagrange multiplier, that controls the trade-off between the penalty term and the fitting term. The function $\phi(\cdot)$ is a positive convex function that penalizes large variations of $f$ in the neighborhood of each vertex. Several penalty kernels have been proposed in literature, in different situations. Among them, we can quote $\phi(s) = s^2$ (known in the context of Tikhonov regularization [17]), $\phi(s) = s$ (total variation [18]), $\phi(s) = \sqrt{s^2+\epsilon^2} - \epsilon^2$ (regularized total variation [19]), and $\phi(s) = r^2 log(1 + s^2/r^2)$ (non linear diffusion [20]).

### B. Unified $p$-Laplace operator on weighted graphs

To get the solution of (22), we consider the following system of equations (Euler-Lagrange equation):

$$\frac{\partial J^\phi_{w,p}(f)}{\partial f(v_i)} + \lambda(f(v_i) - f^0(v_i)) = 0, \forall v_i \in \mathcal{V}, \tag{23}$$

where the first term denotes the variation of (22) with respect to $f$ at a vertex $v_i$. It is easy to show that this variation is, since this quantity depends only on $v_i$ and the edges incident to $v_i$, equal to:

$$\begin{aligned}\frac{\partial J^\phi_{w,p}(f)}{\partial f(v_i)} &\overset{(22)}{=} \frac{\partial \phi(||(\nabla_w f)(v_i)||_p)}{\partial f(v_i)} + \sum_{v_j\sim v_i} \frac{\partial \phi(||(\nabla_w f)(v_j)||_p)}{\partial f(v_i)} \\ &\overset{(8)}{=} \sum_{v_j\sim v_i} \alpha^{\phi,p,f}_{v_iv_j} |f(v_j) - f(v_i)|^{p-2}(f(v_i) - f(v_j)),\end{aligned} \tag{24}$$

with the following notation

$$\alpha^{\phi,p,f}_{v_iv_j} = w(v_i,v_j)^{p/2}\left(\frac{\phi'(||(\nabla_w f)(v_i)||_p)}{||(\nabla_w f)(v_i)||_p^{p-1}} + \frac{\phi'(||(\nabla_w f)(v_j)||_p)}{||(\nabla_w f)(v_j)||_p^{p-1}}\right). \tag{25}$$

This leads us to the proposition of the following general definition of a unified $p$-Laplace operator based on a divergence formulation:

$$(\Delta^\phi_{w,p} f)(v_i) \overset{def}{=} d^*_w\left(\frac{\phi'(||(\nabla_w f)(v_i)||_p)}{||(\nabla_w f)(v_i)||_p^{p-1}} |(d_w f)(v_i,v_j)|^{p-2}(d_w f)(v_i,v_j)\right)(v_i). \tag{26}$$

One can see that this expression (26) is exactly (24).

This formulation unifies the isotropic and anisotropic formulations of the $p$-Laplacian [11, 13] as well as many others under continuous, discrete, local or non-local formulations [21, 22, 23]. For instance, for $\phi(s) = s^q$ and $p = q$, one has $\Delta^\phi_{w,p} = 2q\Delta^a_{w,q}$. For $\phi(s) = s^q$ and $p = 2$, one has $\Delta^\phi_{w,2} = 2q\Delta^i_{w,q}$. More generally, for $\phi(s) = s^q$, one has

$$\frac{1}{q}\langle f, \Delta^\phi_{w,p} f\rangle_\mathcal{V} \overset{(5)}{=} J^\phi_{w,p}(f), \tag{27}$$

and this shows that the unified $p$-Laplacian operator is semi-definite positive.

In most cases (values of $p \neq 2$), the system (23) is nonlinear, and thus is is difficult to find a close solution. Approximated solutions are given in the following sections. Also, the regularization functional $J$ must be convex to ensure that the solution of (23) is also the solution of (22), which depends on $\phi$ and $p$.

### C. Diffusion processes

The first method, that is considered to get the solution of (23) is based on gradient descent of (23):

$$(\Phi f^n)(v_i) = (\Delta^\phi_{w,p} f)(v_i) + \lambda(f^0(v_i) - f^n(v_i)), \forall v_i \in \mathcal{V}, \tag{28}$$

with the initial condition $n = 0$ and $f^n = f^0$. A classical iterative algorithm to get the solution of Equation (28), at a time $n+1$, is the Euler one. An iteration of this algorithm is given by:

$$f^{n+1}(v_i) \quad = f^n(v_i) - \Delta t(\Phi f^n)(v_i), \quad \forall v_i \in \mathcal{V}, \tag{29}$$

where $f^n$ is the parametrization of $f$ by an artificial time $n > 0$. This describes a family of fitted diffusion flows on weighted graphs. This family includes and extends several well-known flows intensively used in image processing and computer graphics. Most of them are formulated without the fitting term ($\lambda = 0$), and have been analyzed by [24] in the context of image processing. In particular, for the regularization kernel $\phi(s) = s^2$ and $p = 2$, we obtain Laplacian-based diffusion, and if $\phi(s) = s$ it corresponds to mean curvature-based diffusion. For vertex filtering with meshes, we can also recover the diffusion equations for mesh fairing proposed in [25, 26] with specific edge weights.

### D. Neighborhood filters

This section describes a second approach to get the solution of (23). Let $\beta^{\phi,p,f}_{v_i,v_j} = \alpha^{\phi,p,f}_{v_i,v_j} |f(v_j) - f(v_i)|^{p-2}$. Equation (29) can be rewritten as:

$$f^{n+1}(v_i) = f^n(v_i) - \Delta t \left[ \sum_{v_j \sim v_i} \beta^{\phi,p,f}_{v_i,v_j} (f^n(v_i) - f^n(v_j)) + \lambda (f^n(v_i) - f^0(v_i)) \right] \quad (30)$$

For $\Delta t = \frac{1}{\lambda + \sum_{v_j \sim v_i} \beta^{\phi,p,f}_{v_i,v_j}}$, equation (30) becomes:

$$f^{n+1}(v_i) = \frac{\lambda f^0(v_i) + \sum_{v_j \sim v_i} \beta^{\phi,p,f}_{v_i,v_j} f^n(v_j)}{\lambda + \sum_{v_j \sim v_i} \beta^{\phi,p,f}_{v_i,v_j}}. \quad (31)$$

This describes a family of neighborhood filters. Indeed, at each iteration, the new value of $f^{n+1}$ at a vertex $v_i$, depends on two quantities: the initial value $f^0(v_i)$, and a weighted average of the filtered values of $f^n$ in the neighborhood of $v_i$.

### E. Related works

The choice of the resolution method (Equation (29) or (31)), of the regularization parameters ($p$, $\lambda$, $\phi$), and of the graph allows to retrieve and to extend several well-known filters proposed in the context of smoothing and denoising vertices coordinates of meshes or points clouds . With Equation (29), $\lambda = 0$, $p = 2$, and $\phi(s) = s^2$, many approaches of the literature can be related to ours with the use of specific edge weights $w(v_i, v_j)$. For meshes, we can quote the Bilateral Mesh filter [27], the Trilateral Mesh filter [28]. For point clouds, we can also recover recently proposed approaches using weights that account for local neighborhoods similarity [29, 30, 31]. To do so, filtered vertices coordinates are obtained by $\hat{\mathbf{p}}_i = \mathbf{p}_i + \delta_i \cdot \mathbf{n}_i$ where $\mathbf{p}_i$ denotes the coordinates of the vertex $v_i$, $\mathbf{n}_i$ is the normal to the point $\mathbf{p}_i$ and $\delta_i$ is an estimated displacement along the normal. Then, $\delta_i$ can be estimated using Equation (29) given an initial estimation of the displacement. With Equation (31), we can also recover diffusion equations for mesh fairing or smoothing proposed in [25, 26, 32].

One strong drawback of many of the above-mentioned methods is that methods conceived for meshes cannot process point clouds and vice-versa. Moreover, very few methods enable to simultaneously process vertices coordinates and colors as well as using patches to describe local configurations. Only our method fulfills all the requirements of a versatile and flexible framework that can process arbitrary vector spaces defined on meshes and point clouds.

## IV. Morphological Operators on Graphs

In this section we propose an adaptation on graphs of new PDEs morphological operators, that go beyond our previous works [12]. We show how such operators can be used to perform morphological filtering and inpainting processing on graphs.

### A. Morphological operators

Continuous-scale morphology [33] defines the flat dilation $\delta$ and erosion $\epsilon$ 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:

$$\frac{\partial f}{\partial t} = +||\nabla f||_p \text{ and } \frac{\partial f}{\partial t} = -||\nabla f||_p, \quad (32)$$

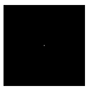
where $f$ is a modified function of $f^0$, $\nabla$ is the gradient operator, $||\cdot||_p$ corresponds to the $\mathcal{L}_p$-norm, and one has the initial condition $f = f^0$ at $t = 0$. With different values of $p$, one obtains different structuring elements: a rhombus for $p = \infty$, a disc for $p = 2$, and a square for $p = 1$ [33]. 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 [12] the discrete PdEs analogue of these PDEs-based dilation and erosion formulations by replacing $\nabla$ by $\nabla^{\pm}_w$ in (32). We propose a new formulation of morphological processes on graphs by considering the $q$-th power of the norm of the gradient $\nabla^{\pm}_w$. This provides the new following expression of dilation and erosion over graphs for a given initial function $f : \mathcal{V} \to \mathbb{R}$, $\forall v_i \in V$ and $0 < p < \infty$:

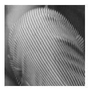
$$\begin{aligned} \frac{\partial f}{\partial t}(v_i, t) &= +||(\nabla^+_w f)(v_i)||^q_p \text{ , and} \\ \frac{\partial f}{\partial t}(v_i, t) &= -||(\nabla^-_w f)(v_i)||^q_p \, . \end{aligned} \quad (33)$$

We will mainly focus on the case of $p = q$ and $0 < p < \infty$ and $p = \infty$. When $q = 1$ we can recover our previous approach [12]. For the case of $p = \infty$, we consider:

$$\begin{aligned} \frac{\partial f}{\partial t}(v_i, t) &= +||(\nabla^+_w f)(v_i)||_\infty \text{ , and} \\ \frac{\partial f}{\partial t}(v_i, t) &= -||(\nabla^-_w f)(v_i)||_\infty \, . \end{aligned} \quad (34)$$

This new expression of PDEs-based morphological operators enables the introduction of adaptivity with the use of different configurations (graph weights, values of $p$ and $q$). Figure 1 shows the interest of such an adaptivity. An image of a pulse located on the center of a scalar grayscale image is considered. Results are shown for different graphs (local: 4-adjacency grid graphs, nonlocal: k-nearest-neighbor graph based on patch distances), and values of $p$ and $q$ that enable to control the shape of the structuring element when no weights are considered ($w = 1$). In the processing, the structuring element $B$ (supposed to be symmetric) is provided by the local neighborhood configurations and expressed by $B(v_i) = \{v_j \sim$


(a) Original Image of a white dot (a pulse) on a black background.


(b) Image used to compute the weights $w$.

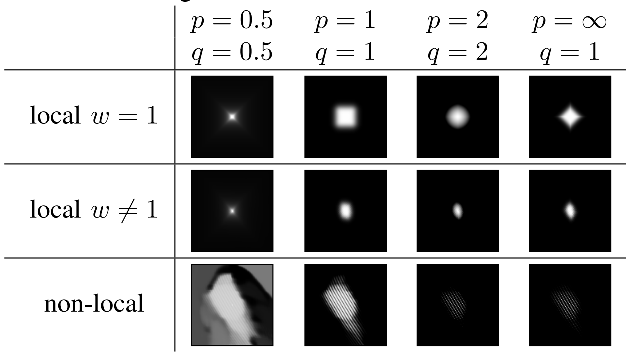


Figure 1: Dilation examples of a pulse image (a) using a weighted graph (weights computed from a textured image (b)). We set $p = q$ for $p < \infty$ and $q = 1$ for $p = \infty$.

$v_i\} \cup \{v_i\}$. When weights are considered, the dilation takes into account the image information and this adaptively modifies the dilation of the pulse. Local configuration consider similarities based on pixel features, whereas nonlocal ones consider patch-based similarities and this latter is more suited for dilations that better take into account fine and repetitive image structures. Figure 1 shows how the shape of the structuring element is deformed according to the chosen parameters.

## B. Morphological Filtering

In this section we provide an interpretation of the latter morphological operators as a family of non-local digital averaging filters that can be expressed by iterative schemes with discretization in time for filtering. We study the cases of $p = \infty$ and $q = p$.

*1) Case of $0 < p < +\infty$ and $q = p$:* the dilation process is now expressed by:

$$\frac{\partial f}{\partial t}(v_i, t) = +\|(\nabla_w^+ f)(v_i)\|_p^p. \tag{35}$$

With $v_j \underset{+}{\sim} v_i = \{v_j \sim v_i | f(v_j) \geq f(v_i)\}$, we have

$$f^{n+1}(v_i) = f^n(v_i) + \Delta t \sum_{v_j \sim v_i} w(v_i, v_j)^{p/2} \left((f^n(v_j) - f^n(v_i))^+\right)^p \tag{36}$$

$$= \left(1 - \Delta t \sum_{v_j \underset{+}{\sim} v_i} w(v_i, v_j)^{p/2} (f^n(v_j) - f^n(v_i))^{p-1}\right) f^n(v_i) + \Delta t \sum_{v_j \underset{+}{\sim} v_i} w(v_i, v_j)^{p/2} (f^n(v_j) - f^n(v_i))^{p-1} f^n(v_j). \tag{37}$$

Equation (37) shows a new family of pseudo-morphological diffusion filters based on dilation process, parameterized by the weight function $w$, parameter $p$, parameter $\Delta t$, and the graph $g$. For the special case of $\Delta t = \frac{1}{\sum_{v_j \underset{+}{\sim} v_i} w(v_i,v_j)^{p/2}(f^n(v_j) - f^n(v_i))^{p-1}}$, we get a new operator that we call the $NLD_p$ operator, expressed by:

$$NLD_p(f^n)(v_i) = \begin{cases} f^{n+1}(v_i) &= f^n(v_i) \quad \text{if } v_j \underset{+}{\sim} v_i = \emptyset \\ f^{n+1}(v_i) &= \dfrac{\sum_{v_j \underset{+}{\sim} v_i} \alpha_{w,p}^{f^n}(v_i, v_j) f^n(v_j)}{\sum_{v_j \underset{+}{\sim} v_i} \alpha_{w,p}^{f^n}(v_i, v_j)} \quad \text{o.w.,} \end{cases} \tag{38}$$

where

$$\alpha_{w,p}^f(v_i, v_j) = w(v_i, v_j)^{p/2} (f(v_j) - f(v_i))^{p-1}$$

and $f^{n+1}(v_i)$ depends of neighbors values $f(v_j)$. By applying the same approach, we can get new filters based on on erosion process, that we call $NLE_p$. These operators define a new family of pseudo-morphologic diffusion filters, that globally behave like an erosion or a dilation process, but also average informations from their neighbors.

*2) Case of $p = \infty$:* the dilation process is then expressed by:

$$f^{n+1}(v_i) = f^n(v_i) + \Delta t \max_{v_j \sim v_i} \left(\sqrt{w(v_i, v_j)} (f^n(v_j) - f^n(v_i))^+\right) \tag{39}$$

To simplify this expression, we consider the vertex $v_d$ such that

$$v_d = \arg\max_{v_j \underset{+}{\sim} v_i} \left(\sqrt{w(v_i, v_j)} (f^n(v_j) - f^n(v_i))\right).$$

With this notation, equation (39) can be rewritten as:

$$f^{n+1}(v_i) = \left(1 - \Delta t \sqrt{w(v_i, v_d)}\right) f^n(v_i) + \Delta t \sqrt{w(v_i, v_d)} f^n(v_d). \tag{40}$$

For the special case of $\Delta t = \frac{1}{\sqrt{w(v_i, v_d)}}$, the dilation PdE can be interpreted as an iterative non-local dilation (NLD) process and expressed by

$$\begin{aligned} f^{n+1}(v_i) &= NLD(f^n)(v_i) \\ &= f^n(v_i) + \|(\nabla_w^+ f^n)(v_i)\|_\infty \\ &= f^n(v_d). \end{aligned} \tag{41}$$

One can easily see that when $w(v_i, v_j) = 1$ one recovers the classical algebraic formulation of the mathematical morphology dilation. Our proposal enables therefore to define new general weighted dilation operators. With the same approach, we can define a non-local erosion operator that we will denote by $NLE$.

## C. Morphological Inpainting

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 $\infty$-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 [1] and references therein for more details). We presented a unifying approach of local geometric methods and non-local exemplar-based ones for inpainting [34] using the framework of discrete non-local regularization on graphs introduced in [11]. 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 $\mathcal{A} \subset \mathcal{V}$ be the subset of vertices with unknown values and $\partial\mathcal{A}$ 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:

$$\begin{cases} (\Delta_{w,\infty} f)(v_i) = 0 & \forall u \in \mathcal{A}, \\ f(u) = f^0(v_i) & \forall u \in \partial\mathcal{A}. \end{cases} \tag{42}$$

The solution $f^*$ is said to be infinity-harmonic [35]. Works of [16] have shown that interpolation problems have a unique solution that can be obtained using the following iterative algorithm:

$$\begin{cases} f^{(0)}(v_i) = f^0(v_i), \\ f^{(n+1)}(v_i) = NLA(f^{(n)})(v_i). \end{cases} \tag{43}$$

The solution is obtained with this simple iterative algorithm based on the NLA operator, where:

$$NLA(f)(v_i) = \frac{1}{2}[NLD(f)(v_i) + NLE(f)(v_i)]. \tag{44}$$

The iterative algorithm presented in (43) converges to the unique solution [16]. 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. At each iteration, only the internal boundary $\partial^-\mathcal{A} = \{v_i \in \mathcal{A} | \exists v_j \sim v_i, v_j \in \partial\mathcal{A}\}$ is inpainted:

$$\begin{cases} NLA(f)(v_i) = \frac{1}{2}[NLD(f)(v_i) + NLE(E)(v_i)] & \forall v_i \in \partial^-\mathcal{A}, \\ NLA(f)(v_i) = f^0(v_i) & \forall v_i \in \partial\mathcal{A}. \end{cases} \tag{45}$$

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

## V. Segmentation of graph-signals

In this section, we propose a framework for the segmentation of graph signals (functions defined on the vertices of graphs). To perform the segmentation, we consider a convex formulation of active contours on graphs. Starting from a continuous formulation, we show how to transpose the latter on weighted graphs using the framework of PdEs along with a minimization strategy

The usual drawback of active contours methods is the existence of local minimizers and hence their sensitivity to the initial condition. A recent method, introduced by Bresson and Chan in [36, 37], proposes to redefine the active contour model into a model which gives global minimizers. In the continuous setting, where images are viewed as functions on a continuous domain $\Omega$, this model is given by:

$$\underset{f(x)\in\{0,1\}}{\arg\min} \left\{ \int_\Omega ||\nabla f(x)||_1 dx + \lambda \int_\Omega g(f^0)(x) f(x) dx \right\}. \tag{46}$$

The transposition of (46) on graphs is obtained using the PdEs framework [11, 14] leading to:

$$\bar{f} \in \underset{f:\mathcal{V}\to\{0,1\}}{\arg\min} \left\{ \sum_{v_i\in\mathcal{V}} \|(\nabla_w f)(v_i)\|_p^p + \lambda \sum_{v_i\in\mathcal{V}} g(f^0)(v_i) f(v_i) \right\}, \tag{47}$$

where $f$ is a labeling function and $f^0$ the signal on the graph. When $\lambda \neq 0$, this energy can be considered as the non-local discrete analogue on graphs of the functional introduced in [37]. We now show how such a minimization can be solved. Problem (47) is non-convex and, as shown in [38] for the continuous analogue, can be reformulated through a convex relaxation. Therefore, a new minimization problem is considered:

$$\hat{f} = \underset{f:\mathcal{V}\to[0,1]}{\arg\min} \left\{ \sum_{v_i\in\mathcal{V}} \|(\nabla_w f)(v_i)\|_p^p + \lambda \sum_{v_i\in\mathcal{V}} g(f^0)(v_i) f(v_i) \right\}. \tag{48}$$

Following the approach in [38], one can show that every level-set of a minimizer of (48) is solution of the original optimization problem (47). As a consequence, to obtain a global solution $\bar{f} : \mathcal{V} \to \{0,1\}$ to the problem of (47), one thresholds any function $\hat{f} : \mathcal{V} \to [0,1]$ that is a solution of (48) and $\bar{f}(v_i) = \chi_\mathcal{S}(v_i)$, where $\mathcal{S} = \{v_i \in \mathcal{V} : \hat{f}(v_i) > t\}$ with $t \in [0,1]$ and $\chi$ is the indicator function defined by $\chi : \mathcal{V} \to \{0,1\}$. For a given vertex, if $v_i \in \mathcal{A}$, then $\chi_\mathcal{A}(v_i) = 1$ and $\chi_\mathcal{A}(v_i) = 0$ otherwise. However, to be able to perform such a minimization approach, one has to show that both parts of the energy (48) do verify the co-area formula. This can be easily shown for the second part of the energy (see [38]). We show now that this is also true for the first part.

*1) **Perimeters and co-area on graphs**:* Now we show that there is a relation, for the case of a sub-graph, between discrete perimeters on graphs and the co-area formula on graphs.

*a) Perimeters on graphs:* Let $\mathcal{A}$ be a set of connected vertices with $\mathcal{A} \subset \mathcal{V}$. We denote by $\partial^+\mathcal{A}$ and $\partial^-\mathcal{A}$, the *external* and *internal* boundary sets of $\mathcal{A}$, respectively. The set $\mathcal{A}^c{=}\mathcal{V} \setminus \mathcal{A}$ is the complement of $\mathcal{A}$. For a given vertex $v_i{\in}\mathcal{V}$, one has: $\partial^+\mathcal{A} = \{v_i{\in}\mathcal{A}^c : \exists v_j{\in}\mathcal{A} \text{ with } (v_i, v_j){\in}\mathcal{E}\}$, $\partial^-\mathcal{A} = \{v_i{\in}\mathcal{A} : \exists v_j{\in}\mathcal{A}^c \text{ with } (v_i, v_j){\in}\mathcal{E}\}$, and $\partial\mathcal{A} = \{(v_i, v_j) \in \mathcal{E} : \exists v_i{\in}\partial^+\mathcal{A} \text{ and } v_j{\in}\partial^-\mathcal{A}\}$. Let us consider non-local regularization functionals based on weighted total variation on graphs $R_{w,p} : \mathcal{H}(V) \to \mathbb{R}$ of a function $f \in \mathcal{H}(V)$: $R_{w,p}(f) = \sum_{v_i\in\mathcal{V}} \|(\nabla_w f)(v_i)\|_p^p$ with $0 < p < +\infty$.

*b) Co-area formula on graphs:* In this subsection, we discuss the co-area formula on graphs. It is very useful in many contexts such as convex relaxation of variational methods on graphs. Let $(\mathcal{V}, \mathcal{E}, w)$ be a weighted graph, $f \in \mathcal{H}(\mathcal{V})$. For $t \in \mathbb{R}$, let $\mathcal{A}_t = \{v_i \in \mathcal{V} : f(v_i) > t\}$. Then the co-area formula is verified for $p = 1$ since $Per_{w,1}(\mathcal{A}) = \int_{-\infty}^{\infty} Per_{w,1}(\mathcal{A}_t) dt$. The proof is obvious since $|a - b| = \int_{-\infty}^{+\infty} |\chi_{\{a>t\}} - \chi_{\{b>t\}}| dt$. In the rest of the paper, we will therefore work exclusively with the case of $p = 1$ since only $R_{w,1}$ does verify the co-area formula.

*2) **Minimization Algorithm on Weighted Graphs**:* To solve the optimization problem (48), we propose to use the Chambolle and Pock algorithm [15] on weighted graphs, in a similar manner as in [39]. Let us consider the following general optimization problem:

$$\min_{x \in X} F(Kx) + G(x), \tag{49}$$

where $X$ and $Y$, are two general finite-dimensional vector spaces, and $F \in \Gamma_0(Y)$, $G \in \Gamma_0(X)$ and $K : X \to Y$ a linear operator. Recently, Chambolle and Pock have proposed the following iterative algorithm [15] to solve efficiently (49):

$$\begin{cases} x^0 = \bar{x}^0 = f, \quad y^0 = 0 \\ y^{n+1} = \text{prox}_{\sigma F^*}(y^n + \sigma K \bar{x}^n), \\ x^{n+1} = \text{prox}_{\tau G}(x^n - \tau K^* y^{n+1}), \\ \bar{x}^{n+1} = x^{n+1} + \theta(x^{n+1} - x^n), \end{cases} \tag{50}$$

where $F^*$ is the conjugate of $F$ [40], $K^*$ is the adjoint operator of $K$, and prox the proximity operator defined as:

$$\text{prox}_f(x) = \arg\min_{y \in Y} \left\{ f(y) + \frac{1}{2}||y - x||^2 \right\}. \tag{51}$$

The convergence of algorithm (50) is guaranteed if $\theta = 1$ and $0 < \tau\sigma L^2 < 1$ where $L = ||K|| = \max_{||x|| \leq 1} ||Kx||$. The segmentation problem of (48) is formulated with $F = ||.||_1$, $K = \nabla_w$ and $G = \lambda\langle ., g(f^0)\rangle$, with $\langle ., . \rangle$ the dot product operator. By replacing $F$, $K$ and $G$, in (50), we can simplify the algorithm. For $y \in Y$, as shown in [39], we have:

$$\text{prox}_{\sigma F^*}(y) = \text{prox}_{\sigma i_B}(y) = \text{proj}_B(y) = \tilde{y},$$

where $\tilde{y}_{ij} = M(y_{ij}) = \frac{y_{ij}}{\max(1, \max(y_{i.}))}$ and

$$i_C = \begin{cases} 0 & \text{for } x \in C \\ +\infty & \text{otherwise}, \end{cases} \tag{52}$$

and $B$ is the unitary $||.||_{\infty,2}$ ball.

For $x \in X$, we can show that:

$$\begin{aligned} \text{prox}_{\tau G}(x) &= \arg\min_{y \in Y} \left\{ \tau\lambda\langle y, g(f^0)\rangle + \frac{1}{2}||y - x||^2 \right\} \\ &= x - \tau\lambda g(f^0). \end{aligned} \tag{53}$$

Thus the algorithm to solve the segmentation problem (48) is reduced to:

$$\begin{cases} x^0 = \bar{x}^0 = f, \quad y^0 = 0, \\ y_{ij}^{n+1} = M(y_{ij}^n + \sigma(d_w \bar{x}^n)(v_i, v_j)), \\ x_i^{n+1} = x_i^n - \tau(d_w^* y^{n+1})(v_i) - \tau\lambda g(f^0)(v_i), \\ \bar{x}_i^{n+1} = x_i^{n+1} + \theta(x_i^{n+1} - x_i^n). \end{cases} \tag{54}$$

This algorithm is parameterized by the structure of the graph (topology and weight function $w$), the functions $f$, $f^0$ and $g(f^0)$, and several parameters ($\lambda$, $\tau$, $\theta$ and $\sigma$). One has to note that it is the first time that such a solution is proposed to solve (48) on general weighted graphs.

## VI. Graph Construction

In this section, we discuss how to build a weighted graph from surfaces data or point clouds. Roughly speaking, two types of graph are considered: local and non-local graphs. The creation of a graph lies on several steps: first vertices are created from raw data to process, second vertices are connected with edges, and finally weights associated to each edge are deduced. The creation of a local graph is really straightforward: only local close neighbors are considered during the creation of edges. In a non-local graph, edges are created between vertices that are spatially far apart. Weights on each edge are deduced from values associated to vertices (the graph signals) and patches can be used to compute a better similarity value accounting local neighborhood similarities.To do so, we propose a new patch definition subdivided in 2 steps: patch localization and patch creation.

### A. Graph creation

Depending on the data under consideration (meshes or point clouds), we devise specific strategies for the creation of the graph, i.e., inferring the set of vertices and edges.

*1) Graph Creation from Meshes:* Graph creation from meshes is straightforward since the set of vertices and edges is known beforehand. Let us consider the meshing $M$ of a surface as a set of vertices $M_V = \{\mathbf{v}_1, \ldots, \mathbf{v}_n\} \in \mathbb{R}^3$ and a set of edges $M_E \subset M_V \times M_V$. The graph $\mathcal{G}$ is therefore defined with $\mathcal{V} = M_V$ and $\mathcal{E} = M_E$. The topology of this graph is local but can be modified arbitrarily to become nonlocal by adding edges between vertices not belonging to adjacent triangles.

*2) Graph Creation from Point Clouds:* Graph creation from point clouds is much more challenging. Indeed, the structuring information is lacking and the data is not naturally organized in a manifold. Therefore, the set of edges cannot be easily determined. Given a point cloud $P$ defined 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 to such a data set. Since point clouds data exhibit a geometrical structure, proximity graphs are preferable: if two data points satisfy particular geometric requirements, the corresponding vertices in the graph are connected by an edge. 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. Among many possible choices, we choose to consider the symmetric $k$ Nearest Neighbor Graph ($k$-NNG). An undirected edge $(v_i, v_j)$ is added between two vertices $v_i$ and $v_j$ if the distance between $\mathbf{p}_i$ and $\mathbf{p}_j$ is among the $k$ smallest distances from either $\mathbf{p}_i$ or $\mathbf{p}_j$ to all the other data points. The value of $k$ will be denoted $k_\mathcal{G}$ for the graph $\mathcal{G}$ associated with the point cloud. The construction of such a graph being computationally expensive for large point clouds, a kD-tree is used [41] to speed-up the $k$ nearest neighbor search.

### B. Graph Weights

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 simply set up to $w(v_i, v_j) = 1$, $\forall (v_i, v_j) \in \mathcal{E}$. To take into account the similarity between the graph signal associated to the vertices, it is more suited to use similarity functions based on distances to weight the edges.

Given an initial function $\mathbf{f}^0 : \mathcal{V} \to \mathbb{R}^m$, computing distances between vertices consists in comparing their features generally depending upon $\mathbf{f}^0$. To this end, each vertex $v_i$ is associated to a feature vector $\boldsymbol{\mathcal{P}}(v_i) \in \mathbb{R}^q$. From this, a usual similarity measure is provided by the Gaussian kernel:

$$w(v_i, v_j) = exp\left(-\frac{||\boldsymbol{\mathcal{P}}(v_i) - \boldsymbol{\mathcal{P}}(v_j)||_2^2}{\sigma^2}\right). \tag{55}$$

Traditionally, one has simply $\boldsymbol{\mathcal{P}}(v_i) = \mathbf{f}^0(v_i)$. However, in image processing an important feature vector is provided by image patches [42]. 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 $\boldsymbol{\mathcal{P}}(v_i) = \left(f^0(v_j) : v_j \in B(v_i, n)\right)^T$ where $B(v_i, n)$ is a square of size $n^2$ centered at $v_i$. This definition of patches is valid only for grid-graphs and cannot be considered for arbitrary graphs. Therefore, we need a new definition of patches that can be used with any graph representation associated to meshes or point clouds.

### C. Patch Definition

As we just mentioned, there is actually no proper definition of patches for graph signals on arbitrary graphs. Some authors have proposed definitions of patches that account for local neighborhood configurations [30, 31], but these methods are not valid for any graph signal. Indeed, they only consider the case of vertex coordinates and their proposal cannot be easily adapted for other graph signals such as vertices colors. Therefore, we propose to introduce a definition of graph-signal patches for arbitrary graphs. To do so, around each vertex we build a two-dimensional grid (the patch) describing the neighborhood. This grid is defined on the tangent plane of the point (i.e., the vertex). Since there are many different ways of orienting a 2D grid on a plane, we devise a strategy to define the orientation of the patch. Once the orientation is known, the patch is created by a weighted average of the graph-signal values in the local neighborhood. We detail these two steps in the sequel.

*1) Orientation:* The first 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 [43], 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 [43]). Figure 2 illustrates this where the $k_n$ nearest neighbors are comprised in a ball of radius $\epsilon$.

Next, patches need to be 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 $\mathbf{p}_i$, one can use the arguments of [30]: principal directions can be estimated as the eigenvectors of a PCA of the covariance matrix of the normals of the neighbors of $\mathbf{p}_i$.

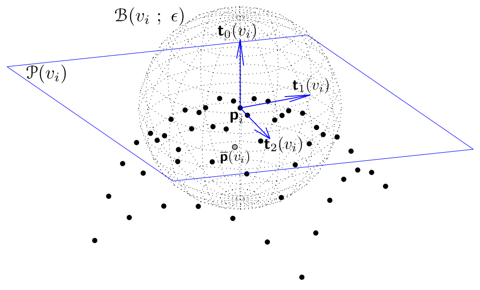


Figure 2: The tangent plane to a point $\mathbf{p}_i$.

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. Unfortunately, because the obtained orientation depends highly on the most predominant axis, one can find different patches orientations for similar 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 $\mathbf{t}_1(v_i)$ from the normal vector $\mathbf{n}(v_i)$. As in [30], we use a PCA to estimate this normal vector: $\mathbf{t}_0(v_i) = \mathbf{n}(v_i)$.

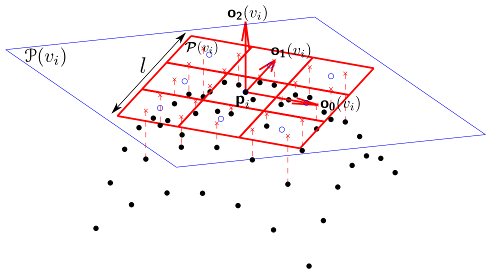
Let $\mathbf{x}, \mathbf{y}, \mathbf{z}$ be the three axis of a 3D space, the first tangent vector $\mathbf{t}_1(v_i)$ is computed with:

$$\begin{cases} \mathbf{t}_1(v_i) = & \mathbf{z} \times \mathbf{t}_0(v_i) \text{ if } |\mathbf{z} \cdot \mathbf{t}_0(v_i)| \neq 1 \\ \mathbf{t}_1(v_i) = & \mathbf{x} \times \mathbf{t}_0(v_i) \text{ otherwise,} \end{cases} \tag{56}$$

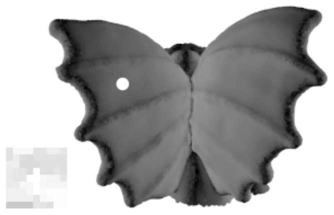
with $\times$ the cross product operator, and $\cdot$ the dot product operator. Then a bitangent vector $\mathbf{t}_2(v_i)$ is computed by $\mathbf{t}_2(v_i) = \mathbf{t}_0(v_i) \times \mathbf{t}_1(v_i)$. The orientations vectors $\mathbf{o}_0(v_i), \mathbf{o}_1(v_i), \mathbf{o}_2(v_i)$ are then respectively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

*2) Patch construction:* Second 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 in the orientation of the patch defined by $(\mathbf{o}_0, \mathbf{o}_1)$. 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 projected on to the patch cell. This value is a spectral value (the points' colors). The set of values inside the patch of the vertex $v_i$

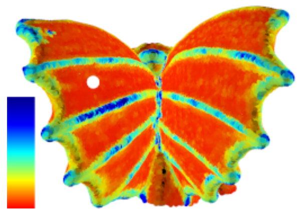
are denoted as $\boldsymbol{\mathcal{P}}(v_i)$. Let $C_k(v_i)$ denotes the $k$th cell of the constructed patch around $v_i$ with $k \in [1, n^2]$. With the proposed patch construction process, one can define the set $V_k(v_i) = \{v_j \mid \mathbf{p}'_j \in C_k(v_i)\}$ as the set of vertices $v_j$ that were affected to the $k$th patch cell of $v_i$. Then, the patch vector is defined as $\boldsymbol{\mathcal{P}}(v_i) = \left( \frac{\sum_{v_j \in V_k(v_i)} w(\mathbf{c}_k, \mathbf{p}_j) f^0(v_j)}{\sum_{v_j \in V_k(v_i)} w(\mathbf{c}_k, \mathbf{p}_j)} \right)^T_{k \in [1, n^2]}$, with $w(\mathbf{c}_k, \mathbf{p}_i) = exp(-\frac{\|\mathbf{c}_k - \mathbf{p}'_i\|_2^2}{\sigma^2})$ where the $\mathbf{c_k}$ are the coordinates' vector of the $k$th patch cell center. This weighting enables to take into account the repartition of the points in the cells of the patch to compute their mean feature vectors. Figure 3(a) summarizes the method of patch construction. Figures 3(b) and 3(c) shows that points with similar geometric configurations are close with respect to the patch distance.



(a)



(b)



(c)

Figure 3: Figure (a) shows the interpolation of the content of the patch. $l$ is the patch length. $\mathbf{o}_1(v_i)$ and $\mathbf{o}_2(v_i)$ are the orientation of the patch $\boldsymbol{\mathcal{P}}(v_i)$ at a point $\mathbf{p}_i$. Elements marked by a "×" symbol correspond to the projected neighbors $\mathbf{p}'_i$ of the point $\mathbf{p}_i$ on the patch. These projections are used to deduce values of each patch cell (a "o" symbol) by a weighted average of the associated graph signal values. Figure (b) shows a point cloud with a selected vertex (in white), and the patch descriptor of that vertex. Figure (c) shows a point cloud colored by the patch-based distance between all points and a given selected one, from most similar (in red) to least similar (in blue).

## VII. Experiments

In this section we illustrate the abilities of the proposed methods and algorithms for signal processing on meshes and point clouds. The typical graph signals we consider are point (resp. vertices) coordinates or colors. Given a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ associated to a mesh (typically a triangular one) or to a point cloud, we will consider an initial graph signal $\mathbf{f}^0 : \mathcal{V} \to \mathbb{R}^m$ with $m = 3$. This signal will be either the vertices coordinates, in this case $\mathbf{f}^0(v_i) = \mathbf{p}_i$, or vertices colors, in this case $\mathbf{f}^0(v_i) = (f^R(v_i), f^G(v_i), f^B(v_i)^T)$ where $f^X(v_i)$ denotes the $X$ color component of the color at the vertex $v_i$. We will mainly consider the last case (the processing of the color associated to vertices) but to be inline with classical mesh processing we will also present some filtering results for vertices coordinates.

### A. *p-Laplacian filtering*

First we provide illustrations of $p$-Laplacian filtering for meshes and point clouds. We will consider Equation (31) for the filtering unless specified otherwise. We first consider a triangular mesh of 200000 points with $\mathbf{f}^0(v_i) = \mathbf{p}_i$. The processing is local and uses Gaussian weights. The central column of Figure 5 presents, for given parameters, how the value of $p$ affects the result of the processing. One important

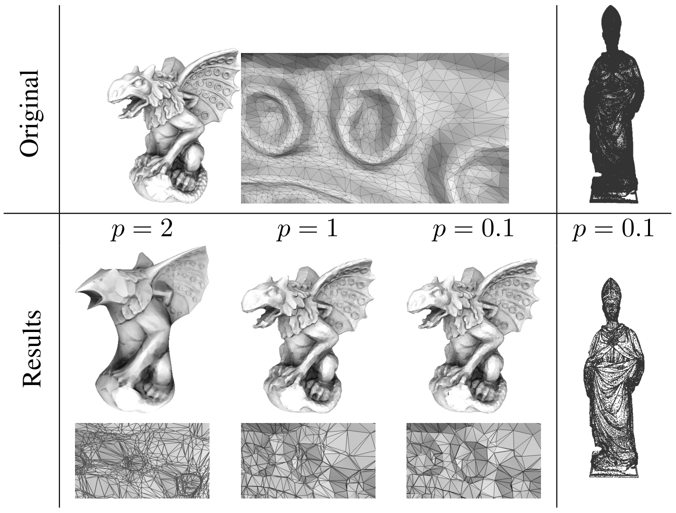


Figure 5: In the central column: filtering of the mesh of a gargoyle, using $\phi(s) = s^2$, $\lambda = 0$, 1000 iterations with various values of $p$, for each of the depicted results, the filtered mesh and a zoom on a specific part are shown. In the right column: simplification of a point cloud of the Saint Eligius statue with $p = 0.1, \lambda = 0, \phi(s) = s^p$ after 10000 iterations.

thing to note here is that the presented graphs all have the same number of vertices. The filtering enables to groups similar vertices around high curvature regions. The second thing to point out is that acting on $p$ enables to perform a filtering of the original graph but also to smooth (for $p \geq 1$) or to preserve sharp curvatures (for $p \leq 1$). This can be particularly interesting when used on point clouds. The right column of Figure 5 presents the result of the filtering of a point cloud of 200000 points with $\mathbf{f}^0(v_i) = \mathbf{p}_i$ and a small value of $p$.

The processing is local (the graph is a 8 nearest neighbor graph) and uses Gaussian weights. This time the effect is easily visible: the filtering has grouped the point around areas of high curvature, enabling a better visualization of the point cloud.

If our approach can be used to filter the vertices coordinates, it can be also used to process any graph signal associated to graphs. To illustrate this, we now consider a color point cloud, where a color is associated to each vertex. From this original 3D colored point cloud, we artificially add Gaussian noise and denoise the color at each vertex with $p = 2$ and

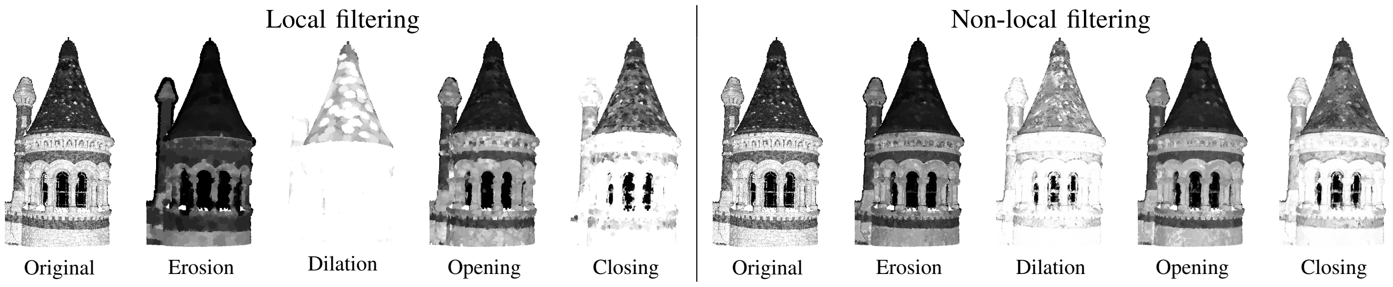


Figure 4: Morphological operators on a colored point cloud (with 219,699 points) after 10 iterations with $p = \infty$, $q = 1$, $k_G = 1000$, and $n^2 = 25$. First column present local filtering: $w = 1$ and $k_G = 8$. Second column presents non-local filtering: $w$ is a similarity measure between patches.

$\lambda = 10^{-4}$ (Figure 6). We consider two different types of graphs. The first one is a local graph (8 nearest neighbor graph) with constant weights ($w(v_i, v_j) = 1$). The second one is a non-local graph (1000 nearest neighbor graph) with patch-based Gaussian weights (using $n^2 = 5 \times 5$ patches defined with the 150 nearest points for each point). The benefit of a non-local approach is evident and much better denoising results are obtained with edges and texture that are preserved. This visual assessment can be quantitatively measured with the PSNR value that is equal to 23.89dB with the non-local approach versus 22.71dB with the local one. This shows the real benefit of the use of a non-local graph with patch-based weights. Figure 6 presents all the results.

### B. Mathematical Morphology

Second we provide illustrations of Morphological filtering and inpainting for meshes and point clouds. We start by providing examples of Morphological filtering of point clouds with $p = \infty$ and $q = 1$ with Equation (39) for the dilation (and associated scheme for the other morphological operators). A 3D colored point cloud is considered and we use the same graph configurations than in Figure 6. Figure 4 shows some morphological processings of a tower of the bishop castle point cloud obtained with an Optech Lidar scanner. As it was previously mentioned, with non constant weights, an adaptive morphological processing is obtained that can better preserve some features of the graph signal, depending on the graph weights. In this example, patch-based weights are considered and therefore repetitive patterns are better preserved while providing the usual attended simplification effects of morphological operators. Figure 7 presents additional morphological filtering results using the proposed $NLE$ and $NLA$ morphological averaging operators with $p = \infty$. The same graphs as in Figure 6 are considered. Again, the interest of patch-based graph weights with non-local graphs is put forward. As this is interesting for filtering, this is all the more interesting for inpainting. To illustrate this, we use the $NLA$ operator to perform inpainting using the approach we proposed in Equation (45).

Figure 8 shows the morphological inpainting results we obtained on a real 3D color point cloud. The point cloud is a real scan of a man and the image that appears on his T-shirt has to be removed. Local and non-local configurations

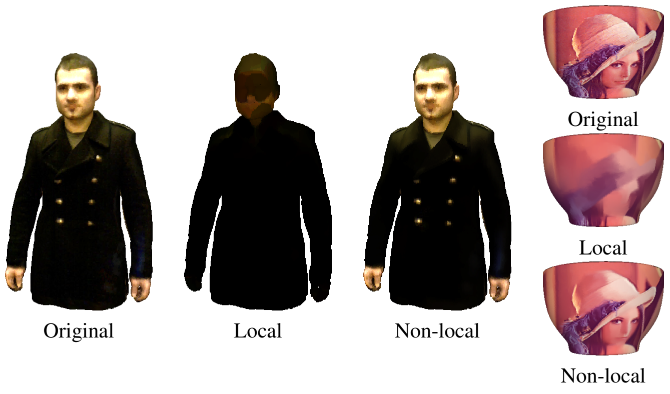


Figure 7: Processing of the colors of a real 3D point cloud of a man with the $NLE$ operator. In the last column: processing of the colors of a generated 3D point cloud (lena on the point cloud of a bowl) with the NLA operator. Results are shown after 600 iterations.

of the $NLA$ operator are used towards this. For non-local inpainting, we set $k_G = 30000$ and $n^2 = 9 \times 9$. Regarding the previous experiments, we have modified the values of these parameters since their choice depends on the point sampling density as well as the area to be recovered by inpainting. As it can be seen, the local morphological inpainting cannot restore correctly the texture of the missing part of the T-shirt. In contrast, the non-local approach was successful in performing this interpolation thanks to patch-based graph weights with non-local graph associated to the $NLA$ operator.

### C. Geometry and Color Inpainting on Point Clouds

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

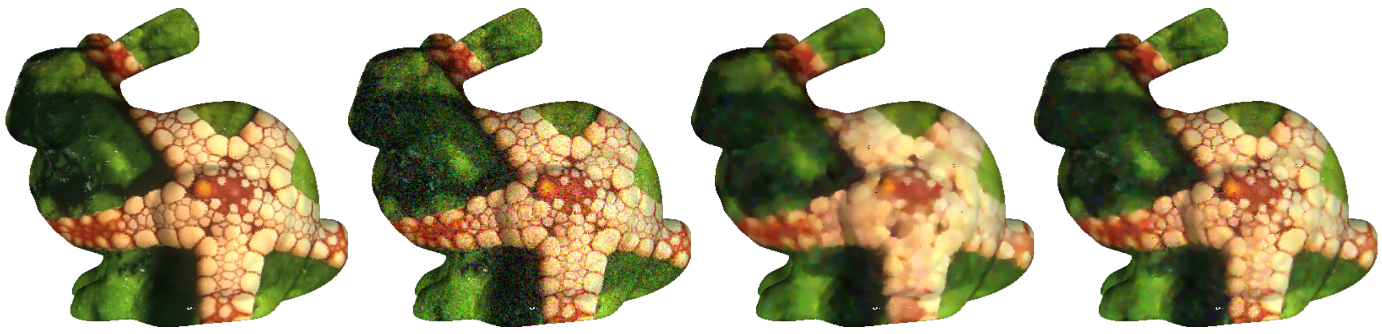

(a) Original (b) Noisy (c) Local (d) Non-local

Figure 6: Denoising of a noisy point cloud corrupted with Gaussian noise of standard deviation $\sigma = 30$. Results are shown after 100 iterations.

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 $||\mathbf{p}_i - \mathbf{p}'_i||_2$ (as in the approaches of [30, 31]). Then the algorithm presented in (43) 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. 9 shows the full process of virtual restoration of an antique broken vase. First the hole is filled with points to cover the missing part, next the color of the geometrically recovered part is inpainted with the algorithm present in Section III.

### *D. Segmentation of graph signals*

To end these experiments, we provide illustrations of the segmentation of graph signals. This section shows some segmentation results of colored 3D point clouds in two classes using the algorithm presented in (54). The raw data consist of sets of 3D points $\mathbf{p}_i$ (i.e., vertices $v_i$) that are associated with CIELAB colors vectors ($f^0 : \mathcal{V} \to \mathbb{R}^3$). In all our experiments, the parameters are $\theta = 1$, $\tau = 1$, $\sigma = 0.25/(w_{max} \times \tau)$ with $w_{max} = \max_{v_i \in \mathcal{V}} \delta_w(v_i)$. Figure 10 shows segmentation result on real example with $g(f^0(v_i)) = (\bar{c}_1 - f^0(v_i))^2 - (\bar{c}_2 - f^0(v_i))^2$, where $\bar{c}_1$ and $\bar{c}_2$ are respectively the average colors of both extracted regions. This corresponds to the Chan-Vese model on graphs. The initial partition function $f$ is initialized from seeds provided by the user. Figure 10 shows the segmentation of real scanned persons. For the first scan, the graph is a local one and the weights are based on the colors of the vertices. This explain why both the short and the hair of the model are extracted in the same class. The second scan is much more challenging. In this case, we want to extract the T-shirt of the man. This is very difficult because of the texture and heterogeneity of the latter. To obtain a good segmentation, we have therefore to rely on a non-local patch-based weighted graph ($k_G = 1000$ and $n^2 = 5 \times 5$). Since the T-shirt is a very heterogeneous region in color, we have used the variance of patches to compute a heterogeneous term that eases the segmentation. We set $g(f^0(v_i)) = (\overline{Var}_1 - f^0(v_i))^2 - (\overline{Var}_2 - f^0(v_i))^2$ where $f^0 : \mathcal{V} \to \mathbb{R}^3$ represents the variance of $\boldsymbol{\mathcal{P}}(v_i)$ for each color channel, and $\overline{Var}_1$, $\overline{Var}_2$ are respectively the average variance of patches of extracted regions for each color channel. This shows the versatility of the proposed approach. To the best of our knowledge, there exist actually no other method that is able to provide such a segmentation on point clouds that exhibit repetitive patterns.

## VIII. Conclusion

In this paper we have proposed an approach for the processing of functions on surfaces or point clouds represented as graphs. From this graph representation, we have redefined the basic differential operators on graphs, that mimic the continuous ones. From these operators, we have shown how to transpose on graphs some PDEs using the framework of Partial difference Equations (PdEs). We have used this latter approach to transcribe, for surfaces and point clouds, many models and algorithms designed for image processing. To illustrate our proposal, three problems have been considered: $p$-Laplacian filtering , morphological operators, and segmentation. Through experiments, we have shown the potentialities and the flexibility of our approach to address these various problems. From a new definition of a patch for 3D point clouds, we have also shown the efficiency and the superiority of nonlocal patch-based schemes as compared to local ones.

**Abderrahim Elmoataz** is a full-time Professor of computer science in the Computer Science Department, Université de Caen Basse-Normandie, France. His research concerns PDEs on graphs with applications in image processing and machine learning.

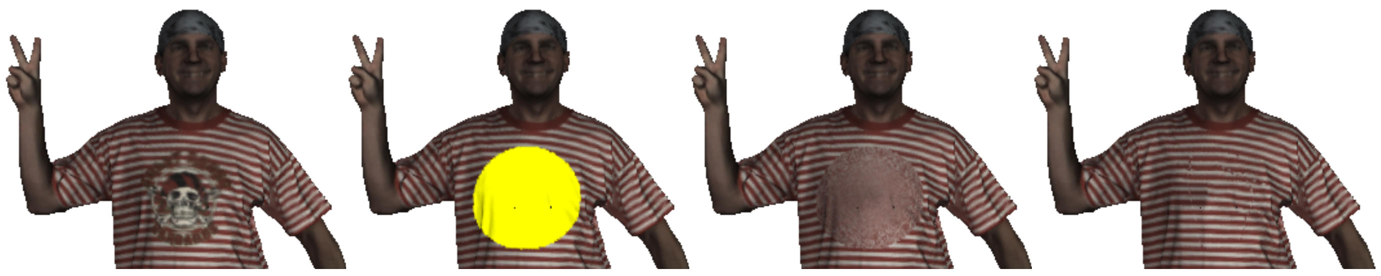

(a) original (b) to inpaint (c) with $w = 1$ (d) with patches

Figure 8: Morphological inpainting of a real scan (127,039 points). From left to right respectively: the original man, the part to be inpainted shown in yellow, the inpainted man with local and non-local approaches.

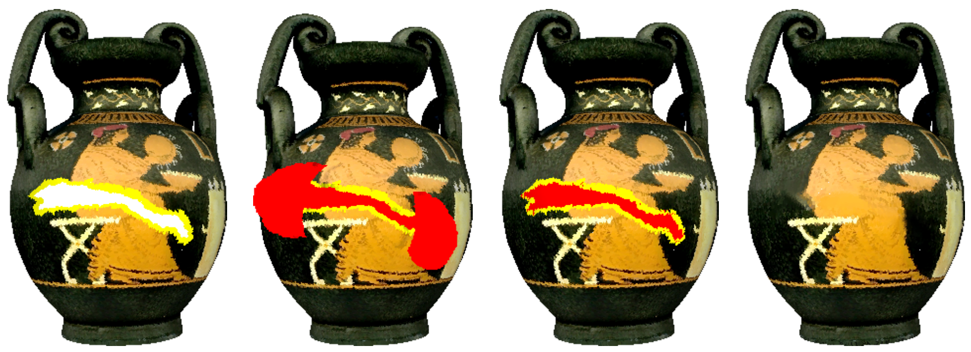

(a) (b) (c) (d)

Figure 9: Virtual restoration of a broken antique vase (with 220,994 points) with $k_G = 4000$ and $n^2 = 81$. (a) broken vase with labeled part to restore (border shown in yellow), (b) points' sampling on the tangent plane, (c) geometric filtering result, (d) color inpainting result

**François Lozes** received the M.Sc. degree in computer science from the Université de Caen Basse-Normandie, France in 2011. Since 2011 he is a PhD student in computer science with the School of Engineers of Caen (ENSICAEN) and Université of Caen, France. His research mainly concerns the analysis of data on 3D meshes and 3D point clouds using graph signal processing.

**Olivier Lézoray** received the M.Sc. and Doctoral degrees in computer science from the Universiteé de Caen Basse-Normandie, France, in 1996 and 2000, respectively. From 1999 to 2000, he was an Assistant Professor with the Computer Science Department, Université de Caen Basse-Normandie. From 2000 to 2009, he was an Associate Professor Communications, Networks and Services Department, Cherbourg Institute of Technology. Since 2010, he has been a Full Professor at the Cherbourg Institute of Technology. His research concerns color image segmentation and filtering (graph-based variational and morphological methods) and machine learning techniques for image mining (neural networks and support vector machines).

## References

[1] P. Arias, G. Facciolo, V. Caselles, and G. Sapiro, "A variational framework for exemplar-based image inpainting," *Int. J. Comput. Vision*, vol. 93, no. 3, pp. 319–347, 2011.

[2] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005, vol. 2, pp. 60–65.

[3] J. Stam, "Flows on surfaces of arbitrary topology," *ACM T. Graphics*, vol. 22, no. 3, pp. 724–731, 2003.

[4] A. Spira and R. Kimmel, "Geometric curve flows on parametric manifolds," *J. Comput Phys.*, vol. 223, no. 1, pp. 235–249, 2007.

[5] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, vol. 153, Springer, 2004.

[6] M Bertalmio, L Cheng, S Osher, and G Sapiro, "Variational problems and partial differential equations on implicit surfaces," *J. Comput Phys.*, vol. 174, no. 2, pp. 759–780, 2001.

[7] Steven J Ruuth and Barry Merriman, "A simple embedding method for solving partial differential equations on surfaces," *J. Comput Phys.*, vol. 227, no. 3, pp. 1943–1961, 2008.

[8] R. Lai and T. F. Chan, "A framework for intrinsic image processing on surfaces," *Comput. Vis. Image Und.*, vol. 115, no. 12, pp. 1647–1661, 2011.

[9] Jian Liang, Rongjie Lai, Tsz Wai Wong, and Hongkai

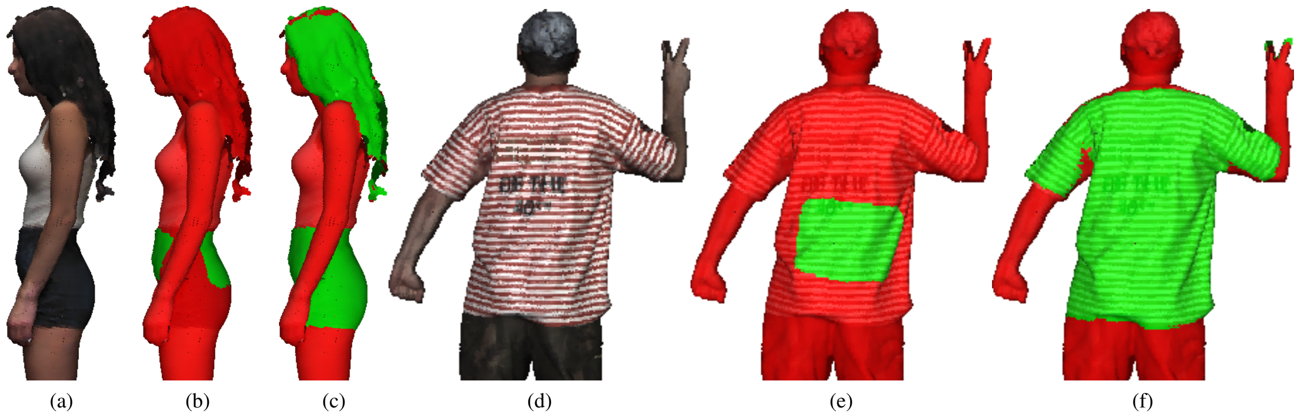


Figure 10: Segmentation of a real 3D scan of a woman and a man. From left to right respectively: colored point cloud to analyzed (a) and (d), initial labels (b) and (e), final segmentation (c) and (f).

Zhao, “Geometric understanding of point clouds using laplace-beltrami operator,” in *CVPR*, June 2012, pp. 214–221.

[10] R. Lai, J. Liang, and H.K. Zhao, “A local mesh method for solving pdes on point clouds.,” *Inverse Probl. Imag.*, vol. 7, no. 3, pp. 737–755, 2013.

[11] A. Elmoataz, O. Lezoray, and S. Bougleux, “Nonlocal discrete regularization on weighted graphs: a framework for image and manifold process ing,” *IEEE T. Image Process.*, vol. 17, no. 7, pp. 1047–1060, 2008.

[12] V.-T. Ta, A. Elmoataz, and O. Lézoray, “Nonlocal pdes-based morphology on weighted graphs for image and data processing,” *IEEE T. Image Process.*, vol. 26, no. 2, pp. 1504–1516, june 2011.

[13] O. Lezoray, V.-T. Ta, and A. Elmoataz, “Partial differences as tools for filtering data on graphs,” *Pattern Recogn. Lett.*, vol. 31, no. 14, pp. 2201–2213, 2010.

[14] O. Lézoray, A. Elmoataz, and V.-T. Ta, “Nonlocal pdes on graphs for active contours models with applications to image segmentation and data clustering,” in *ICASSP*, 2012, pp. 873–876.

[15] A. Chambolle and T. Pock, “A first-order primal-dual algorithm for convex problems with applications to imaging,” *J. Math. Imaging. Vis.*, vol. 40, no. 1, pp. 120–145, May 2011.

[16] A. Elmoataz, X. Desquesnes, and O. Lezoray, “Non-local morphological pdes and p -laplacian equation on graphs with applications in image processing and machine learning,” *IEEE J. Sel. Top. Signa.*, vol. 6, no. 7, pp. 764 –779, 2012.

[17] A. N. Tikhonov, “Solution of incorrectly formulated problems and the regularization method,” *Soviet Math. Dokl*, 1963.

[18] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Phys. D*, vol. 60, no. 1-4, pp. 259–268, Nov. 1992.

[19] D. Dobson and O. Scherzer, “Analysis of regularized total variation penalty methods for denoising,” *Inverse Probl.*, vol. 12, no. 5, pp. 601, 1996.

[20] P. Perona and J. Malik, “Scale-space and edge detection using anisotropic diffusion,” *IEEE T. Pattern Anal.*, vol. 12, pp. 629–639, 1990.

[21] F. R. Chung, “Spectral graph theory,” *CBMS Regional Conf. Ser. in Math.*, vol. 92, pp. 1–212, 1997.

[22] D. Luo, H. Huang, C. H. Q. Ding, and F. Nie, “On the eigenvectors of $p$-laplacian,” *Mach. Learn.*, vol. 81, no. 1, pp. 37–51, 2010.

[23] G. Gilboa and S. Osher, “Nonlocal linear image regularization and supervised segmentation,” *Multiscale Model. Sim.*, pp. 595–630, 2007.

[24] J. Weickert, *Anisotropic Diffusion In Image Processing*, 1996.

[25] M. Desbrun, M. Meyer, P. Schröder, and Alan H. Barr, “Implicit fairing of irregular meshes using diffusion and curvature flow,” in *SIGGRAPH*, 1999, pp. 317–324.

[26] S. Morigi, M. Rucci, and F. Sgallari, “Nonlocal surface fairing,” in *SSVM*, vol. 6667 of *Lecture Notes in Computer Science*, pp. 38–49. Springer Berlin Heidelberg, 2012.

[27] S. Fleishman, I. Drori, and D. Cohen-Or, “Bilateral mesh denoising,” *ACM T. Graphics*, vol. 22, no. 3, pp. 950–953, July 2003.

[28] P. Choudhury and J. Tumblin, “The trilateral filter for high contrast images and meshes,” in *EGWR*, 2003, pp. 186–196.

[29] J.-E. Deschaud and F. Goulette, “Point cloud non local denoising using local surface descriptor similarity,” in *IAPRS*, 2010, vol. XXXVIII, pp. 109–114.

[30] J. Digne, “Similarity based filtering of point clouds,” in *CVPRW*, june 2012, pp. 73 –79.

[31] T. Guillemot, A. Almansa, and T. Boubekeur, “Non local point set surfaces,” in *3DIMPVT*, 2012.

[32] Y. Zhang and A. Ben Hamza, “Vertex-based diffusion for 3-d mesh denoising,” *IEEE T. Image Process.*, vol. 16, no. 4, pp. 1036–1045, 2007.

[33] R. W. Brockett and P. Maragos, “Evolution equations for continuous-scale morphological filtering,” *IEEE T. Signal Proces.*, vol. 42, no. 12, pp. 3377–3386, 1994.

[34] M. Ghoniem, Y. Chahir, and A. Elmoataz, “Geometric and texture inpainting based on discrete regularization on

graphs," in *ICIP*, 2009, pp. 1349–1352.

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

[36] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model," *J. Math. Imaging. Vis.*, vol. 28, no. 2, pp. 151–167, 2007.

[37] X. Bresson and T.F. Chan, "Non-local unsupervised variational image segmentation models," *UCLA CAM Report 08-67*, 2008.

[38] T. Chan, S. Esedoglu, and M. Nikolova, "Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models," *SIAM J. Appl. Math.*, vol. 66, no. 5, pp. 1632–1648, 2006.

[39] M. Hidane, O. Lézoray, and A. Elmoataz, "Nonlinear Multilayered Representation of Graph-Signals," *J. Math. Imaging. Vis.*, vol. 45, no. 2, pp. 114–137, 2013.

[40] H.H. Bauschke and P.L. Combettes, *Convex analysis and monotone operator theory in Hilbert spaces*, Springer, 2011.

[41] 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*, vol. 45, no. 6, pp. 891–923, Nov. 1998.

[42] A.Buades, B.Coll, and J.-M. Morel, "Image denoising methods. a new nonlocal principle," *SIAM Rev.*, vol. 52, no. 1, pp. 113–147, 2010.

[43] F. Lozes, A. Elmoataz, and O. Lézoray, "Nonlocal processing of 3d colored point clouds," in *ICPR*, 2012, pp. 1968–1971.