Fast 3D surface reconstruction from point clouds using graph-based fronts propagation

Abdallah El Chakik, Xavier Desquesnes, Abderrahim Elmoataz

UCBN, GREYC - UMR CNRS 6972, 6.Bvd Marechal Juin, 14050 Caen, FRANCE

Abstract. This paper proposes a surface reconstruction approach that is based on fronts propagation over weighted graphs of arbitrary structure. The problem of surface reconstruction from a set of points has been extensively studied in the literature so far. The novelty of this approach resides in the use of the eikonal equation using Partial difference Equation on weighted graph. It produces a fast algorithm, which is the main contribution of this study. It also presents several examples that illustrate this approach.

1 Introduction

The main goal of this work is to propose a fast surface reconstruction method from point clouds, using a graph-based representation. This reconstruction is performed using a Partial difference Equation (PdEs) fronts propagation algorithm based on weighted graph.

Brief Literature Overview. Surface reconstruction from point clouds is an important problem in geometric modeling. Given a set of points $X = \{x_1, x_2, .., x_n\} \subset \mathbf{R}^n$ sampled from some unknown surface S, the surface reconstruction problem is to construct a surface $\hat{S}$ from the observed data X such as $\hat{S}$ approximates S.

Most surface representation techniques for point clouds reconstruction methods are classified into two categories, namely explicit and implicit methods. Explicit surface representations prescribe the surface location and geometry in an explicit manner and are mainly based on Delaunay triangulations or dual Voronoï diagrams. A popular technique is to construct a polyhedral surface from the input set of points using the Voronoï diagram [8]. Implicit surface representations embed surfaces as a co-dimension one level set of a scalar-valued function. In [5], a variational level set method was proposed, it introduced a distance-based energy functional, solved by level set method. Recently, this work was extended in [6, 7]. In this paper, we focus on the level set transcription on weighted graph.

Level set method. The level set formulation to describe a curve evolution has been introduced by Osher-Sethian [1], and is used in many works for 3D surface reconstruction based on front propagation. In [4], Claisse and Frey solved

the surface reconstruction problem using the following equation :

$$\frac{\partial_\phi}{\partial_t}(t,x) = |\nabla_\phi(t,x)|(\beta k(\phi)(t,x) + (\alpha d)(x)), \tag{1}$$

where $k(\phi)(t,x)=\nabla.n(\phi)(t,x)=\Big(\nabla.(|\frac{\nabla_u}{|\nabla_u|})t(tx)\Big)$ is the local mean curvature of the surface considered, α and $\beta \in \mathbf{R}$ and $d(x)$ is here the distance between x and the initial point set in $\mathbf{R}^3$. An important drawback of the level set approach stems from the expanse by embedding the front in $\mathbf{R}^d$ as the level set of $d+1$ dimensional function. Considerable computational labor is required per time step.

Contribution. We propose in this work a different and efficient approach that reduces the computational time without loss of precision. Our contribution is the transcription of the problem from $\mathbf{R}^3$ to a weighted graph. Indeed, any set of discrete data can be modeled as a weighted graph $G = (V, E, w)$ where V is the set of vertices that represents the data, E is the set of weighted edges and w is a weight function that represents the interactions between the data (see Section 2 for more details). We demonstrate that the graph representation allows to significantly reduce the amount of space points to be treated by the different surface reconstruction algorithms, thus increasing their performance. Then we extend the previously introduced PdE based fronts propagation method on weighted graphs in [3] to the 3D surface reconstruction problem. This method is based on the resolution of the following equation : $\mathcal{F}(u)\|(\nabla_w^- T)(u)\|_p = 1$, where ∇_w^- is an upwind discrete weighted gradient on a graph, T is the arrival time function of the fronts and $\mathcal{F}$ is the propagation speed function.

Paper organization. The rest of this paper is organized as follows. Section 2 presents a general definition of Partial difference Equations on weighted graph. It also describes our fronts propagation method on weighted graphs. Section 3 presents our graph-based surface reconstruction method. Section 4 presents some experiments. Finally, Section 5 concludes this paper.

2 Partial Difference Equations on Weighted Graphs

We begin briefly by reviewing some basic definitions and operators on weighted graphs.

Notions and Definitions. We assume that any discrete domain can be modeled by a weighted graph. Let $G = (V, E, w)$ be a weighted graph composed of two finite sets : $V = \{u_1, ..., u_n\}$ of n vertices and $E \subset V \times V$ a set of weighted edges. An edge $(u, v) \in$ E connects two adjacent vertices u and v. The weight w_{uv} of an edge (u, v) can be defined by a function $w : V \times V \to \mathbf{R}^+$ if $(u, v) \in E$, and $w_{uv} = 0$ otherwise. We denote by $N(u)$ the neighbor of a vertex u, i.e. the subset of vertices that share an edge with u.

Let $f : V \to \mathbf{R}$ be a discrete real-valued function that assigns a real value $f(u)$ to each vertex $\mathbf{u} \in \mathbf{V}$. We denote by $\mathscr{H}(\mathbf{V})$ the Hilbert space of such functions.

Operators on Weighted Graphs. For a better comprehension of the next Section, we now quickly recall some operators on weighted graphs as they are defined in [3, 14]. Considering a weighted graph $G = (V, E, w)$ and a function $f \in \mathscr{H}(\mathbf{V})$, the *weighted discrete partial derivative operator* of $\mathbf{f}$ is :

$$(\partial_v f)(u) = \sqrt{w_{uv}}(f(v) - f(u)) \tag{2}$$

Based on this definition, two weighted directional difference operators are defined. The weighted directional external and internal difference operators are respectively :

$$\begin{aligned} (\partial_v^+ f)(u) &= \sqrt{w_{uv}}(f(v) - f(u))^+ and \\ (\partial_v^- f)(u) &= \sqrt{w_{uv}}(f(v) - f(u))^- \end{aligned} \tag{3}$$

with $(x)^+ = max(0, x)$ and $(x)^- = -min(0, x)$.
The weighted gradient of a function $f \in \mathscr{H}(\mathbf{V})$ at vertex u is the vector of all edge directional derivatives :

$$(\nabla_w f)(u) = (\partial_v f(u))_{v \in V}^T \tag{4}$$

And the weighted morphological external and internal gradient $(\nabla_w^+ f)(u)$ and $(\nabla_w^- f)(u)$ are :

$$(\nabla_w^\pm f)(u) = \left(\left(\partial_v^\pm f\right)(u)\right)_{v \in V}^T. \tag{5}$$

2.1 Front Propagation on weighted graphs

In this section we will present the fronts propagation approach on weighted graphs.

Let $G = (V, E, w)$ be a weighted graph. A front evolving on G is defined at initial time as a subset $\Omega_0 \subset V$, and is implicitly represented by a level set function ϕ_0 such that ϕ_0 equals 1 in Ω_0 and -1 on its complementary.
Then, the front propagation is described by the following equation

$$\begin{cases} \frac{\partial \phi}{\partial t}(u) &= (\mathcal{F}(u) \| (\nabla_w \phi)(u) \| \\ \phi_0(u) &= \phi_0 \end{cases} \tag{6}$$

with $\mathcal{F} \in \mathscr{H}(\mathbf{V})$, and $w : V \times V \to \mathbf{R}^+$ is the weighted function. Only considering the case $\mathcal{F} \geq 0$, and with $\phi(u, t) = t - T(u)$, The authors have shown in [3] that previous equation can be rewritten as

$$\begin{aligned} \frac{\partial_\phi(u, t)}{\partial t} &= \mathcal{F}(u) \| (\nabla_w^+ (t - T))(u) \|_p \\ &= \mathcal{F}(u) \| (\nabla_w^- T)(u) \|_p = 1, \\ &\| (\nabla_w^- T)(u) \|_p = P(u) \end{aligned} \tag{7}$$

where $P(u) = 1/\mathcal{F}(u)$.

This equation is the stationary version of the level set equation (6) that corresponds to the well-known eikonal equation and $T : V \to \mathbf{R}$ is the arrival-time function that associates the arrival time of Γ to each vertices of V. This function can also be considered as a distance function that provides the distance between each vertex u and Ω_0.

In [3], the authors have proposed several numerical schemes to solve such equations for different $\mathcal{L}_p$ norms. They also presented an efficient algorithm inspired from Fast Marching that allows to compute the propagation and the arrival time of many fronts evolving on a graph. In this case, equation (7) is associated with a label function that mark each vertex u with the label of the first front that reach u. The label function $L : V \to [0, K]$ is initialized as

$$L(u) = \begin{cases} i \in [1, K] & \text{if u belongs to front } i \text{ at initial time} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

where K is the number of fronts. Interested readers should refers to [2, 3] for more details. Such algorithm has been successfully used for geodesic distance computation on weighted graphs, image segmentation and data clustering. In the next section, we will propose a new surface reconstruction method based on this algorithm.

3 Method

Our reconstruction method consists of completing the initial point clouds with a very dense set of points that will be part of the resulting reconstructed surface. The completion is performed by selecting new points from a very dense set of candidate points, generated in the neighborhood of initial points, and including initial points. Our approach considers the set of candidate points as a weighted graph (constructed from the set) and the surface to be reconstructed as a subset of the vertices of this graph. Working on this graph, and by analogy with the level set method, the surface to be reconstructed is considered as the interface between two inner and outer fronts evolving on the whole graph. These two fronts are driven according to a potential field that controls their propagation speed, and defined on the graph vertices such that the two fronts collapse on the object boundary.

Graph construction. The first step extends the initial point clouds in order to generate candidate points and constructs the associated weighted graph. In order to precisely fill in the holes of the object, we need a very dense additional point clouds in the neighborhood of the initial points. But high density point clouds penalizes the computational efficiency due to the high number of candidates to be treated. For this reason, we propose to use an adaptive approach

that allows to add a very dense additional points only where it is necessary (near the initial points), and very sparse additional points elsewhere.

We consider the initial point clouds to be represented by a set of points $X = \{x_1, x_2, ..., x_n\} \subset \mathbf{R}^n$. This initial point clouds X is extended with new points as follows : Let C be the candidate points that are regularly added to the neighborhood of the initial points, such that the density of the new points is very high near the initial points and decreases as we move away. This is performed by the adaptive triangulation method proposed by [4] that produces a triangulated adapted mesh which vertices include initial and candidates points. Let X' be the joint set of initial and candidate points ($X' = X \cup C$).

We denote $G(V, E, w)$ the weighted graph constructed from the previously obtained mesh. The set of vertices V represent the points of X', such that the vertex $v_i \in V$ represents the point $x'_i \in X'$. Let V_0 be the set of vertices that corresponds to the initial points (X). The set of edges E is given by the triangulated mesh edges. The weight function w defines a similarity between the two vertices of each edge. This similarity is based on the position in space of the associated points, we have $w(v_i, v_j) = -exp(d(x'_i, x'_j)^2/\sigma^2) \, \forall (v_i, v_j) \in E$. With this definition, the weight function holds the spatial relations of the extended point clouds X'.

We will now present the fronts initialization and the potential field definition, both based on the same distance map $\mathcal{D}$ computed from initial point clouds. This distance is computed as follows.

Distance Map. The distance map is a function $\mathcal{D} : V \to \mathbf{R}^+$ that associates each vertex u of G with the distance between the set V_0 and u.

The distance map is computed using the Fast Marching algorithm on graphs (7)(see Sec.2), for a single front initialized on initial points (i.e., $\Omega_0 = V_0$), and with constant potential function ($P = 1$).

We recall that this algorithm produces both a label function (L) for fronts propagation and the associated arrival-time function T that can be considered as a distance map between Ω_0 and each vertex $u \in V$. In this case, function T provides distance from the single set V_0 and we have $\mathcal{D} = T$.

Inner and outer fronts initialization. Inner and outer fronts are initialized equidistantly from the initial point clouds, using the previous distance function $\mathcal{D}$. The inner front Γ_i is initialized by the subset Ω_0^i of vertices that lies inside the object and at a distance k from the initial points. We have $\Omega_0^i = \{u \mid \mathcal{D} = k \pm \varepsilon \text{ and } u \text{ inside}\}$. Similarly, the outer front Γ_o is initialized by the subset Ω_0^o of vertices that lies outside the object and at a distance k from the initial points. We have $\Omega_0^o = \{u \mid \mathcal{D} = k \pm \varepsilon \text{ and } u \text{ outside}\}$.

In the case where the fronts are not equidistant, the nearest front from the surface to be reconstructed will always be favored and fronts may collapse far from the object surface.

Potential field. In the case where the object has thin parts, the inner front will be favored and the fronts will collapse outside the object. To prevent this problem, we introduce a potential field that controls the fronts propagation such that the fronts moves very slowly near the initial points (near the object boundary) and move faster elsewhere. The better potential function is given by the distance map function $\mathcal{D}$. Indeed the distance map is almost null near initial points which guarantees that the fronts will be slowly propagated or stopped near the surface to be reconstructed. Then the potential function is defined as $P = \mathcal{D}$.

Surface reconstruction. Once the inner and outer fronts are initialized, we set a label to each graph vertices as follows : the vertices belonging to Ω_0^i are labeled by 1 ($L = 1$) and the vertices belonging to Ω_0^o are labeled by 2 ($L = 2$) and the rest of vertices are labeled by 0 ($L = 0$).

Then we propagate those labels on the graph using the Fast Marching algorithm on graphs (7) presented in Sec.2. The algorithm is initialized as $\Omega_0 = \Omega_0^i \cup \Omega_0^o$, and the potential function is given by $P = \mathcal{D}$.

Due to the potential field that slows down the fronts on the neighborhood of the object boundary, both fronts collapse on this boundary. Then, the vertices that lies inside the object are labeled by 1 and the vertices that lies outside the object are labeled by 2 and the resulting reconstructed surface is the vertices belonging to the inner labels and have at least one neighborhood belonging to the outer labels.

Remark. We can apply a spacial regularization on the selected vertices to adjust the vertices positions to top-up the resulting surface smoothness.

Complexity. If the graph is totally connected, the complexity of the proposed model is $O(N^3)$, where N is the number of nodes in the graph.

4 Experiments

This section demonstrate the speed and the robustness of our surface reconstruction approach by applying it on different examples. First, we explain our approach on a 2D points set for better comprehension, we show our approach results on 3D points set and we compare our approach results with some other approaches.

Figure 1 shows the initial 2D points and the adapted mesh created. The initial points number is 790 points, the adapted mesh produce 15451 points and 30868 triangles. The adapted mesh will be transformed into a weighted graph. We define a single label on the initial vertices (vertices to be reconstructed) that will propagate on the whole graph and compute the distance map of each point to the nearest initial points using the equation (7), figure 2 shows the distance map computed.

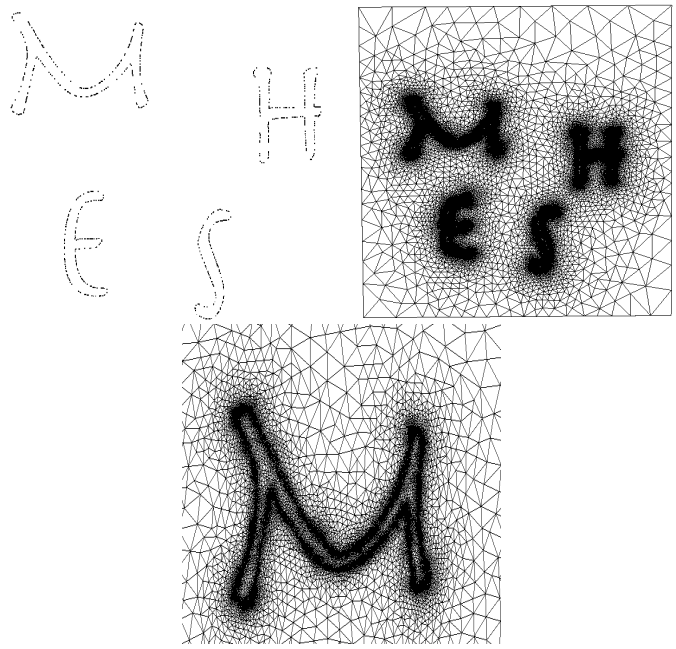

Fig. 1. The adapted mesh. Left : initial points; Right : the adapted mesh, this mesh contain 15451 points and 30868 triangles.

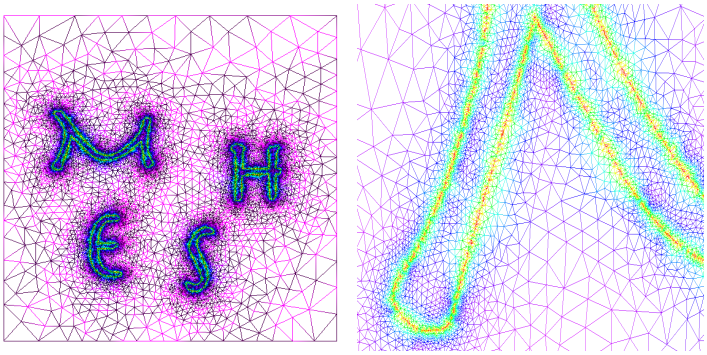

Fig. 2. Distance map computed by propagating a single front from the initial points on the whole graph. The colors represent each point distance value from the intial points. The initial points distance value is red.

Once our distance map is computed, we define inner and outer labels using this distance map. Figure 3 shows the labels and the object reconstructed.

Figure 4 shows a cut of the face adapted mesh. One can see the high points density near the initial points and the low points density moving away from the initial points. The initial points number is 67206, the adaptive triangulation pro-

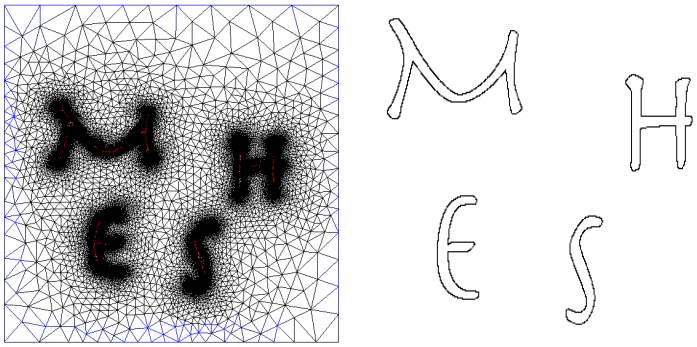

Fig. 3. Left : the labeled mesh. We assign to each label a color, the red color represent the vertices belonging to inner label ($L = 1$), The vertices colored by blue belonging to the outer label ($L = 2$), The rest of vertices colored by black are labeled by 0 ($L = 0$) and represent the propagation space of the inner and outer labels; Right : our approach surface reconstruction result that produce 6421 connected vertices.

duces 952869 points. Figure 5 shows the results of our reconstruction approach,

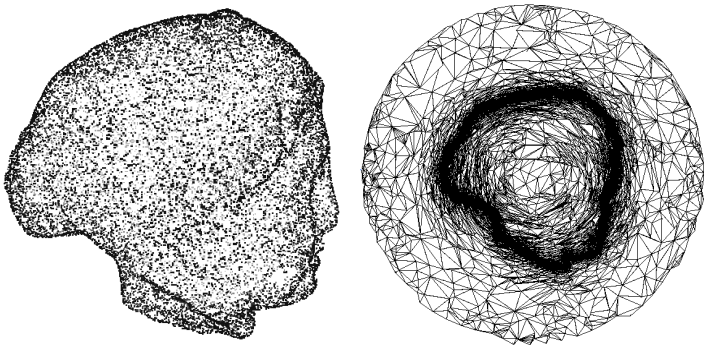

Fig. 4. Left : initial face point clouds (venus) to be reconstructed that contain 67026 points; Right : a cut of the face adapted mesh, the full face adapted mesh contain 952869 points and 5147018 triangles.

one can see the reconstructed objects smoothness. The resulting reconstructed object : face (points : 579992, triangles : 1063855), dragon (points : 451275, triangles :703488), duck (points : 498779 , triangles : 836483).

We tested our approach on a voxels image example to prove that our approach deals with multiple data types.We show our approach result on the Stanford dragon point clouds [15]. We transformed our point clouds to a voxels 3D image and constructed our weighted graph $G(V, E, w)$ such that a single

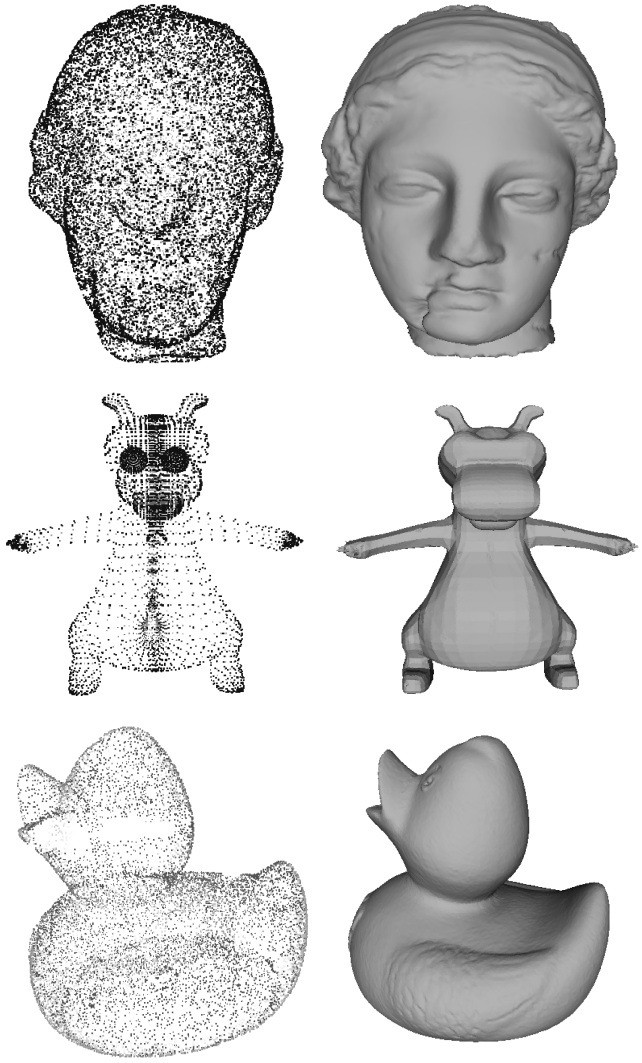

Fig. 5. Surface reconstruction result using our approach. First column : initial point clouds; Second column : our approach surface reconstruction results. One can see that the holes in the surface (for the duck example) are reconstructed.

vertex is associated with each point in $\mathbf{R}^3$. The weight function is constant ($w(u, v) = 1$, $\forall (u, v) \in E$).

Figure 6 illustrate the result of our surface reconstruction approach on the 3D voxels image (grid size : 240×169×107). The surface reconstruction algorithms computational time is 65 seconds for this example. One can see the time computational difference between this examples and the adapted meshes examples,

Point Cloud	Adapted points number	Reconstruction time
cart	15451	**1.50 seconds**
duck	943137	**38.38 seconds**
Face	952869	**38.98 seconds**
Dragon	991583	**39.8 seconds**

Table 1. Our method time computational for different point clouds examples.

thanks to the adaptive triangulation thats minimize the weighted graph vertices number.

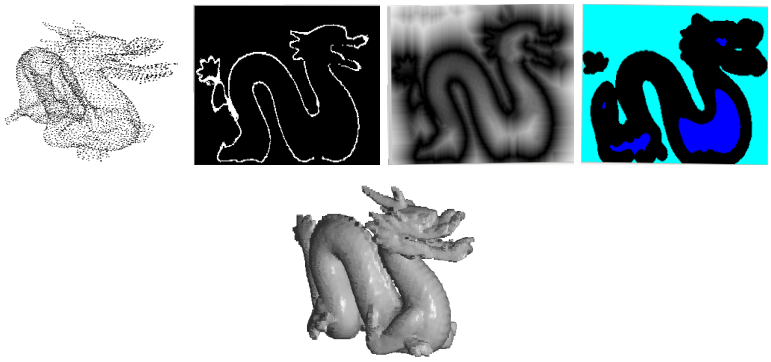

Fig. 6. Surface reconstruction example on 3D voxels image ; Left to right : dragon initial point clouds, cut of dragon voxels 3D image, distance map computed by propagation a single front starting from the initial points and evolving on the whole graph, labels definition (the blue color represent the inner labels and the green color represent the outer labels and the the black color represent the space label propagation), the surface reconstruction result of the dragon.

Comparison with some other approaches. Following the quantitative information given in [9], we compare the performances of our reconstruction approach on the Stanford bunny point clouds [15] with different methods [10, 11, 12, 13], figure 7 shows our surface reconstruction result on this example. Table 2 presents the difference between our approach and the other approaches. In terms of computation time, our method is comparable to that of the MPU method, which is one of the fastest geometrically-adaptive reconstruction methods according to [12]. The Power Crust method is about 10 times slower, and the Poisson method is about 4 times slower than our approach. The FFT oppe et al. and methods are about 2 time slower but the first suffers from large memory and the second produce some holes in the final reconstructed object. In term

of reconstruction quality, The method by Hoppe et al. and the Power Crust method generate a smooth surface with some holes that are still visible, The MPU method provide a smooth surface without holes, but with some artefacts. The FFT, Poisson and our method accurately reconstruct the surface of the bunny. In terms of peak memory usage, our approach have a reasonable memory usage which corresponds to the amounts of the memory to store the dense graph constructed from the adapted mesh.

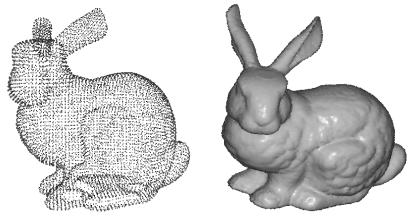

Fig. 7. Our approach result on Stanford bunny example. Left : the bunny initial point clouds; Light : our approach reconstruction result.

Method	**Time**	**Peak memory**	**Triangles**
Power Crust	504	2601	1,610,433
Poisson method	188	283	783,127
FFT method	93	1700	1,458,356
Hoppe et al.	82	230	630,345
MPU	78	421	2,121,041
Our method	45	980	2,759,146

Table 2. Different methods Computational time for the Stanford bunny. Computational time (seconds), peak-memory usage (mega-bytes) and number of triangles of the reconstructed surfaces of three range data sets.

Advantages. This method offers several advantages. First of all, for adapted meshes, the graph representation allows to have very dense additional points only where it is necessary (near initial points), and very sparse additional points elsewhere. This significantly reduces the number of points to be treaded by the algorithm, and thus the computational time. On the contrary, traditional methods using three dimensional regular grids of voxels have to choose between precision (with a very dense grid) and computational efficiency. Second, all processes are performed using a single general algorithm which allows to deal with many fronts on weighted graphs of arbitrary topology, and can be used for many types of data (not only 3D point clouds). Finally, no spatial discretization is needed, thanks to the equations being directly expressed in a discrete form.

5 Conclusion

In this paper, we proposed a point clouds fast surface reconstruction algorithm based on fronts propagation over weighted graphs. We show that our approach deals with multiple types of data and produces robust results. In addition, this method is fast and does not need large memory requirements.

References

1. J. A. Sethian : *Level Set Methods and Fast Marching Methods. Some Fine Journal*, Cambridge University Press, 1999.
2. X. Desquesnes, A. Elmoataz, O. L'ezoray : *PDEs Level Sets on Weighted Graphs.* International Conference on Image Processing (IEEE), 4(2):125, 2011.
3. X. Desquesnes, A. Elmoataz, O. Lzoray, V-T. Ta : *Efficient Algorithms for Image and high Dimensional Data Processing Using Eikonal Equation on Graphs. Some Fine Journal*, International Symposium on Visual Computing, Vol. LNCS 6454, pp. 647-658, 2010.
4. A. Claisse, P. Frey : *A nonlinear PDE model for reconstructing a regular surface from sampled data using a level set formulation on triangular meshes q.* J. Comp. Phys., 2011.
5. H.K. Zhao, S. Osher, B. Merriman, and M. Kang : *Implicit and nonparametric shape reconstruction from unorganized points using variational level set method.* Computer Vision and Image Understanding, 80(3):295319, 2000.
6. J. Ye, I. Yanovsky, B. Dong, R. Gandlin, A. Brandt, and S. Osher : *Multigrid Narrow Band Surface Reconstruction via Level Set Functions.* Submitted for publication, 2008.
7. T. Goldstein, X. Bresson, and S. Osher : *Geometric Applications of the Split Bregman Method: Segmentation and Surface Reconstruction.* UCLA CAM Report, pages 0906, 2009.
8. B. Mederos, N. Amenta, L. Velho, and L.H. de Figueiredo : *Surface reconstruction from noisy point clouds.* In Proceedings of the third Eurographics symposium on Geometry processing, page 53. Eurographics Association, 2005.
9. Jalba, A.C., Roerdink, J.B.T.M : *Efficient surface reconstruction using generalized coulomb potentials.* IEEE Transactions on Visualization and Computer Graphics 13, 15121519 (November 2007.
10. Amenta, N., Choi, S., Kolluri, R.K : *The power crust.* Proceedings of the sixth ACM symposium on Solid modeling and applications. pp. 249266. SMA 01, ACM, New York, NY, USA (2001).
11. Braude, I., Marker, J., Museth, K., Nissanov, J., Breen, D : *Contour-based surface reconstruction using mpu implicit models.* GRAPHICAL MODELS 69, 2007.
12. Kazhdan, M., Bolitho, M., Hoppe, H : *Poisson surface reconstruction.* Proceedings of the fourth Eurographics symposium on Geometry processing. pp. 6170. SGP 06, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland (2006).
13. Kazhdan, M.M. : *Reconstruction of solid models from oriented point sets.* Symposium on Geometry Processing. pp. 7382 (2005).
14. A. Elmoataz , O.Lezoray, S. Bougleux : *Weighted Nonlocal Discrete Regularization on Graphs: a framework for Image and Manifold.* Processing IEEE Transactions On Image Processing, 17 (7), pp: 1047-1060, 2008.
15. Stanford 3D scanning repository, http://graphics.stanford.edu/data/3Dscanrep/.