Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems dealing with Visibility

Amael Delaunoy, Emmanuel Prados

▶ **To cite this version:**

Amael Delaunoy, Emmanuel Prados. Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems dealing with Visibility. International Journal of Computer Vision, 2011, 95 (2), pp.100-123. 10.1007/s11263-010-0408-9 . inria-00546105

HAL Id: inria-00546105
https://inria.hal.science/inria-00546105

Submitted on 24 May 2011

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

International Journal of Computer Vision manuscript No.
(will be inserted by the editor)

Gradient Flows for Optimizing Triangular Mesh-based Surfaces: Applications to 3D Reconstruction Problems dealing with Visibility

Amaël Delaunoy · Emmanuel Prados

Received: date / Accepted: date

Abstract This article tackles the problem of using variational methods for evolving 3D deformable surfaces. We give an overview of gradient descent flows when the shape is represented by a triangular mesh-based surface, and we detail the gradients of two generic energy functionals which embody a number of energies used in mesh processing and computer vision. In particular, we show how to rigorously account for visibility in the surface optimization process. We present different applications including 3D reconstruction from multiple views for which the visibility is fundamental. The gradient correctly takes into account the visibility changes that occur when a surface moves; this forces the contours generated by the reconstructed surface to match with the apparent contours in the input images.

Keywords Triangle Mesh-based Surface · Gradient Descent Flow · Surface Evolution · Variational Methods · Shape Gradient · Visibility · 3D Reconstruction · Multi-view Stereovision

1 Introduction

Variational methods are commonly used in computer vision and computer graphics to compute, improve and process surface interfaces. Such approaches consist of defining an energy whose minimum is reached by the surface of the object of interest. In particular, this framework has been widely used in 3D reconstruction problems, see for example [5,13,15,22,23,24,26,29,35,40,41,46,47,49]. In mesh processing, geometric flows have been extensively used in different applications such as texture synthesis [2], mesh denoising [9,30] and shape matching [12].

This article focuses on the optimization of *2D surfaces of* $\mathbb{R}^3$ represented by *triangle meshes*, via gradient descent methods. We rigorously establish and detail the *gradient flows* of some *generic* energies which encompass a large number of energies used in computer vision and for which the normal to the surface and the *visibility* appear in their formulation. We demonstrate the interest of this contribution by illustrating it via several applications, in particular applications in 3D reconstruction from multiple calibrated cameras.

1.1 Considered Energies: from Weighted Area Functionals to Functionals that account for Visibility

In this paper, we first consider the following generic energy:

$$E(\mathcal{S}) = \int_{\mathcal{S}} g(\mathbf{x}, \mathbf{n}(\mathbf{x}))\mathrm{d}s \,, \tag{1}$$

where $\mathcal{S}$ is a $2D$ surface embedded into $\mathbb{R}^3$. Here $g : \mathbb{R}^3 \times \mathbb{S}^2 \to \mathbb{R}$ is a scalar function defined on the surface that eventually depends on the normal $\mathbf{n}$ to the surface $\mathcal{S}$; $\mathbb{S}$ being the unit sphere of $\mathbb{R}^3$. $\mathrm{d}s$ is the element of area of the surface. This generic energy is very classical and is called a weighted area functional. It has already been studied in the literature in the continuous framework, see in particular [16,17,42]. A number of energies proposed in the computer vision, image processing and mesh processing literature are particular cases of this energy.

Research was supported by the Agence Nationale pour la Recherche within the Flamenco project (Grant ANR-06-MDCA-007).

Amaël Delaunoy
INRIA Rhône-Alpes, Perception team, LJK – Grenoble, France
Tel.: +33 4 76 61 54 47
Fax.: +33 4 76 61 54 54
E-mail: Amael.Delaunoy@inrialpes.fr

Emmanuel Prados
INRIA Rhône-Alpes, Perception team, LJK – Grenoble, France
Tel.: +33 4 76 61 54 47
Fax.: +33 4 76 61 54 54
E-mail: Emmanuel.Prados@inrialpes.fr

In this paper we are also considering more complex families of energies. Generally in inverse problems, all rests on a priori knowledge (models, regularizations etc) and the data fidelity. A common solution to inverse problems is provided by minimizing some criteria which compares the real input data to some synthetic data generated by the models. Also, to be complete, the comparisons must be done on the data space. In particular, as explained in [15,39], in 3D reconstruction problems from image data, a natural solution would be a surface such that the images generated from the model are more similar to the observed images (i.e. the data). This naturally leads to formulate the problem as the minimization of an error measure between the observed and predicted values of pixels, carried out over all pixels in all input images. This thus brings us to minimize some energy of the form [15,39]:

$$E(\mathcal{S}) = \int_{\mathcal{I}} g\big(\, \pi_{\mathcal{S}}^{-1}(\mathbf{p}),\, \mathbf{n}(\pi_{\mathcal{S}}^{-1}(\mathbf{p}))\,\big)\, \mathrm{d}\mathbf{p}\,, \tag{2}$$

where $\mathcal{I}$ is the set on which data is defined (the set of all pixels in the image, in the case of the 3D reconstruction problems from images), and $\pi_{\mathcal{S}}^{-1}(\mathbf{p})$ is the point of the surface corresponding to $\mathbf{p}$ (the surface point which is viewed in pixel $\mathbf{p}$, in the 3D reconstruction problems from images). In the case where $\pi_{\mathcal{S}}^{-1}(\mathbf{p})$ does not reproject onto the surface $\mathcal{S}$, $\pi_{\mathcal{S}}^{-1}(\mathbf{p})$ is a point on the background B. $\mathrm{d}\mathbf{p}$ is the area measure on the sensor. g gives the error measure for the data point $\mathbf{p}$. Such functionals are generally called reprojection errors. One of the major properties in image formation, also one of the major problems in computer vision, is that only visible (i.e. unoccluded) elements are present in the image. Functionals (2) can then be rewritten as an integral over the surface (instead of the image) by counting only the visible points [34,39,47]. This gives, by a simple change of variables:

$$E(\mathcal{S}) = \int_{\mathcal{S}\cup B} \overline{\mathbf{g}}(\mathbf{x}, \mathbf{n}(\mathbf{x})) \cdot \mathbf{n}(\mathbf{x})\; \nu_{\mathcal{S}}(\mathbf{x})\, \mathrm{d}s\,. \tag{3}$$

This involves adapting the measure on the surface [39] and the expression of $\overline{\mathbf{g}}$ which directly depends on the projection model being used. In most cases, e.g. for orthographic, perspective or linear pushbroom cameras, we have $\overline{\mathbf{g}}(\mathbf{x},\mathbf{n}) = -g(\mathbf{x},\mathbf{n})k(\mathbf{x})\mathbf{d}(\mathbf{x})$, where $k(\mathbf{x})$ is a specific scalar function of $\mathbf{x}$ and $\mathrm{d}(\mathbf{x})$ is the projection vector of $\mathbf{x}$ according to the camera. For example, for a perfect pinhole camera model, the adequate $\overline{\mathbf{g}}$ is given by $\overline{\mathbf{g}}(\mathbf{x},\mathbf{n}) = -g(\mathbf{x},\mathbf{n})\,\frac{1}{\mathbf{x}_z^3}\mathbf{x}$, see [34,39,47]. For a linear pushbroom camera, we have $\overline{\mathbf{g}}(\mathbf{x},\mathbf{n}) = -g(\mathbf{x},\mathbf{n})\,\frac{1}{v\cdot\mathbf{d}(\mathbf{x})\,\mathrm{d}_z(\mathbf{x})}\mathbf{d}(\mathbf{x})$, where $\mathbf{d}(\mathbf{x})$ is the vector joining $\mathbf{x}$ and the optical center of the sensor at corresponding time; v is a vector depending on the speed of the satellite, see [18]. In (3), B is the surface behind $\mathcal{S}$ that corresponds to the background (i.e. the points on the data set which do not correspond to any point of the surface model of the object of interest). $\nu_{\mathcal{S}}(\mathbf{x})$ is the visibility function $\nu_{\mathcal{S}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ such that:

$$\nu_{\mathcal{S}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \text{ is visible from the camera,} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

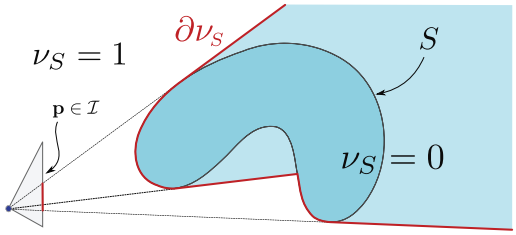

Fig. 1 *Banana Shape* seen from a vantage point (See [14]). The energy defined over the image explains the visibility interface $\partial\nu_{\mathcal{S}}$ (in red) of the surface $\mathcal{S}$.

Finally, by using the separation technique proposed by [47], we can rewrite energy (3) as an integral over only the visible surface:

$$E(\mathcal{S}) = \int_{\mathcal{S}} \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x})) \cdot \mathbf{n}(\mathbf{x})\; \nu_{\mathcal{S}}(\mathbf{x})\, \mathrm{d}s\,. \tag{5}$$

To obtain $\mathbf{g}$ from $\overline{\mathbf{g}}$, we refer the reader to Section 5.3.2 and especially Equation (49), as well as [47], which comprehensively details this step. Figure 1 illustrates the case of an energy defined over a visible volume.

Let us emphasize that handling properly the visibility term is a non-trivial undertaking. In particular, this is one of the major difficulties in the stereovision problem. Previous works cope with this difficulty more or less elegantly. Most often the authors approximate the visibility in some pre-processing steps which can be completely prior to the whole algorithm or else inside the iterations of the minimization process [3,13,20,22,34,39,50].

Only recently, some authors [15,47] manage to rigorously and fully account for visibility in the optimization process. In their recent work, Yezzi and Soatto [47] (for convex surfaces) and Gargallo et al. [15] (in the general case) provide the exact gradient of the reprojection error. This computation is done in the mathematical framework of continuous (smooth) surfaces as used in [42]. This work has shown that a proper handling of the visibility automatically forces the apparent contours generated by the reconstructed surface to perfectly match with the apparent contours in the observed data. This makes the use of additional energy terms like ballooning [44], visual hull [19,37] or contours unnecessary and significantly reduces the minimal surface bias present in many other problem formulations.

In this paper, we rigorously minimize energies (1) and (5). But contrary to the works [15,47] which deal with continuous surfaces, here we consider triangle mesh-based surfaces.

1.2 Gradient Descent Optimization

Energy (5) is rather difficult to optimize. In particular, the complexity is due to the dependency on the normal and the visibility term. There exist several tools to minimize energy functionals. Recent methods, such as graph-cuts for example, allow us to find global minimum. At the present time these global optimization techniques are limited to rather simple energies [28] and, based on the current state-of-the-art, it seems extremely difficult to apply them for minimizing energies such as (5). Recent advances also allow such minimization via total variation and convex relaxation methods [24,26], however it is difficult to apply those methods for functionals depending on the surface normal. The recent work of [27] allows to take the normal into account but is not directly applicable to general energy functionals like (1) and (5). Taking the normal into account in the minimization allows to produce high quality 3D models [31]. In this paper we compute the derivatives of the generic functionals (1) and (5) which allow us to minimize it via gradient descents [6,13,16,42], see Section 2.

1.3 Triangle Mesh-based Representation

Surface representation based on polyhedral and especially triangular meshes are the most commonly used in graphics. Moreover, the design of graphics hardware makes triangular meshes the most natural way to represent surfaces in a number of applications. In the variational framework, this type of discrete representations leads to the Lagrangian setting. Lagrangian methods are generally contrasted with Eulerian approaches which are mainly based on the level set representation [32]. During the last decades, Eulerian methods have become very popular mainly because they allow us to naturally deal with topological changes. In particular, this last setup has been extensively used for 3D reconstruction problems [5,13,15,22,23,25,29,35,40,41,46,49]. Nevertheless, recent advances in mesh processing allow Lagrangian methods to enjoy the same facilities [33,51]. In other respects, in the Eulerian methods, the gradient is computed in the continuous framework. Technically, computing the gradient in the continuous framework is more complicated than in a discrete framework, since the first one requires functional analysis when the second one only needs differential calculus. Furthermore, in practice in Eulerian approaches, one finally needs to discretize the continuous gradient flow since the level set function is also discretized on a grid. Also, this discretization (which is usually obtained using discrete differential operators [30] or finite element modeling [11]) is sometimes difficult to obtain, as we can see for example in Sections 3 and 4. By directly considering discrete surfaces, this last step is not necessary in our case.

Now, let us note that Langragian methods may be classified into two different approaches: The first strategy consists in 1) formulating the problem with continuous surfaces and computing the gradient in the continuous framework, then 2) discretizing the continuous gradient in order to apply it to the discrete surface. The second strategy consists in 1) formulating directly the problem with the discrete surface representation, then 2) computing the exact gradient of the formulated energy. Clearly, the second strategy is better than the first one: in fact in the first strategy we do not know if finally in practice the discrete representation minimizes something; in the best case, we do not know which exact energy the computed solution really minimizes since it does not necessarily corresponds to the representation. Also, surprisingly, a number of works follow the first strategy, see for example [3,10,51]. In particular, the gradient is computed using normal velocities whereas the second approach may have tangential components which leads to more coherent flow. In this paper, we follow the second strategy as other authors: for example, Slabaugh and Unal [38] who deal with surface segmentation, Eckstein et al. [12] who are interested in shape matching, or Vu et al. [45] who proposes a complete multi-view stereovision algorithm. We also use the similar strategy presented by Debreuve et al. [7] who deal with discrete parametric active contours for segmentation or Dziuk and Elliott [11] using finite element modeling on evolving surfaces. More exactly, we detail the exact gradient flow of energy (1) in which the surface $\mathcal{S}$ is explicitly a discrete surface based on triangular meshes, and also extend it to visibility-driven energies (5).

1.4 Contributions

In this article, we first give an overview of gradient descent flows with deforming surfaces, when represented by triangular surface meshes. Here, the gradient is the one of the energy defined with the discrete surface; we do not need to approximate and discretize it when we finally evolve the surface. Even though the visibility plays a key role in computer vision, until now it has been managed more or less elegantly. It is clearly a key difficulty in the field, which has been recently solved in the theory of continuous surfaces [14,15]. Here, we show how to rigorously deal with the visibility in the framework of discrete surface representations and we give the gradient flow of generic energies which encompasses a large class of energies used in computer vision (see Section 4).

We then illustrate the presented results by giving the gradient of commonly used functionals in computer vision and graphics, and we emphasize the 3D reconstruction applications for which the visibility is fundamental; we thus show in Section 5 how one can apply mesh evolution techniques to

3D reconstruction applications from multiple views. In particular in Section 5.3 we focus on the multi-view stereovision problem and we propose a successful algorithm which, since it fully accounts for visibility, automatically aligns contour generators with image contours.

This article generalizes our previous conference paper [8] in which the considered energy does not depend on the normal. This allows us to present here a larger spectrum of applications including multi-view shape from shading and multi-view photometric stereo (see Section 5.5).

2 Gradient Flows

In this section, we are interested in minimizing energy functionals defined on surfaces with respect to the surface representation. Whatever surface representation one chooses, the energy minimization should be consistent with it in order to be sure the energy minimized is the right one. It has been common to minimize such energy by performing gradient descent. Computing an adequate gradient corresponding to the representation of the surface is not trivial, and the following gives a way for computing generalized gradient flows for an energy $E(\mathcal{S})$, where $\mathcal{S}$ is a 2-dimensional surface in $\mathbb{R}^3$, g is a differentiable scalar function and $\mathbf{n}$ is the Gauss map for $\mathcal{S}$.

In the following, we explain how to obtain such gradients firstly in the theory of continuous smooth surfaces, and finally using triangular meshes.

2.1 Gradient Descent in the Continuous Case

Let M denote the set of all admissible 2D-manifolds embedded in $\mathbb{R}^3$, and $\mathcal{S} \in M$. Let $\mathbf{v}$ be a vector in the tangent space of M, denoted by $T_F M$, associated with an inner product $\langle \cdot, \cdot \rangle_F$. Let $E(\mathcal{S}) : M \to \mathbb{R}$ be a surface functional as defined previously (1) such that its Gâteaux Derivative in the direction $\mathbf{v}$ can be expressed as : $DE(\mathcal{S}, \mathbf{v}) \equiv \frac{\mathrm{d}}{\mathrm{d}\tau} E(S + \tau \mathbf{v})\Big|_{\tau=0}$. Then the gradient of E at $\mathcal{S}$ is the unique vector $\nabla_M E(\mathcal{S}) \in T_F M$ such that $DE(\mathcal{S}, \mathbf{v}) = \langle \nabla_M E(\mathcal{S}), \mathbf{v} \rangle_F$ for all $\mathbf{v} \in T_F M$. See for instance Solem and Overgaard [42] for a more detailed explanation.

Then, the gradient descent flow of an energy $E(\mathcal{S})$ as the form:

$$\begin{cases} \mathcal{S}(0) = \mathcal{S}^0 \,, \\ \dfrac{\partial \mathcal{S}(t)}{\partial t} = -\nabla_M E(\mathcal{S}(t)) \,. \end{cases} \tag{6}$$

2.2 Gradient Descent for Polyhedral Meshes

In practice, we often deal with discrete representations of the surface. Also whichever this representation is, computing exactly the gradient of the energy including directly the discrete representation of the surface is much more suitable than computing the gradient in an ideal continuous framework (with continuous surfaces) and then discretizing the continuous gradient accordingly to the discrete representation. In fact, the second strategy has two significant drawbacks. First, the continuous gradient lets often appear terms that are very difficult to discretize and which sometimes do even not really make sense in the case of discrete surfaces. For example the notion of surface curvature on mesh representation has a lot of different approximations. Second, since the discrete object we are practically handling is not deformed following the exact gradient but by an approximation of it, finally, we do not exactly know what we are minimizing. In this paper, we then compute the exact gradient of the energy including directly the discrete representation of the surface in the same way as [11, 12].

Let the mesh $\mathbf{X} = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$ be the piecewise planar polyhedral representation of $\mathcal{S}$. Vertices of $\mathbf{X}$ are denoted by $\mathbf{x}_k$ and $\mathcal{S}$ is deformed by moving vertices $\mathbf{x}_k$. We denote by $\phi_k : \mathcal{S} \to \mathbb{R}$ the piecewise linear, interpolating basis function such that $\phi_k(\mathbf{x}_k) = 1$ and $\phi_k(\mathbf{x}_i) = 0$ if $i \neq k$. Then any point $\mathbf{x}$ on the surface $\mathcal{S}$ can be defined such that $\forall \mathbf{x} \in \mathcal{S}$, $\mathbf{x} = \sum_k \mathbf{x}_k \phi_k(\mathbf{x})$, where we also have $\forall \mathbf{x} \in \mathcal{S}$, $\sum_k \phi_k(\mathbf{x}) = 1$.

Let the set $\{\mathbf{V}_k\}$ be a parametrized vector field defined on all the vertices $\mathbf{x}_k$ of the mesh $\mathbf{X}$ representing the surface deformation. $\{\mathbf{V}_k\}$ can be naturally extended on $\mathcal{S}$ by a piecewise linear vector field on $\mathcal{S}$. For convenience, we denote this extension $\mathbf{V} : \mathbf{V}(\mathbf{x}) = \sum_k \mathbf{V}_k \phi_k(\mathbf{x})$.

Then, evolving $\mathbf{X}$, by moving its vertices $\mathbf{x}_k$ following $\mathbf{V}_k$ is equivalent to deform the surface $\mathcal{S}$ by the dense deformation following $\mathbf{V}$.

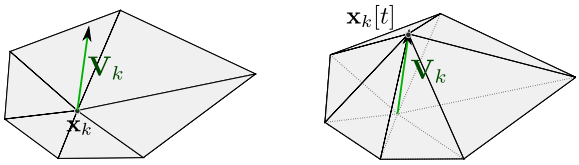

Fig. 2 Local evolution of a surface point $\mathbf{x}_k$ under induced velocity $\mathbf{V}_k$ used to compute the gradient.

The gradient of the energy is computed using shape gradient [7]. We consider the evolution of this energy according to the deformation $\mathbf{V}$. In other words, we assume that the vertices $\mathbf{x}_k[t]$ of $\mathbf{X}[t]$ are moving according to $\mathbf{x}_k[t] = \mathbf{x}_k^0 + t\mathbf{V}_k$ (See Figure 2). The method for computing the gradient of $E(\mathcal{S})$ consists in computing the directional derivative of $E(\mathcal{S}[t])$ for this deformation and then in rewriting it as a scalar product of $\mathbf{V}$, i.e. as $\langle \mathbf{V}, \mathbf{G} \rangle$, $\mathbf{G}$ being independant of $\mathbf{V}$. The obtained vector $\mathbf{G}$ is called the *gradient* and the energy necessarily decreases when deforming the

surface according to its opposite direction $-\mathbf{G}$. Indeed, for $\mathbf{x}_k[t] = \mathbf{x}_k^0 - t\mathbf{G}_k$, we have

$$[E \circ \mathcal{S}]'(0) = -\langle \mathbf{G}, \mathbf{G} \rangle \leq 0,$$

see [12].

Let us recall now that, as underlined by [6,12], the notion of gradient depends on the underlying scalar product. In this work we will only consider the L^2 inner product which has the advantage of taking into account the area of the triangles contrarily to the pointwise scalar product, which is necessary if the surface is not a regular mesh. Let $A = \{\mathbf{a}_k\}$ and $B = \{\mathbf{b}_k\}$ be vector fields on the mesh $\mathbf{X}$. Let a and b be their linear extension on the whole surface $\mathcal{S}$. Then their L^2 scalar product is:

$$\begin{aligned} \langle A, B \rangle_{L^2} &= \int_{\mathcal{S}} \langle a(\mathbf{x}), b(\mathbf{x}) \rangle \, \mathrm{d}s \\ &= \int_{\mathcal{S}} \left\langle \sum_k \mathbf{a}_k \phi_k(\mathbf{x}), \sum_k \mathbf{b}_k \phi_k(\mathbf{x}) \right\rangle \mathrm{d}s \\ &= A^T \, M \, B \, , \end{aligned} \tag{7}$$

where $M = \{m_{ij}\}$ is the mass matrix defined by $m_{ij} = Id_3 \int_{\mathcal{S}} \phi_i(x)\phi_j(x)\mathrm{d}s$. In the last line of Equation (7), A and B are the matricial representations of the vector fields. They are column vectors containing successively $\mathbf{a}_k$ and $\mathbf{b}_k$ vectors. Then the gradient becomes:

$$\nabla E(\mathbf{X}) = M^{-1} \frac{\partial E}{\partial \mathbf{X}}(\mathbf{X}) \, , \tag{8}$$

where

$$\frac{\partial E}{\partial \mathbf{X}}(\mathbf{X}) = \left[\frac{\partial E}{\partial \mathbf{x}_1}(\mathbf{X}) \; \frac{\partial E}{\partial \mathbf{x}_2}(\mathbf{X}) \ldots \frac{\partial E}{\partial \mathbf{x}_n}(\mathbf{X}) \right]'$$

corresponds to the gradient associated with the pointwise inner product $< A, B >= \sum_k \mathbf{a}_k \cdot \mathbf{b}_k$.

One classically approximates M by the diagonal mass lumping $\tilde{M}$, where $\tilde{m}_{ii}$ is the area of the Voronoi dual cell of $\mathbf{x}_i$ times the identity matrix Id_3, see e.g. [12]. It follows that the L^2 gradient descent flow is:

$$\begin{cases} \mathbf{X}[0] = \mathbf{X}^0 \, , \\ \dfrac{\partial \mathbf{X}[t]}{\partial t} = -\tilde{M}^{-1} \dfrac{\partial E}{\partial \mathbf{X}}(\mathbf{X}[t]) \, . \end{cases} \tag{9}$$

2.2.1 Triangle Mesh Representation and Notations

The previous results are valid for any polyhedral representation. Also in practice and in the following, we will focus on triangle representation that are easier to understand and more simple to handle using computers.

Let $\mathcal{S}_j$ be the j^{th} triangle of the mesh and $\mathbf{x}_k$ be a vertex of $\mathcal{S}_j$. Let us consider the parametrization on the triangle $\mathcal{S}_j$ such that

$$\mathbf{x}(\mathbf{u}) = \mathbf{x}_k + u \, \overrightarrow{\mathbf{x}_k \mathbf{x}_{k1}} + v \, \overrightarrow{\mathbf{x}_k \mathbf{x}_{k2}} \, , \tag{10}$$

where $\mathbf{x}_{k1}$ and $\mathbf{x}_{k2}$ are the two other vertices of the triangle $\mathcal{S}_j$ such that $(\mathbf{x}_k, \mathbf{x}_{k1}, \mathbf{x}_{k2})$ is a counter-clockwise triangle. Here, $\mathbf{u} = (u, v) \in T$, where

$$T = \{(u, v) | u \in [0, 1] \text{ and } v \in [0, 1 - u]\} \, .$$

Figure 3 illustrates this representation and parametrization. Too be rigorous, we should write $\mathbf{x}_k^j(\mathbf{u})$ instead of $\mathbf{x}(\mathbf{u})$, since the parametrization depends on j and k. Nevertheless, in order to simplify equations and improve the clarity of the paper, we remove these indexes in the rest of the paper. In the following, when we use $\mathbf{x}(\mathbf{u})$, the choice of the associated j and k is directly given by the context. On each triangle $\mathcal{S}_j$, we denote by ϕ_k the linear interpolating basis function that verifies $\phi_k(\mathbf{x}(\mathbf{u})) = (1 - u - v) \; , \forall (u, v) \in T$.

We denote by A_j the surface area of triangle $\mathcal{S}_j$ and by $\mathbf{n}_j$ its outward surface normal. A_j and $\mathbf{n}_j$ can easily be defined with respect to the triangle vertices such that:

$$A_j = \frac{1}{2} \, |\overrightarrow{\mathbf{x}_k \mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k \mathbf{x}_{k2}}| \quad \text{and} \quad \mathbf{n}_j = \frac{\overrightarrow{\mathbf{x}_k \mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k \mathbf{x}_{k2}}}{2 \, A_j} \, ,$$

where the operator $\wedge$ denotes the cross product. Indeed it is easy to show that the area surface measure on the surface $\mathrm{d}s$ can be written using the parametrization $\mathbf{u}$ such that:

$$\mathrm{d}s = 2 \, A_j \, \mathrm{d}\mathbf{u} \, . \tag{11}$$

In the following, we also denote by $\mathbf{e}_{j,k}$ the opposite edge of the vertex $\mathbf{x}_k$ in the triangle $\mathcal{S}_j$ such that $\mathbf{e}_{j,k} = \overrightarrow{\mathbf{x}_{k1} \mathbf{x}_{k2}}$.

$\mathcal{J}_k$ represents the set of triangles containing vertex $\mathbf{x}_k$ and the set $\mathcal{K}_j$ is the set of indexes of the three vertices of triangle $\mathcal{S}_j$ (See Figure 3.).

3 Gradient of Weighted Area Functionals

3.1 Continuous Case

Let $\mathcal{S} \in M$ be the surface to deform in order to minimize the following classical weighted area functional (1):

$$E(\mathcal{S}) = \int_{\mathcal{S}} g(\mathbf{x}, \mathbf{n}(\mathbf{x})) \mathrm{d}s \, ,$$

where g is a differentiable scalar function defined all over the surface. Then using shape gradient as described previously, one can rewrite the differential of $E(\mathcal{S})$ under a linear deformation V in order to find the expression of $\nabla E(\mathcal{S})$. As used in [15] and shown in [42], the gradient descent flow of the functional defined in (1) has the form:

$$\nabla E(\mathcal{S}) = \nabla \cdot (g_{\mathbf{n}} + g\mathbf{n}) \, , \tag{12}$$

where $g_{\mathbf{n}}$ is the gradient on the unit sphere $\mathbb{S}$.

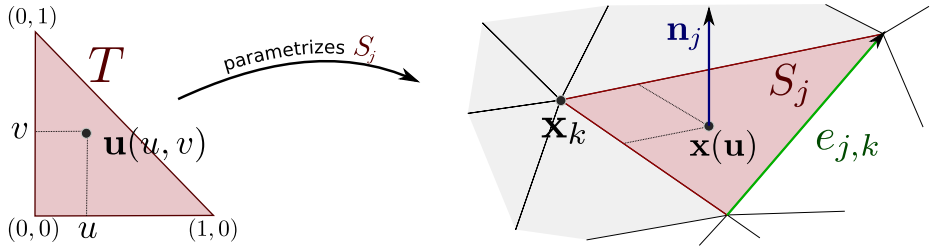

Fig. 3 Local parametrization of the discrete representation of the surface into a triangle mesh. Figure on the left shows the local parametrization $\mathbf{u}(u,v)$ of a surface point. On the right, we show the notations used in the paper where $\mathbf{x}_k$ is the current vertex of $\mathcal{S}$, $\mathcal{S}_j$ is the current facet around $\mathbf{x}_k$ and $\mathbf{e}_{j,k}$ is the opposite edge of $\mathbf{x}_k$ in $\mathcal{S}_j$. The gray area represent the set of triangles around $\mathbf{x}_k$ denoted by $\jmath_k$.

3.2 Gradient for Triangle Mesh-based Surfaces

In this paragraph, we consider the discretization on the surface and the gradient descent flow described in the previous section, when the surface is represented by a triangular mesh. Also we consider the case where the energy functional to be minimized is:

$$\begin{aligned} E(\mathcal{S}) &= \int_{\mathcal{S}} g(\mathbf{x}, \mathbf{n}(\mathbf{x}))\, \mathrm{d}s \\ &= \sum_j 2\, A_j \int_T g(\mathbf{x}(\mathbf{u}), \mathbf{n}_j)\, \mathrm{d}\mathbf{u}\,, \end{aligned} \tag{13}$$

where A_j and $\mathbf{n}_j$ are defined in Section 2.2.1.

Let us focus on the evolution of $E(S)$ under the induced velocity $\mathbf{V}$ on triangle $\mathcal{S}_j$ only. We have

$$E(\mathcal{S}_j[t]) = 2\, A_j[t] \int_T g(\mathbf{x}(\mathbf{u}) + t\mathbf{V}(\mathbf{x}(\mathbf{u})), \mathbf{n}_j[t])\, \mathrm{d}\mathbf{u}\,, \tag{14}$$

where $A_j[t]$ is the area of the triangle $\mathcal{S}_j[t]$ and $\mathbf{n}_j[t]$ is its normal at time t. By simple derivation we get

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} E(\mathcal{S}_j[t])\Big|_{t=0} &= 2\, A_j'[0] \int_T g(\mathbf{x}, \mathbf{n}_j)\, \mathrm{d}\mathbf{u} \\ &+ 2\, A_j \int_T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{n}_j) \cdot \mathbf{V}(\mathbf{x})\, \mathrm{d}\mathbf{u} \\ &+ 2\, A_j \int_T \nabla_{\mathbf{n}} g(\mathbf{x}, \mathbf{n}_j) \cdot \mathbf{n}_j'[0]\, \mathrm{d}\mathbf{u}\,. \end{aligned} \tag{15}$$

Above, in order to simplify equations, we have removed the dependency in $\mathbf{u}$ by writing $\mathbf{x}$ instead of $\mathbf{x}(\mathbf{u})$. In the sequel, we will use this abuse of notation. In order to rewrite $\frac{\mathrm{d}}{\mathrm{d}t} E(\mathcal{S}_j[t])\Big|_{t=0}$ as a scalar production of $\mathbf{V}$, we use also the fact that $\forall \mathbf{x} \in \mathcal{S}_j$, $\mathbf{V}(\mathbf{x}) = \sum_{k \in K_j} \mathbf{V}_k \phi_k(\mathbf{x})$, and then use the expressions of $A_j'[0]$ and $\mathbf{n}_j'[0]$ computed in appendices A.1 and A.2:

$$\begin{aligned} A_j'[0] &= \frac{\mathrm{d}}{\mathrm{d}t} A_j[t]\Big|_{t=0} = \frac{1}{2}\, (\mathbf{n}_j \wedge \overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}}) \cdot \mathbf{V}_k\,, \\ \mathbf{n}_j'[0] &= \frac{\mathrm{d}}{\mathrm{d}t} \mathbf{n}_j[t]\Big|_{t=0} = \frac{\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k - \left((\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k) \cdot \mathbf{n}_j\right) \mathbf{n}_j}{2\, A_j}\,. \end{aligned} \tag{16}$$

It follows that:

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} E(\mathcal{S}_j[t])\Big|_{t=0} = \sum_{k \in K_j} \mathbf{V}_k \cdot \Bigg\{ & \frac{\mathbf{n}_j \wedge \mathbf{e}_{j,k}}{2A_j} \int_{\mathcal{S}_j} g(\mathbf{x}, \mathbf{n}_j)\, \mathrm{d}s \\ & + \int_{\mathcal{S}_j} \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{n}_j) \phi_k(\mathbf{x})\, \mathrm{d}s \\ & - \frac{\mathbf{e}_{j,k}}{2\, A_j} \wedge \int_{\mathcal{S}_j} g_{\mathbf{n}}(\mathbf{x}, \mathbf{n}_j)\, \mathrm{d}s \Bigg\}, \end{aligned} \tag{17}$$

where we define $g_{\mathbf{n}} = \nabla_n g(\mathbf{x}, \mathbf{n}_j) - \langle \nabla_n g(\mathbf{x}, \mathbf{n}_j), \mathbf{n}_j \rangle\ \mathbf{n}_j$, where $\nabla_n g(\mathbf{x}, \mathbf{n}_j)$ is the gradient of g with respect to the second variable (i.e. $\mathbf{n} \in \mathbb{R}^3$).

It then immediately follows that

$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} E(\mathcal{S}[t])\Big|_{t=0} = \sum_k \mathbf{V}_k \cdot \Bigg[\sum_{j \in \jmath_k} & \left\{ \int_{\mathcal{S}_j} \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{n}_j) \phi_k(\mathbf{x})\, \mathrm{d}s \right. \\ & \left. - \frac{\mathbf{e}_{j,k}}{2A_j} \wedge \int_{\mathcal{S}_j} g(\mathbf{x}, \mathbf{n}_j)\, \mathbf{n}_j + g_{\mathbf{n}}(\mathbf{x}, \mathbf{n}_j)\, \mathrm{d}s \right\} \Bigg], \end{aligned} \tag{18}$$

Finally the part in brackets gives the k^{th} component of $\dfrac{\partial E}{\partial \mathbf{X}}$ and one has to use Equation (9) in order to get the gradient and optimize the mesh $\mathbf{X}$.

3.3 Comparison of the Continuous Gradient with the Gradient for Triangle Meshes

By looking at both gradients, one may note similarities. First it is worth it to notice that the discrete gradient is written with respect to well defined quantities that can be easily expressed, such as edges vectors, or triangle area.

On the other hand the continuous gradient has terms depending on the curvature for instance, which is well defined in the theory of continuous differential surfaces, but is hard to discretize on mesh representations. By making a piecewise linear surface assumption, the obtained gradient directly accounts for those intrinsic properties.

4 Gradient of Functionals defined on Visible Surface

As described in the introduction, computer vision applications are most likely able to deal with what the camera sees,

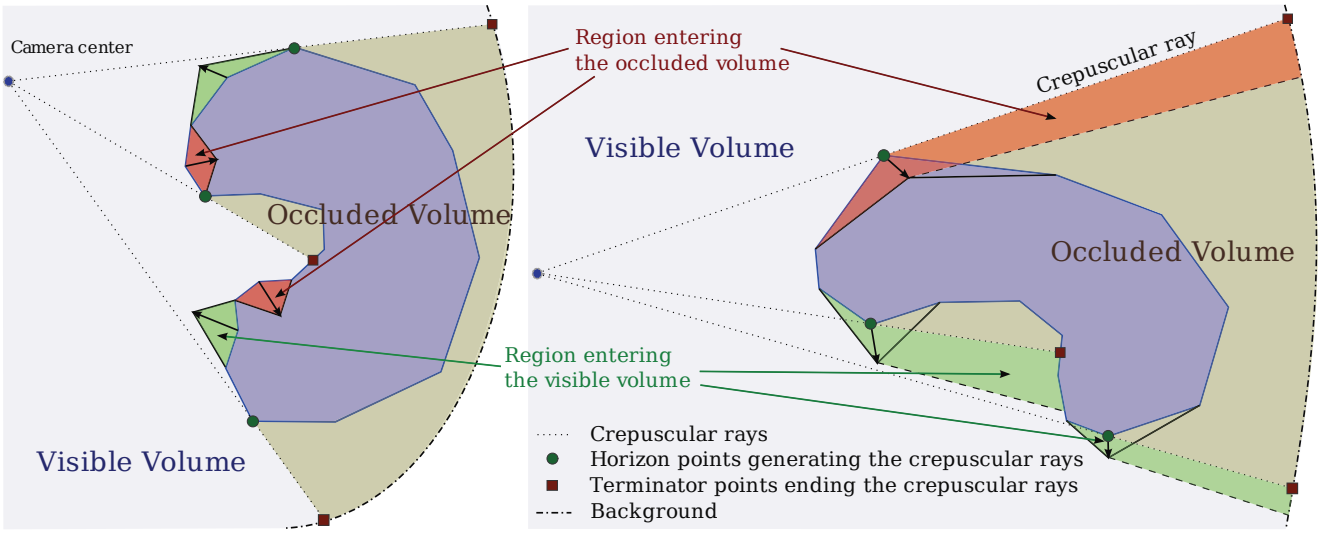

Fig. 4 Geometric representation of the change of visibility when moving the mesh. Contrary to the interior (left), movements of the horizon (right) strongly affect the movement of the visible interface between visible and occluded volumes by creating a movement of the crepuscular rays.

i.e. the projected image. Such projection imply $3D$ information such as depth and occlusion, and a camera model. An energy functional can be expressed accordingly as in Equation (5), where the energy can be defined on the visible interface (See also Figure 1). Often some quantities can be estimated (it can be for instance color, photometric normals, reflectance, etc) and used to compare them with data in the input images. In the sequel we denote such energies reprojection error functionals. As shown for example in [15,48], an accurate Bayesian formulation of computer vision problems (as e.g. in 3D reconstruction) yields the minimization of some reprojection error functionals instead of classical weighted area functionals.

Let us then consider the energy functional (5)

$$E(\mathcal{S}) = \int_{\mathcal{S}} \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x})) \cdot \mathbf{n}(\mathbf{x}) \; \nu_{\mathcal{S}}(\mathbf{x}) \, \mathrm{d}s \, .$$

In this section, we compute the gradient of this functional with respect to the shape $\mathcal{S}$.

In practice, the direction of the 3D vector $\mathbf{g}$ corresponds to the direction of the viewing ray. Thus the reprojection error functionals generally verify $\mathbf{g}(\mathbf{x}, \mathbf{n}_{\partial\mathcal{V}}(\mathbf{x})) \cdot \mathbf{n}_{\partial\mathcal{V}}(x) = 0$ for all points $\mathbf{x}$ on the horizon of the visibility interface ($\partial\mathcal{V}$ being the visibility interface and $\mathbf{n}_{\partial\mathcal{V}}$ its normal, see Figure 1). Then, by Gauss' divergence theorem, we can rewrite $E(\mathcal{S})$ as an integral over $\mathbb{R}^3$, see [14,15].

$$\begin{aligned} E(\mathcal{S}) &= \int_{\partial\mathcal{V}} \mathbf{g}(\mathbf{x}, \mathbf{n}_{\partial\mathcal{V}}(\mathbf{x})) \cdot \mathbf{n}_{\partial\mathcal{V}}(\mathbf{x}) \, \mathrm{d}s \\ &= -\int_{\mathbb{R}^3} \nabla \cdot \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x})) \; \nu_{\mathcal{S}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \, . \end{aligned} \tag{19}$$

Let $\mathcal{S}[t]$ be a variation of $\mathcal{S}$ such that $\mathcal{S}[t] = \mathcal{S} + t\,V$. By the product rule, the derivative of the energy with respect to t is

$$\begin{aligned} \frac{d}{dt} E(\mathcal{S}[t])\Big|_{t=0} = &- \int_{\mathbb{R}^3} \nabla \cdot \frac{d}{dt}\Big(\mathbf{g}(\mathbf{x}, \mathbf{n}[t](\mathbf{x}))\Big)\Big|_{t=0} \; \nu_{\mathcal{S}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \\ &- \int_{\mathbb{R}^3} \nabla \cdot \Big(\mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x}))\Big) \; \frac{d}{dt} \nu_{\mathcal{S}[t]}(\mathbf{x})\Big|_{t=0} \, \mathrm{d}\mathbf{x} \, . \end{aligned} \tag{20}$$

For notation simplicity we have denoted $\mathcal{S} = \mathcal{S}[0]$, $\mathbf{n} = \mathbf{n}[0]$.

In Section 4.2.1, we write the first integral of Equation (20) linearly with respect to $\mathbf{V}$. We then describe the corresponding part of the gradient of $E(\mathcal{S})$. We denote it $\mathbf{G}^{norm}$.

In the second integral of Equation (20), the normal does not depend on t. Thus, this term is the derivative of a quantity (which does not depend on $\mathcal{S}$ and does not vary in time) integrated over the visible volume. In other words, the second term of Equation (20) is the derivative (with respect to the shape $\mathcal{S}$) of an energy

$$\tilde{E}(\mathcal{S}) = \int_{\mathbb{R}^3} f(\mathbf{x}) \nu_{\mathcal{S}}(\mathbf{x}) \, \mathrm{d}\mathbf{x} \, ,$$

where $f(\mathbf{x})$ does not depend on $\mathcal{S}$;

$$f(\mathbf{x}) = -\nabla \cdot \Big(\mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x}))\Big) .$$

It is the derivative of a scalar field integrated over a visible volume.

Until now all equations are valid for both the continuous case and the discrete case. In order to find the gradient expression, the idea is to see what happens to the energy under an induced velocity on the surface. In the continuous case under a continuous vectorial field V, and under a piecewise linear vectorial field $\mathbf{V}$ defined by the discrete field $\{V_k\}$ in the other case. In both cases we have the following geometric interpretation of the changes in the energy when the surface is deforming.

Since this energy $\tilde{E}(\mathcal{S})$ is an integral over the visible volume, its variations are only due to the variations of the visible volume. Also, as illustrated by Figure 4 (reduced to the 2D case for simplicity), when the vertex $\mathbf{x}_k$ is moving according to $\mathbf{V}_k$ we have to separate two cases:

1. when all the triangles adjacent to $\mathbf{x}_k$ are visible, the variation of the visible volume is just the sum of the tetrahedron formed by the adjacent triangles and the moved point $\mathbf{x}_k + \mathbf{V}_k$ (see Figure 4, left). The corresponding gradient computation is detailed in Section 4.2.2. By replacing $f(\mathbf{x})$ by $-\nabla \cdot \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x}))$, this gives a second part of the gradient of $E(\mathcal{S})$; we denote it $\mathbf{G}^{int}$.
2. when $\mathbf{x}_k$ is generating occluding contours in images, i.e. when it is a *horizon* point, its movement affects the visibility of other points located behind it (called *terminator* points). So the variation of the visible volume is the sum of the first case term, plus the volume swept out by the crepuscular rays generated by the horizon movement (see Figure 4, right). The corresponding gradient computation is detailed in Section 4.2.3. Again, by replacing $f(\mathbf{x})$ by $-\nabla \cdot \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x}))$, this gives a third part of the gradient of $E(\mathcal{S})$ that we denote it $\mathbf{G}^{horiz}$.

Now let us first remind the expression of the gradient in the continuous case (Section 4.1) and then compute it when using triangle meshes (Section 4.2).

4.1 Gradient in the Continuous Case

The differential of the energy (5) in the case of continuous surfaces is the work of Gargallo [14] and is:

$$\nabla \cdot \Big(\mathbf{g}_{\mathbf{n}} \cdot \mathbf{n} + \mathbf{g}\Big)\nu_{\mathcal{S}} - \mathbf{x}^t \nabla \mathbf{n}\,(\mathbf{g} - \mathbf{g}')\delta(\mathbf{x} \cdot \mathbf{n})\nu_{\mathcal{S}}\,, \qquad (21)$$

where δ is a Dirac distribution function and $\mathbf{g}'$ is the value of $\mathbf{g}$ at the terminator of the current point. In the presence of a discrete surface, terms like the curvature $\nabla \mathbf{n}$ are difficult to handle and have to be approximated. The following section shows that by computing the gradient with respect to the true representation of the surface, one can use a new formulation using intrinsic surface properties.

4.2 Gradient for Triangle Meshes

Here, we derive the exact gradient (with respect to the shape) of these functionals in the case where the surface is represented by a triangular mesh. Here, contrary to our previous conference article [8], the function $\mathbf{g}$ may also depends on the normal of the surface.

Then we can summarize the way gradient descent is performed in that case. The variation of an energy that depends on the visibility and the surface normal can be decomposed into three different cases. The three terms are:

1. the one due to the change in the normal $\mathbf{G}^{norm}$ (Equation (25)) that corresponds to Section 4.2.1,
2. the term due to the movement of points on the fully visible areas $\mathbf{G}^{int}$ (Equation (30)) of Section 4.2.2,
3. and finally the term due to the movement of points on occluding contours $\mathbf{G}^{horiz}$ (Equation (36)) that makes global (in the sense on the whole surface) changes of the energy (Described in Section 4.2.3).

4.2.1 Term due to the Variation of the Normal

In this section, we are going to rewrite the first integral term appearing in equation (20), linearly with respect to $\mathbf{V}$. This will directly give us the first term of the gradient of $E(\mathcal{S})$ defined in (5). First, using Gauss' divergence theorem again, let us rewrite it as an integral over the surface. This gives:

$$\begin{aligned}
&-\int_{\mathbb{R}^3} \nabla \cdot \frac{d}{dt}\mathbf{g}(\mathbf{x}, \mathbf{n}[t](\mathbf{x}))\Big|_{t=0} \nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
&= \int_{\mathcal{S}} \frac{d}{dt}\mathbf{g}(\mathbf{x}, \mathbf{n}[t](\mathbf{x}))\Big|_{t=0} \cdot \mathbf{n}(\mathbf{x})\,\nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}s \\
&= \sum_j \int_{\mathcal{S}_j} \frac{d}{dt}\mathbf{g}(\mathbf{x}, \mathbf{n}_j[t])\Big|_{t=0} \cdot \mathbf{n}_j\,\nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}s \\
&= \sum_j \int_{\mathcal{S}_j} (D_{\mathbf{n}}\mathbf{g}(\mathbf{x}, \mathbf{n}_j)\mathbf{n}_j'[0]) \cdot \mathbf{n}_j\,\nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}s\,,
\end{aligned} \qquad (22)$$

where $D_{\mathbf{n}}$ is the differential with respect to the second variable and where $\mathbf{n}_j'[0]$ is the derivative of the normal $\mathbf{n}_j[t]$ of the triangle $\mathcal{S}_j[t]$ at time $t = 0$. Here we have assumed that the reprojection error functional verifies

$$\left(\frac{d}{dt}\mathbf{g}(\mathbf{x}, \mathbf{n}_{\partial\mathcal{V}}[t](\mathbf{x}))\right) \cdot \mathbf{n}_{\partial\mathcal{V}}(x) = 0$$

for all points $\mathbf{x}$ on the horizon lines of the visibility interface; it is generally the case in practice, see [14, 15].

The computation of $\mathbf{n}_j'[0]$ is detailed in appendix A.2. We have

$$\mathbf{n}_j'[0] = \frac{1}{2A_j}\left(\left(\sum_{k\in K_j} \mathbf{e}_{j,k} \wedge \mathbf{V}_k\right) - \left(\left(\sum_{k\in K_j} \mathbf{e}_{j,k} \wedge \mathbf{V}_k\right) \cdot \mathbf{n}_j\right)\mathbf{n}_j\right). \qquad (23)$$

So

$$\begin{aligned}
&\int_{\mathbb{R}^3} \nabla \cdot \frac{d}{dt}\mathbf{g}(\mathbf{x}, \mathbf{n}[t](\mathbf{x}))\Big|_{t=0} \nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}\mathbf{x} \\
&\quad= \sum_j \int_{\mathcal{S}_j} \mathbf{n}_j'[0] \cdot D_{\mathbf{n}}\mathbf{g}(\mathbf{x}, \mathbf{n}_j)^T \mathbf{n}_j\,\nu_{\mathcal{S}}(\mathbf{x})\mathrm{d}s \\
&\quad= \sum_k \mathbf{V}_k \cdot \Big[\sum_{j\in J_k} (-1)\frac{\mathbf{e}_{j,k}}{2\,A_j} \wedge \\
&\qquad \int_{\mathcal{S}_j} P_{n_j{}^\top}(D_{\mathbf{n}}\mathbf{g}(\mathbf{x}, \mathbf{n}_j)^T \mathbf{n}_j)\,\nu_{\mathcal{S}}(\mathbf{x})\,\mathrm{d}s\Big]\,,
\end{aligned} \qquad (24)$$

where $P_{n_j\top}(D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j)$ is the projection on the orthogonal plane to $\mathbf{n}_j$ of $D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j$. $\mathbf{e}_{j,k}$ is defined in appendix A.1. Roughly $\mathbf{e}_{j,k}$ is the opposite edge of the vertex $\mathbf{x}_k$ in the triangle $\mathcal{S}_j$. Finally, writing the previous expression on the mesh parametrization $\mathbf{x}(\mathbf{u})$ yields: which can be rewritten as :

$$\mathbf{G}_k^{norm} = \sum_{j\in\jmath_k} \mathbf{e}_{j,k} \wedge \int_T P_{n_j\top}(D_{\mathbf{n}}\mathbf{g}(\mathbf{x}(\mathbf{u}),\mathbf{n}_j)^T\mathbf{n}_j)\,\nu_{\mathcal{S}}(\mathbf{x}(\mathbf{u}))\,\mathrm{d}\mathbf{u}. \quad (25)$$

4.2.2 Term Due to the Tetrahedra of the Visible Adjacent Triangles

In this paragraph, let us focus on the variation of the energy caused by the variation of the visible volume corresponding to the tetrahedra formed by the visible adjacent triangles of vertex $\mathbf{x}_k$ and $\mathbf{x}_k + \mathbf{V}_k$. (See Figure 2)

In the sequel, we denote $\mathcal{S}_j$ the j^{th} triangle of the mesh. Following the variation of the energy caused by the visible adjacent triangles of $\mathbf{x}_k$, we have

$$\tilde{E}(\mathcal{S}[t]) - \tilde{E}(\mathcal{S}[0]) = \sum_j \varpi_{j,\mathbf{V}_k} \int_{Vol[j,\mathbf{V},t]} f(\mathbf{x})\,\mathrm{d}\mathbf{x} + \dots\,, \quad (26)$$

where $Vol[j,\mathbf{V},t]$ is the volume of the tetrahedron formed by the vertices of the *visible* triangle $\mathcal{S}_j$ and the point $\mathbf{x}_k + \mathbf{V}_k$. (See Figure 2) The sign $\varpi_{j,\mathbf{V}_k}$ specifies if matter has been added to or removed from the object volume and it is equal to the sign of $\mathbf{n}_j \cdot \mathbf{V}_k$, where $\mathbf{n}_j$ is the outward surface normal of triangle $\mathcal{S}_j$. The dots part "..." on the right of equation (26) is null except if the vertex $\mathbf{x}_k$ is a horizon point, which means that this point is on the occluding contour; this additional part will be detailed in the next paragraph.

Now, we parametrize the volume $Vol[j,\mathbf{V},t]$ by the point $\mathbf{x}(u,v,w) = \mathbf{x}(u,v) + w\ t\mathbf{V}_k$, where $\mathbf{x}(u,v) = \mathbf{x}(\mathbf{u})$ parametrizes the triangle $\mathcal{S}_j$ as defined previously and shown in figure 3; The local parametrization is such that $\mathbf{u}(u,v) \in T = \{(u,v) \mid u \in [0,1] \text{ and } v \in [0,1-u]\}$ and $w \in [0,1-u-v]$. By a change of variables, Equation (26) becomes

$$\sum_j \varpi_{j,\mathbf{V}_k} \int_T \int_0^{\phi_k(\mathbf{u})} \Big\{ f(\mathbf{x}(\mathbf{u}) + wt\mathbf{V}_k) \times |det(\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}}, \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}}, t\mathbf{V}_k)| \Big\}\,\mathrm{d}w\mathrm{d}\mathbf{u} + \dots \quad (27)$$

Let A_j be the area of the triangle $\mathcal{S}_j$. It is easy to show that $\varpi_{j,\mathbf{V}_k}|det(\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}}, \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}}, t\mathbf{V}_k)| = 2t\,A_j\mathbf{V}_k \cdot \mathbf{n}_j$. Then Equation (27) becomes

$$2t\,\mathbf{V}_k \cdot \left[\sum_j A_j\mathbf{n}_j \int_T \int_0^{\phi_k(\mathbf{u})} f(\mathbf{x}(\mathbf{u}) + wt\mathbf{V}_k)\,\mathrm{d}w\mathrm{d}\mathbf{u} \right] + \dots \quad (28)$$

It follows that the limit of $\frac{\tilde{E}(\mathcal{S}[t]) - \tilde{E}(\mathcal{S}[0])}{t}$ when t tends to zero is

$$\mathbf{V}_k \cdot \left[2 \sum_j A_j\mathbf{n}_j \int_T f(\mathbf{x}(\mathbf{u}))\,\phi_k(\mathbf{x}(\mathbf{u}))\,\mathrm{d}\mathbf{u} \right]. \quad (29)$$

Now the derivative of the energy can be expressed as a scalar product between the velocity $\mathbf{V}_k$ and a quantity that corresponds to the gradient as explained previously. The part in square brackets corresponds then to the interior term of the gradient of $E(\mathcal{S})$ with respect to $\mathbf{x}_k$:

$$\mathbf{G}_k^{int} = 2 \sum_j A_j\mathbf{n}_j \int_T \nabla\cdot\mathbf{g}(\mathbf{x}(\mathbf{u}),\mathbf{n}_j)\,\phi_k(\mathbf{x}(\mathbf{u}))\,\mathrm{d}\mathbf{u}\,, \quad (30)$$

where the sum is on all the (completely) visible triangles containing vertex $\mathbf{x}_k$.

4.2.3 Term due the movement of the crepuscular cone

In this section we are then going to compute the additional term which appears when $\mathbf{x}_k$ is a horizon point on the occluding contour. In this case the energy variation during a surface movement is due to the volume created by the crepuscular cones. This movement is not affected by the dependency in $\mathbf{n}(\mathbf{x})$ since this is purely due to the visibility changes.

Let $H_{k,j}$ be the vector such that $[\mathbf{x}_k, \mathbf{x}_k + H_{k,j}]$ is the edge of the triangle $\mathcal{S}_j$ generating the horizon. The volume corresponding to the movement of the horizon can be parametrized by the points $\mathbf{y}(u,v)$ of the triangle $\{\mathbf{x}_k, \mathbf{x}_k + H_{k,j}, \mathbf{x}_k + v\ t\mathbf{V}_k\}$ generated by the movement of the horizon. More rigorously, it can be parametrized as the set of points $\mathbf{x}(u,v,r) = r\ \mathbf{y}(u,v)$ where $\mathbf{y}(u,v) = \mathbf{x}_k + u\ H_{k,j} + v\ t\mathbf{V}_k$; r corresponds to the depth of $\mathbf{x}$ in the view point direction; $r \in [1, \mathcal{T}_{(u,v)}]$, where it corresponds to the terminator of $\mathbf{x}_k$ when $r = \mathcal{T}_{(u,v)}$.

Let us note that $\mathbf{y}(u,v)$ depends on t; we emphasize this dependency by denoting $\mathbf{y}_t(u,v)$. By a change of variable, we get $\tilde{E}(\mathcal{S}[t]) - \tilde{E}(\mathcal{S}[0]) =$

$$\dots - \frac{1}{2} \int_T \int_{r\in[1,\mathcal{T}_{(u,v)}]} \Big\{ f(r\ \mathbf{y}_t(u,v)) (H_{k,j} \wedge\ t\mathbf{V}_k) \cdot \frac{\mathbf{y}_t(u,v)}{|\mathbf{y}_t(u,v)|} r^2 \Big\} \mathrm{d}r\mathrm{d}u\mathrm{d}v\,, \quad (31)$$

where "..." corresponds to the part described in the previous paragraph. It follows that the limit of $\frac{\tilde{E}(\mathcal{S}[t]) - \tilde{E}(\mathcal{S}[0])}{t}$ when t tends to zero is the term in Equation (30) plus the following term

$$-\frac{1}{2} \int_T \int_{[1,\mathcal{T}_u]} f(r\ \mathbf{y}(u))\ (H_{k,j} \wedge \mathbf{V}_k) \cdot \frac{\mathbf{y}(u)}{|\mathbf{y}(u)|} r^2 \mathrm{d}r\mathrm{d}u\mathrm{d}v\,,$$

(32)

where $\mathbf{y}(u) = \mathbf{x}_k + u\, H_{k,j}$ and $\mathcal{T}_u = \mathcal{T}_{(u,0)}$. Let us denote $\mathcal{L}(u)$ such that

$$\mathcal{L}(u) = \int_{[1,\mathcal{T}_u]} f(r\,\mathbf{y}(u)) r^2 \mathrm{d}r \,. \tag{33}$$

The right-hand part of equation (32) can be rewritten as

$$-\mathbf{V}_k \cdot \frac{1}{2} \int_{u\in[0,1]} \mathcal{L}(u) \left(\frac{\mathbf{y}(u)}{|\mathbf{y}(u)|} \wedge H_{k,j} \right) (1-u) \mathrm{d}u \,, \tag{34}$$

where $\mathcal{L}(u) = \int_{1,\mathcal{T}_u} f(r\,\mathbf{y}(u)) r^2 dr$. Here we have $f(\mathbf{x}) = -\nabla \cdot \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x}))$. Here we can explicit $\mathcal{L}(u)$ as :

$$\begin{aligned} \mathcal{L}(u) &= -\int_{1,\mathcal{T}_u} \nabla g(r\mathbf{y}(u)) \cdot \frac{\mathbf{y}(u)}{\mathbf{y}(u)_z^3} \mathrm{d}r \\ &= -\big[g(T(\mathbf{y}(u))) - g(\mathbf{y}(u)) \big] \frac{1}{\mathbf{y}(u)_z^3} \,, \end{aligned} \tag{35}$$

where we have denoted $\mathbf{g}(\mathbf{x}) = g(\mathbf{x}, \mathbf{n}) \dfrac{\mathbf{x}}{\mathbf{x}_z^3}$ and $T(\mathbf{y}(u))$ is the point located behind $\mathbf{y}(u)$ in the direction of the viewpoint (i.e. its terminator point).

This gives the third part of the k^{th} component of the gradient of $E(\mathcal{S})$ defined in Equation (5):

$$\begin{aligned} \mathbf{G}_k^{horiz} = \sum_{H_{k,j}} \frac{1}{2} \int_0^1 \Big\{ & \big(g\left(T\left(\mathbf{y}(u)\right)\right) - g\left(\mathbf{y}(u)\right) \big) \\ & \times \left(\frac{\mathbf{y}(u)}{|\mathbf{y}(u)|^4} \wedge H_{k,j} \right) (1-u) \Big\} \mathrm{d}u \,, \end{aligned} \tag{36}$$

where the sum is on the edges containing $\mathbf{x}_k$ and which generate horizons. $H_{k,j}$ are the horizon edges around vertex $\mathbf{x}_k$.

4.2.4 Total Gradient Descent Flow

Finally, as explained in Section 2.2, the gradient descent flow depends on the choice of an inner product for the gradient. Here we use the L^2 gradient that allow to account for triangle area variations. As described previously and shown in [6, 12], changing the metric can result in more coherent gradient flows, but this study is out the scope of this paper. In this case the gradient descent flow for a point $\mathbf{x}_k$ corresponding to our energy functional defined on a visible domain is:

$$\begin{cases} \mathbf{x}_k(0) = \mathbf{x}_k^0 \,, \\ \dfrac{d\mathbf{x}_k}{dt} = -\dfrac{1}{A_k} \left\{ \mathbf{G}_k^{int} + \mathbf{G}_k^{horiz} + \mathbf{G}_k^{norm} \right\} \,. \end{cases} \tag{37}$$

5 Applications

This section shows how the gradient descent framework presented in the previous sections can be used to perform mesh evolution. In particular several concrete examples that are commonly used in computer vision and graphics are presented. For more clarity and comparison purposes, the following examples use the L^2-gradient.

In the following examples, the mesh evolution algorithm that includes the remeshing and the topology changes is done using the Delaunay topology-adaptive meshes proposed by [33] written using CGAL [4].

We are performing minimization via gradient descent, which starts from some initialization. In most examples the initial shape is the visual hull of the scene, where we assume silhouettes can be easily computed or are given, or that stereoscopic segmentation [47] can be easily performed. It is also important to notice that the horizon term is performed only on existing contour generators, so that the topology has to be close to the final one, or that contour generators can be created thanks to the interior term. Also, since it is based on energy differences one may add more or less importance to it. In particular in the following experiments, we add a weight λ_H to control this amount, which is empirically determined. If λ_H is too big, the horizon term will make the contour generators of the surface oscillates around their corresponding occluding contours.

Finally, all firgures shown in this section are displayed using flat shading rendering.

5.1 Mean Curvature Flow

One of the most commonly used gradient flow is the mean curvature flow, that minimizes the surface area. This energy is often used to perform mesh smoothing, or more often as a smoothing energy term. The associated energy functional is simply

$$E(\mathcal{S}) = \int_{\mathcal{S}} \mathrm{d}s \,, \tag{38}$$

and its associated continuous gradient is $\nabla E = \kappa$, where κ is the surface's curvature.

However, one needs some approximations in order to apply it to discrete surface representations since notion of discrete curvature is not clear. On triangular meshes, different approximated flows have been proposed like laplacian approximation or the umbrella operator. Meyer et al [30] have computed the discrete gradient flow that minimizes Equation (38) with respect to the triangle mesh representation. It is easy to show that the same result can be obtained using the presented approach. Following Section 2, we have

$$\frac{d}{dt} E(\mathcal{S}[t]) \Big|_{t=0} = \sum_k \mathbf{V}_k \cdot \sum_{j\in\mathcal{J}_k} \frac{1}{2} \mathbf{n}_j \wedge \mathbf{e}_{j,k} \,. \tag{39}$$

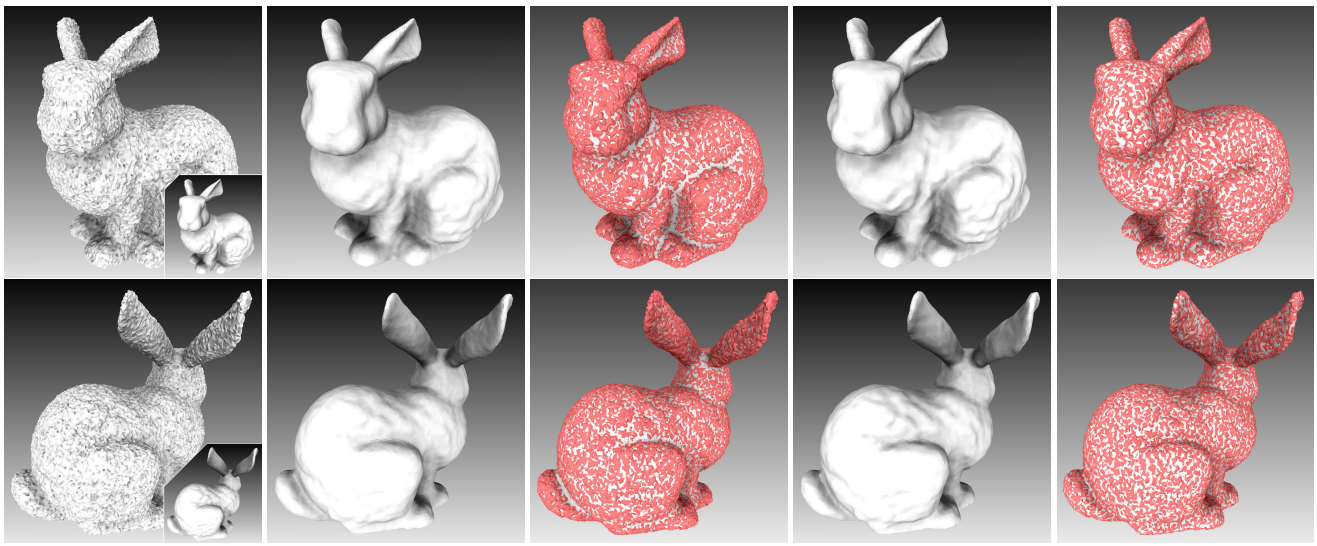

Fig. 5 Evolution of smoothing algorithms on the Stanford Bunny data. From left to right: input noisy data; denoised mesh obtained using the mean curvature flow (MCF) of Section 5.1; MCF result versus the noisy input (in red); denoised mesh obtained using the normal smoothing algorithm (NS) of Section 5.2; NS result versus the noisy input (in red). Smoothing the normals help preserving details better than the mean curvature flow and does not over-smooth the result at concavities and convexities as much as the mean curvature flow. (Color Online)

The evolution of one vertex k then follows

$$\frac{d\mathbf{x}_k}{dt} = \frac{1}{A_k} \sum_{j \in \jmath_k} \frac{1}{2}\, \mathbf{e}_{j,k} \wedge \mathbf{n}_j \;, \tag{40}$$

where A_k is the area of the neighborhood $\jmath_k$ of vertex k. Note that this result is exactly the same as the results in [30], but is just expressed differently. The given formulation can be useful for applications where edges and surface normals have been previously computed. Figure 5 illustrates this algorithm on the Stanford Bunny data [1].

5.2 Normal Field Integration

One of the applications is to align a surface with respect to an external normal field. For instance, in 3D reconstruction, one recover the surface by integrating photometric normals [5,20,43]. Let $\mathbf{h}$ be a unit vector field in $\mathbb{R}^3$. Integrating this vector field $\mathbf{h}$ such that the surface normals $\mathbf{n}$ correspond to it involves minimizing the following functional

$$\begin{aligned} E(\mathcal{S}) &= \frac{1}{2} \int_{\mathcal{S}} |\mathbf{h} - \mathbf{n}|^2 \, \mathrm{d}s \\ &= \sum_j \int_{\mathcal{S}_j} (1 - \mathbf{h} \cdot \mathbf{n}_j) \, \mathrm{d}s \;. \end{aligned} \tag{41}$$

As explained in Section 3, it is easy to show that the following gradient descent flow is:

$$\frac{d\mathbf{x}_k}{dt} = \frac{1}{A_k} \sum_{j \in \jmath_k} \frac{1}{2}\, \mathbf{e}_{j,k} \wedge \int_T (\mathbf{n}_j - \mathbf{h}) \, \mathrm{d}\mathbf{u} \;. \tag{42}$$

5.2.1 Normal Smoothing

As an example we show now that this approach can be easily applied to normal smoothing with efficiency, and is barely more time consuming than the mean curvature flow. Let $\mathbf{h}_j$ be the weighted average normal of the triangle j and its neighborhood N_j:

$$\mathbf{h}_j = \frac{\sum_{l \in N_j} \alpha_l \, \mathbf{n}_l}{\| \sum_{l \in N_j} \alpha_l \, \mathbf{n}_l \|} \;, \tag{43}$$

where the weights α_l can be chosen depending on the application. It could for instance be the area A_l of the triangle l, a Gaussian weight or more simply one can set $\alpha_l = 1$. Therefore the energy functional to be minimized can be expressed as:

$$E(\mathcal{S}) = \sum_j \int_{\mathcal{S}_j} (1 - \mathbf{h}_j \cdot \mathbf{n}_j) \, \mathrm{d}s \;, \tag{44}$$

where $\mathbf{h}_j$ is constant over the surface. Note that we cannot directly apply the results presented in Section 3.2 to energy (44) since it depends on the normal at several surface points. To minimize energy (44), we consider $\mathbf{h}_j$ to be fixed while updating the surface. In other words, we alternately update the surface and the vector field $\mathbf{h}$. The gradient descent flow corresponding to the normal smoothing energy with respect to the surface is

$$\frac{d\mathbf{x}_k}{dt} = \frac{1}{A_k} \sum_{j \in \jmath_k} \frac{1}{2}\, \mathbf{e}_{j,k} \wedge (\mathbf{n}_j - \mathbf{h}_j) \;. \tag{45}$$

Note the similarity of the above equation with the mean curvature flow (Equation (40)), and the fact that the gradient

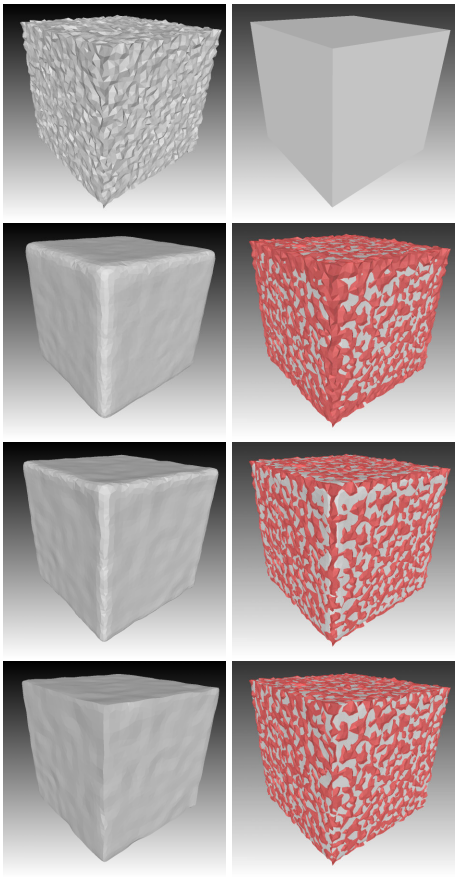

Fig. 6 Evolution of smoothing algorithms on simple cube data. From top to bottom: noisy input and corresponding original surface; mean curvature flow smoothing; normal smoothing; median filtering. The red part corresponds to the input noisy mesh being displayed together with the results. (Color Online)

flow with respect to the surface is barely more time consuming. Results are shown in Figure 5, where the evolution is stopped once the noise is no longer visible. The figure also shows a comparison between the mean curvature flow. In particular, when displayed with the input noisy mesh, one may notice the different density of noise in the mesh (shown in red). Since this is a Gaussian noise, the quantity of noise should be uniform all over the surface, as it is almost the case for normal smoothing. However, proportion of the visible noisy mesh in the mean curvature flow example is much denser in the convex parts and disappears in concavities. While mean curvature flow is popular for surface regularization and smoothing, normal smoothing preserve details better and does not over-smooth as much as the mean curvature flow.

5.2.2 Median Filtering

Similarly, one may choose the external vector field $\mathbf{h}_j$ to be the median vector of the neighborhood N_j of $\mathcal{S}_j$ to perform median filtering on the mesh. $\mathbf{h}_j$ can be computed by first computing the spherical coordinates (θ_m, ϕ_m) of each normal $\mathbf{n}_m \in N_j$. Then by sorting the spherical coordinates (θ_m, ϕ_m) along each component, one can obtain the median normal by getting back to cartesian coordinates. Results on a noisy cube and comparison with the previous smoothing algorithms are shown in Figure 6.

5.3 Multi-view Stereovision

In this example, we detail how the gradient defined in Section 4 can be used in three-dimensional surface reconstruction from images.

Multi-view stereo is the problem of recovering the shape of scenes using cameras, by assuming that matching or correspondences between different views can be obtained (by usually considering Lambertian constant brightness assumptions). Given a set of images of a scene taken from different camera positions, the goal is to reconstruct the shape $\mathcal{S}$, and optionally the appearance, of the object. Since it is the inverse problem of image rendering, this problem can be modeled in a Bayesian framework by minimizing the difference between the images of the reconstructed model and the observed ones, i.e. the reprojection error. The correct variational interpretation of the Bayesian analysis yields a minimization of some energy functional defined on the images [15,23,34,39,47] and requires the visibility of the surface points to be carefully accounted for. Also, only recently, some authors [15,47] manage to rigorously and fully account for visibility in the optimization process. These works have proved the interests of doing so by showing that this forces the contours generators of the reconstructed surface to match with the apparent contours in the observed images. This makes the use of additional energy terms [19,37,44] unnecessary and significantly reduces the minimal surface bias.

The multi-view stereovision algorithm we propose here is based on the same modeling as the one proposed by [15]. As Gargallo et al. we fully account for visibility, and then acquire the same benefits (matching of the apparent contours and removing the minimal surface bias). The significant difference with respect to [15] relies on the surface representation. As [47], Gargallo et al. use the Eulerian formulation and implement their algorithm in the level-set framework. Here, we use a triangle mesh-based representation, as described in the previous sections. Below, we describe the modeling of the problem and then detail the exact gradient for our discrete representation obtained by using the results developed in Section 4.2.

5.3.1 Modeling of the Reprojection Error

In order to be able to compare the whole observed images (data) with the images generated by the model, it is crucial to define and model the background. Also, as shown by [15,47], this allows us to be sure that the estimated foreground surface does not shrink to an empty set (which is the global optimum for most cost functionals used in other work). Moreover accounting for the background allows the contours generated by the recovered object to match with the occluding contours of the images. Whereas most of the previous work assumes that the background is known (e.g. simply modeling it as uniformly black, or by exploiting given silhouette images), here we also estimate the background images $B_i : \mathcal{I}_i \rightarrow \mathbb{R}^3$, under the single assumption that these images are smooth, similarly as [47]. Here i corresponds to the index of the camera and $\mathcal{I}_i$ is its image domain.

Now, let us assume that the scene surface $\mathcal{S}$ is Lambertian and the illumination static. Let $C : \mathcal{S} \rightarrow \mathbb{R}^3$ be the radiance function that associates colors to the points on the surface. Ideally, the color $I_i(\mathbf{p})$ observed at pixel $\mathbf{p}$ of image I_i should be equal to the color $C(\pi_{i,\mathcal{S}}^{-1}(\mathbf{p}))$ of its back-projection $\pi_{i,\mathcal{S}}^{-1}(\mathbf{p})$ onto the surface or, in the case where $\mathbf{p} \notin \pi_i(S)$, to the color $B_i(\mathbf{p})$ of the same pixel on the background images (π_i denoting the projection associated with camera i). Thus, the reprojection error of the surface is

$$E_{data} = \frac{1}{2}\sum_i \left[\int_{\pi_i(\mathcal{S})} \left(I_i(\mathbf{p}) - C(\pi_{i,\mathcal{S}}^{-1}(\mathbf{p}))\right)^2 \mathrm{d}\mathbf{p} + \int_{\mathcal{I}_i - \pi_i(\mathcal{S})} \left(I_i(\mathbf{p}) - B_i(\mathbf{p})\right)^2 \mathrm{d}\mathbf{p} \right] . \quad (46)$$

Finally, in order to well pose the problem, we use as a prior on $\mathcal{S}$ an additional smoothing area energy. As described previously, smoothing the normals gives a better prior than the commonly used mean curvature flow. It helps not to over-smooth the surface and to preserve geometric details better. The considered energy is

$$E_{R\mathcal{S}} = \int_{\mathcal{S}} (1 - \mathbf{h}(\mathbf{x}) \cdot \mathbf{n}) \, \mathrm{d}s \, ,$$

where $\mathbf{h}$ is the average normal of the considered surface point. We also assume that the background images are smooth by adding the total variation term

$$E_{RB} = \sum_i \int_{\mathcal{I}_i} |\nabla B_i(\mathbf{p})| \mathrm{d}\mathbf{p}.$$

The total variation helps preserving edges and does not over-smooth the object boundaries. In practice, the background images can be identified before the surface optimization by giving the silhouettes or doing some stereoscopic segmentation.

5.3.2 Minimization of the Total Energy

For optimizing our total energy

$$E_{total} = E_{data} + \lambda_{\mathcal{S}} E_{R\mathcal{S}} + \lambda_B E_{RB} \, , \quad (47)$$

we perform gradient descents alternately with respect to B_i and $\mathcal{S}$.

The computation of the gradients with respect to B_i is classical since it is image-based [47]. For a fixed shape $\mathcal{S}$ and a fixed C, we have

$$\nabla E_{total}(B) = -\sum_i \left[(I_i - B_i)(1-h) + \lambda_B \nabla \cdot \frac{\nabla B_i}{\|\nabla B_i\|} \right], \quad (48)$$

where h is the characteristic function that indicates if $\mathbf{u}$ is covered by the projection of the surface $\mathcal{S}$ ($h(\mathbf{p}) = 1$) or not ($h(\mathbf{p}) = 0$, $\mathbf{p}$ being explained by the background); λ is the smoothness parameter; $C(\mathbf{x})$ is computed by taking the mean color of the projection in the image I_i where $\mathbf{x}$ is visible.

Let us now detail the gradient of E_{total} with respect to $\mathcal{S}$. Since E_{RB} do not depend on $\mathcal{S}$ and $E_{R\mathcal{S}}$ is classical [9, 47]), the main point is to compute the gradient of E_{data} with respect to $\mathcal{S}$. To apply the results presented in the previous sections, we first need to rewrite the energy E_{data} as an integral over only the visible surface. For simplicity of notation, we are going to give the gradient for a single camera and so we remove the dependency on i. For several cameras, the gradient will be the sum of the gradients associated with each camera.

As explained in the introduction, the first step, to be able to apply our previous results, is to rewrite the energy as an integral over the surface instead of the image. This change of variable implies the geometric model of the camera which we assume to be a pinhole perspective camera model in this work. It also involves adapting the measure on the surface [39] and in counting only the visible points [15,34,47]. This can be achieved by $d\mathbf{u} = -\frac{\mathbf{x}\cdot\mathbf{n}(\mathbf{x})}{\mathbf{x}_z^3}\nu_{\mathcal{S}}(\mathbf{x})\mathrm{d}s$ where $\mathrm{d}s$ is the classical surface area measure and $\mathbf{x}_z$ is the depth of $\mathbf{x}$. Thus, by using the separation technique proposed by [47], the energy functional becomes (for a *single* image):

$$\begin{aligned} E_{data}(\mathcal{S}) &= -\int_{\mathcal{S}} g_I(\mathbf{x}) \frac{\mathbf{x}\cdot\mathbf{n}(\mathbf{x})}{\mathbf{x}_z^3} \nu_{\mathcal{S}}(\mathbf{x}) \mathrm{d}s \\ &\qquad + \int_{\mathcal{I}-\pi(\mathcal{S})} g_B(\mathbf{p}) \mathrm{d}\mathbf{p} \, , \\ &= -\int_{\mathcal{S}} [g_I(\mathbf{x}) - g_B(\pi(\mathbf{x}))] \frac{\mathbf{x}\cdot\mathbf{n}(\mathbf{x})}{\mathbf{x}_z^3} \nu_{\mathcal{S}}(\mathbf{x}) \mathrm{d}s \\ &\qquad + \int_{\mathcal{I}} g_B(\mathbf{p}) \mathrm{d}\mathbf{p} \, , \end{aligned} \quad (49)$$

where $g_I(\mathbf{x})$ is $\frac{1}{2}\left[I(\pi(\mathbf{x}))-C(\mathbf{x})\right]^2$ and $g_B(\mathbf{p})$ is $\frac{1}{2}\left[I(\mathbf{p})-B(\mathbf{p}))\right]^2$. The right-hand term of equation (49) does not depend on $\mathcal{S}$, so in the following we intentionally omitted it as it does not contribute to the gradient expression (with respect to $\mathcal{S}$). Hence, denoting $g(\mathbf{x}) = g_I(\mathbf{x}) - g_B(\pi(\mathbf{x}))$ for convenience, the energy to be minimized with respect to $\mathcal{S}$ is

$$E_{data}(\mathcal{S}) = -\int_{\mathcal{S}} g(\mathbf{x})\frac{\mathbf{x}\cdot\mathbf{n}(\mathbf{x})}{\mathbf{x}_z^3}\nu_{\mathcal{S}}(\mathbf{x})\mathrm{d}s. \tag{50}$$

Now, the energy functional $E_{data}(\mathcal{S})$ is of the form of Equation (5) with

$$\mathbf{g}(\mathbf{x}) = -\frac{g(\mathbf{x})}{\mathbf{x}_z^3}\mathbf{x}.$$

The gradient descent flow for the shape is then directly given by (37)

$$\begin{cases} \mathbf{x}_k(0) = \mathbf{x}_k^0\,, \\ \dfrac{d\mathbf{x}_k}{dt} = -\dfrac{1}{A_k}\left\{\,\mathbf{G}_k^{int} + \mathbf{G}_k^{horiz}\,\right\}\,, \end{cases} \tag{51}$$

where $\mathbf{G}_k^{int}$ and $\mathbf{G}_k^{horiz}$ are respectively obtained from Equations (30) and (36) where $\mathbf{g}(\mathbf{x})$ is replaced by $-\frac{g(\mathbf{x})}{\mathbf{x}_z^3}\mathbf{x}$.

5.3.3 Experiments for the multi-view stereovision application

We have implemented our algorithm using the Delaunay topology-adaptive meshes proposed by [33]. The visibility is computed using OpenGL Z-buffer. The evolution is done using a multi-resolution scheme and starting from the visual hull. Horizons are located using the changes of signs of the dot products of facet normals and viewpoint directions. The *terminator* error metric is computed using OpenGL Shader Language.

As in [15], we first reconstruct three uniformly colored balls arranged on a plane (20 images of resolution 640×480), see Figure 7. This way we ensure the importance of the *horizon* term as the color gradient is null over the surface except at the interfaces between objects in images (which correspond to object boundaries). Using only the *interior* term (Section 4.2.2), the surface shrinks due to the minimal bias. By using the horizon term only (given in Section 4.2.3), we correctly reconstruct and separate the balls, and occluding contours correctly reproject in the images. Then we tested our algorithm on synthetic Lambertian data for the Stanford dragon mesh (Figure 8) composed of 32 images of resolution 640×480. The result shows the correct reconstruction of the dragon, even though the texture is smooth and some parts in shadow are dark (See Table 1). Here the initial shape was a visual hull automatically computed from a stereoscopic segmentation algorithm [47].

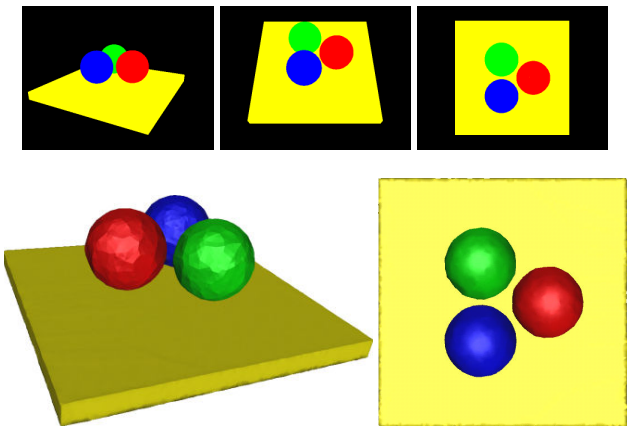

Fig. 7 The balls sequence. Top row: 4 of 20 input images. Bottom row: results with the horizon term computed in Section 4.2.3 from different viewpoints.

	Dragon Diffuse images (Figure 8)
Acc. 95%	0.241mm
Comp. 0.5mm	98.3%

Table 1 Numerical evaluation of the proposed method for the dragon sequence that shows accuracy at 95% and completeness at 0.5mm following [36]. (compared to ground truth)

	templeSparseRing		dinoSparseRing	
	accu. (mm)	compl. (%)	accu. (mm)	compl. (%)
Gargallo [15]	1.05	81.9	0.76	90.7
Gargallo [14]	0.79	96.8	0.50	97.7
Our approach	0.73	95.9	0.89	93.9

Table 2 Results for the temple and dino datasets. For each dataset, accuracy and completeness scores are given. Results of Gargallo et al. are shown for comparison purposes.

Finally, we tested our method on the classical Dino (16 images of 640×480) and Temple (16 images of 640×480 images) datasets from the Middlebury repository (Figure 9). The results and a comparison with selected approaches that motivated our work is presented Table 2. The results of [14, 15] are done in the continuous domain using level set implementation. We refer to the Middlebury benchmark website [36] for evaluation with state-of-the-art reconstruction. We can see that our method is comparable to state-of-the-art, but the main contribution here results in giving a unified framework for photo-consistency optimization that correctly handle visibility using triangular meshes. Additional terms like ballooning forces or silhouettes terms can now be understood, by only dealing with the reprojection error criteria.

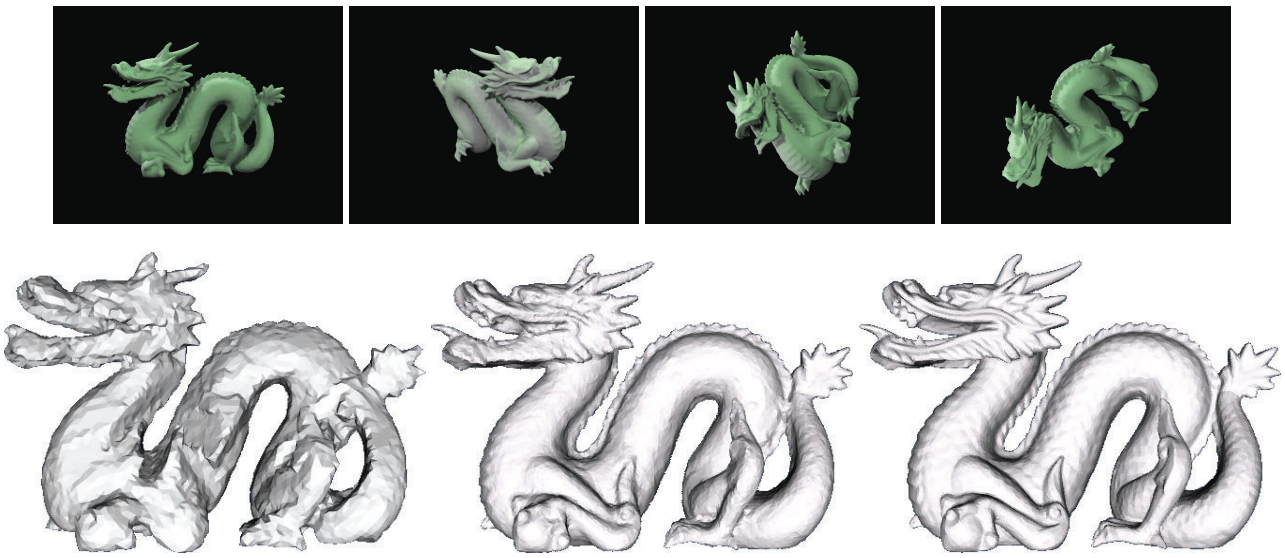

Fig. 8 Synthesized dragon sequence. Top row: 4 of 32 diffuse input images. Bottom row: Initial shape; recovered shape by our algorithm; ground truth model.

5.4 Lambertian 3D Reconstruction using Illumination

In the previous case, the illumination was not taken into account. In fact the estimated color of the surface compared to the input images was the estimated radiance of the object (which was supposed to be Lambertian). However, radiance contains shading, inter-reflections, cast shadows and other non-Lambertian phenomena. The illumination adds additional cues that can be used to estimate surface normals. Considering this, shading can be used in order to recover the geometry of textureless regions. This also allows to separate the surface albedo from the radiance and allows to do more realistic object relighting.

In this section, we only consider the case of Lambertian surfaces. In particular, this allows to consider the multi-view Shape From Shading problem (SFS) and the multi-view Photometric Stereo problem (PS). These problems consist of recovering the 3D shape of a scene by exploiting the information contained in the shading of the corresponding images. Basically in multi-view photometric stereo, the images are generated with varying lighting (typically, each point of the surface must be seen with three different lights). Whereas in multi-view shape from shading, the lighting is the same for all the images. The SFS problem is therefore less well-posed than the PS problem. It then needs some additional constraints. In SFS, we classically assume that the reflectance properties are homogeneous over the whole scene.

A solution to such problems would be a surface $\mathcal{S}$ such that the images generated from that surface are very similar to the observed images (i.e. the data). This naturally leads to formulate the problem as the minimization of an error measure between the observed and predicted values of pixels.

For simplicity, here we are going to consider only Lambertian scenes illuminated by point light sources. This work can nevertheless be extended to other parametric reflectance models and to more realistic lighting conditions for example as done in [21,49]. For a point $\mathbf{x}$ of the surface $\mathcal{S}$, the radiance equation for the i^{th} image is then

$$\begin{aligned} I_i(\pi_i(\mathbf{x})) &= \rho(\mathbf{x}) \left(\sum_{l=1}^{n_L^i} L_l^i \nu_{l,\mathcal{S}}^i(\mathbf{x})(\mathbf{n}(\mathbf{x}) \cdot \mathbf{l}_l^i(\mathbf{x})) + E_0 \right) \\ &= R(\mathbf{x}, \mathbf{n}(\mathbf{x}), \mathcal{S}) \ . \end{aligned} \tag{52}$$

Above L_l^i and $\mathbf{l}_l^i$ are respectively the light intensity color and the light direction of the l^{th} light in the i^{th} image. $\nu_{l,\mathcal{S}}^i(\mathbf{x})$ is the visibility of the l^{th} light source of the i^{th} image at point $\mathbf{x}$ according to $\mathcal{S}$. The additional term E_0 corresponds to the ambient lighting.
A natural energy functional to be considered can be written as :

$$E(\mathcal{S}) = \frac{1}{2} \sum_i \int_{\mathcal{I}_i} \Big(I_i(\mathbf{p}) - R(\pi_{i,\mathcal{S}}^{-1}(\mathbf{p}), \mathbf{n}(\pi_{i,\mathcal{S}}^{-1}(\mathbf{p})), \mathcal{S}) \Big)^2 \mathrm{d}\mathbf{p}, \tag{53}$$

see Jin et al. [21] for details.

To minimize this energy, we alternately minimize it with respects to the shape $\mathcal{S}$ and to the albedo ρ. For a fixed shape

Fig. 9 Dino and Temple sequence (input data courtesy of [36]). From left to right: 1 of 16 input images; estimated background images (scaled by 2 for visualization); estimated radiance; estimated mesh seen from a different viewpoint.

$\mathcal{S}$, the optimal albedo $\rho(\mathbf{x})$ is obtained using :

$$\rho(\mathbf{x}) = \frac{\sum_i I_i\left(\pi_i(\mathbf{x})\right) \nu_{\mathcal{S},i}(\mathbf{x})}{\sum_i \left(\sum_{l=1}^{n_L^i} L_l^i \nu_{l,\mathcal{S}}^i(\mathbf{x})(\mathbf{n}(\mathbf{x}) \cdot \mathbf{l}_l^i(\mathbf{x})) + E_0\right) \nu_{\mathcal{S},i}(\mathbf{x})} . \tag{54}$$

When we assume that the albedo is homogeneous, the denominators and numerators of the above equation have to be integrated on the whole surface $\mathcal{S}$. It has to be then multiplied by the adequate factor corresponding to the camera model; $\mathbf{n}(\mathbf{x}) \cdot \mathbf{x}/\mathbf{x}_z^3$ for a pinhole camera.

Now, let us fix the albedo and optimize energy (53) with respects to the shape. To simplify, we are going to neglect the variations of the visibility of the light sources $\nu_{l,\mathcal{S}}(\mathbf{x})$ when the shape is deforming. These variations are null almost everywhere. They are Dirac functions with a support restricted to the shadow boundaries. In other words, we are neglecting shadow information. Practically in the following, this is equivalent to assuming that $R(\mathbf{x}, \mathbf{n}, \mathcal{S})$ does not depend on $\mathcal{S}$.

In other respects, as previously done in other applications, let us note that the gradient of the energy (53) is the sum of the gradients associated with each one of the images. In the sequel we only compute the gradient associated with one image. By assuming that the camera is a pinhole, we can rewrite this energy as:

$$E(\mathcal{S}) = \int_{\mathcal{S}} \mathbf{g}(\mathbf{x}, \mathbf{n}(\mathbf{x})) \cdot \mathbf{n}(\mathbf{x})\, \nu_{\mathcal{S}}(\mathbf{x})\, \mathrm{d}s\,, \tag{55}$$

where

$$\mathbf{g}(\mathbf{x}, \mathbf{n}) = -\frac{1}{2}\left(I(\pi(\mathbf{x})) - R(\mathbf{x}, \mathbf{n})\right)^2 \frac{\mathbf{x}}{\mathbf{x}_z^3} \tag{56}$$

and where

$$R(\mathbf{x}, \mathbf{n}) = \rho(\mathbf{x}) \left(\sum_{l=1}^{n_L} L_l \nu_l(\mathbf{x})(\mathbf{n} \cdot \mathbf{l}_l(\mathbf{x})) + E_0\right) . \tag{57}$$

We can then directly use the results of Section 4. The gradient is split into three parts: one for the term depending on normals, one for the interior term, and a last one for the horizon term.
For the term depending on normals, we have :

$$\mathbf{G}_k^{norm} = -\sum_{j\in\mathcal{J}_k} \mathbf{e}_{j,k}\wedge \int_T P_{n_j\top}(D_{\mathbf{n}}\mathbf{g}(\mathbf{x}(\mathbf{u}),\mathbf{n}_j)^T\mathbf{n}_j)\,\nu_{\mathcal{S}}(\mathbf{x}(\mathbf{u}))\mathrm{d}\mathbf{u} \quad (58)$$

where $P_{n_j\top}(D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j)$ is the projection of $D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j$ on the tangent plane of the surface. It can be re-written as $D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j - \left(\mathbf{n}_j\cdot D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j\right)\mathbf{n}_j$. Here we have

$$D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j) = \rho(\mathbf{x})\left(I(\pi(\mathbf{x})) - R(\mathbf{x},\mathbf{n}_j)\right) \times \frac{\mathbf{x}}{\mathbf{x}_z^3}\left(\sum_{l=1}^{n_L} L_l\nu_l(\mathbf{x})\mathbf{l}_l(\mathbf{x})^T\right); \quad (59)$$

$\frac{\mathbf{x}}{\mathbf{x}_z^3}\left(\sum_{l=1}^{n_L} L_l\nu_l(\mathbf{x})\mathbf{l}_l(\mathbf{x})^T\right)$ being a 3 matrix, so

$$D_{\mathbf{n}}\mathbf{g}(\mathbf{x},\mathbf{n}_j)^T\mathbf{n}_j = \rho(\mathbf{x})\left(I(\pi(\mathbf{x})) - R(\mathbf{x},\mathbf{n}_j)\right) \times \left(\frac{\mathbf{x}}{\mathbf{x}_z^3}\cdot\mathbf{n}_j\right)\left(\sum_{l=1}^{n_L} L_l\nu_l(\mathbf{x})\mathbf{l}_l(\mathbf{x})\right). \quad (60)$$

The k^{th} component of the *interior* term is:

$$\mathbf{G}_k^{int} = \sum_j A_j\mathbf{n}_j\int_T \nabla\cdot\mathbf{g}(\mathbf{x}(\mathbf{u}),\mathbf{n}_j)\,\phi_k(\mathbf{u})\,\mathrm{d}\mathbf{u}\,, \quad (61)$$

where the sum is on the set of the (completely) *visible* triangles $\mathcal{S}_j$ containing the vertex $\mathbf{x}_k$, and where

$$\nabla\cdot\mathbf{g}(\mathbf{x}(\mathbf{u}),\mathbf{n}_j) = -\left(I(\pi(\mathbf{x})) - R(\mathbf{x},\mathbf{n})\right) \times \left(D\pi(\mathbf{x})^T\nabla I(\mathbf{x}) - \nabla_{\mathbf{x}}R(\mathbf{x},\mathbf{n}_j)\right)\cdot\frac{\mathbf{x}}{\mathbf{x}_z^3}\,. \quad (62)$$

Above, all the terms are explicit at the exception of $\nabla_{\mathbf{x}}R(\mathbf{x},\mathbf{n}_j)$. In fact $\nabla_{\mathbf{x}}R(\mathbf{x},\mathbf{n}_j) = \nabla\rho(\mathbf{x})\mathrm{L}(\mathbf{x},\mathbf{n}_j) + \rho(\mathbf{x})\nabla_{\mathbf{x}}\mathrm{L}(\mathbf{x},\mathbf{n}_j)$ where we denote $\mathrm{L}(\mathbf{x},\mathbf{n}_j) = \sum_{l=1}^{n_L} L_l\nu_l(\mathbf{x})(\mathbf{n}_j\cdot\mathbf{l}_l(\mathbf{x})) + E_0$. The computation of the term $\nabla_{\mathbf{x}}R(\mathbf{x},\mathbf{n}_j)$ and $\nabla_{\mathbf{x}}\mathrm{L}(\mathbf{x},\mathbf{n}_j)$ are detailed and discussed in Appendix A.3.
The k^{th} component of the *Horizon* term is:

$$\mathbf{G}_k^{horiz} = -\sum_{H_{k,j}}\frac{1}{2}\int_0^1 \mathcal{L}(u)\left(\frac{\mathbf{y}(u)}{|\mathbf{y}(u)|}\wedge H_{k,j}\right)(1-u)\mathrm{d}u\,, \quad (63)$$

where $\mathcal{L}(u) = -\left[h(T(\mathbf{y}(u))) - h(\mathbf{y}(u))\right]\frac{1}{\mathbf{y}(u)_z^3}$ with $h = \frac{1}{2}\left(I(\pi(\mathbf{x})) - R(\mathbf{x},\mathbf{n}(\mathbf{x}))\right)^2$ and $T(\mathbf{y}(u))$ is the *terminator* point of $\mathbf{y}(u)$.

5.5 Multi-view Normal Integration

In this section, we present an application for integrating surfaces from multiple normal maps like for instance the one of Chang et al.[5] developed in the level sets framework. Such normals can for instance be obtained via photometric stereo that uses a single fixed camera and a moving light source [20,43]. Having different illumination conditions for one particular view-point allows to estimate the surface normals. By integrating this normal field, it is possible to recover the 3D geometry of the scene. This can be done using the previously described method for normal field integration. However, since photometric stereo is a vision-based application that allows to recover normals for each pixel in the image, the energy functional is based on camera modeling and therefore the energy can be expressed as a reprojection error functional. The gradient descent corresponding to this problem then directly follows the approach presented in Section 4.

The problem can be solved by minimizing the following energy functional:

$$E(\mathcal{S}) = \sum_i\int_{\mathcal{I}}\frac{1}{2}\left(N_i(\mathbf{p}) - \mathbf{n}(\pi_{\mathcal{S}}^{-1}(\mathbf{p}))\right)^2\mathrm{d}\mathbf{p}\,, \quad (64)$$

where $N(\mathbf{p})$ is the normal in input image and $\mathbf{n}(\mathbf{x})$ is the normal of the surface $\mathcal{S}$ at point $\mathbf{x}$. As the norms of N and $\mathbf{n}$ are equal to 1, for simplicity one can rewrite Equation (64) for a *single* image as

$$E(\mathcal{S}) = \int_{\mathcal{I}}\left(1 - N(\mathbf{p})\cdot\mathbf{n}(\pi_{\mathcal{S}}^{-1})\right)\mathrm{d}\mathbf{p}\,. \quad (65)$$

Rewriting it as an integral over the visible surface, we have:

$$E(\mathcal{S}) = \int_{\mathcal{S}}\left(1 - N(\pi(\mathbf{x}))\cdot\mathbf{n}(\mathbf{x})\right)\frac{\mathbf{x}\cdot\mathbf{n}(\mathbf{x})}{\mathbf{x}_z^3}\nu_{\mathcal{S}}(\mathbf{x})\mathrm{d}s\,, \quad (66)$$

which has a similar form as the one used previously for the reprojection error with $g(\mathbf{x},\mathbf{n}) = (1 - N\cdot\mathbf{n})$. This way one can use previous results when the energy functional also depends on the normal. The differential of the energy with respect to a vertex $\mathbf{x}_k$ for the term due to the normal is

$$\mathbf{V}_k\cdot\sum_j\mathbf{e}_{j,k}\wedge\int_{T_j}\left\{\left(N - (N\cdot\mathbf{n}_j)\,\mathbf{n}_j\right)\frac{\mathbf{x}\cdot\mathbf{n}_j}{\mathbf{x}_z^3}\,\nu_{\mathcal{S}}(\mathbf{x})\,\phi_k(\mathbf{x}(\mathbf{u}))\right\}\mathrm{d}\mathbf{u}\,. \quad (67)$$

Then, to get the complete gradient, one has to sum (the gradient corresponding to Equation (67)) with the term due to the differential of a quantity integrated over a visible volume (containing the *interior* term and the *horizon* term). The

interior term is null because on the triangle $\nabla \cdot \mathbf{n}_j = 0$ and because $(\nabla N) \cdot \mathbf{x} = 0$. The Horizon term is:

$$\mathbf{V}_k \cdot \sum_{H_{k,j}} \frac{1}{2} \int_0^1 \left\{ \Big(\mathbf{n}_j - \mathbf{n}(T(\mathbf{y}(u))) \Big) \cdot N(u) \right. \\ \left. \frac{\mathbf{y}(u) \wedge H_{k,j}}{|\mathbf{y}(u)| \, [\mathbf{y}(u)]_z^3} (1-u) \right\} \mathrm{d}u \,, \tag{68}$$

where $T(\mathbf{y}(u))$ is the *terminator* of the current point $\mathbf{y}(u)$ (located behind $\mathbf{y}(u)$ in the view point direction). Note that compared to the gradient in the continuous case, we have here a lower complexity (we are missing the divergence operator and have instead the vectorial product with the opposite edge of the triangle). It is more natural to implement on triangular meshes than the previous continuous case [14] :

$$\nabla \Big((N - (N \cdot \mathbf{n})\mathbf{n}) \frac{\mathbf{x} \cdot \mathbf{n}}{\mathbf{x}_z^3} + (N - \mathbf{n})^2 \frac{\mathbf{x}}{\mathbf{x}_z^3} \Big) \nu_{\mathcal{S}} \\ + \big((N - \mathbf{n})^2 - (N - \mathbf{n}')^2 \big) \frac{\mathbf{x}^t \nabla \mathbf{n} \mathbf{x}}{\mathbf{x}_z^3} \delta(\mathbf{x} \cdot \mathbf{n}) \nu_{\mathcal{S}} \,. \tag{69}$$

As described previously, this can be extended to multi-view photometric stereo methods, where normals are estimated using reflectance and lighting conditions. Then it is possible to integrate this normal field estimated for each image pixel in order to recover the full 3D shape [5,20,43]. In the following, we illustrate the approach with different example, where we also add a smoothness term . The corresponding energy is:

$$E_{R\mathcal{S}} = \int_{\mathcal{S}} (1 - \mathbf{h}(\mathbf{x}) \cdot \mathbf{n}) \, \mathrm{d}s \,,$$

where $\mathbf{h}(\mathbf{x})$ corresponds to the mean of all the normals viewed from each camera at point $\mathbf{x}$. The corresponding gradient is a straight forward application of Section 5.2.

Figure 10, and 12 illustrates our method on synthetic examples for multi-view normal field integration. First, we tested the multiview normal integration algorithm on the simple Ellipse dataset. By using only the term depending on the normal, the surface shrinks. The horizon term allows to constraint the surface such that it matches the image contours. It naturally gives boundary conditions for the normal integration and allows to start from surfaces that does not fully contain the object of interest.

The second experiment shown in Figure illustrates the approach on a CAD designed mesh. The original mesh as twice more vertices and triangles than the reconstructed one. Moreover, since we use a coherent gradient descent flow with respect to the mesh representation, we do not assume normal velocity like in [3,10], making vertices move to appropriate locations. This makes the recovered triangles nicely matches image edges even though the mesh resolution is not very high. Our method can then be used to reconstruct surfaces with sharp edges which is, as far as we know, not possible using implicit surface representations.

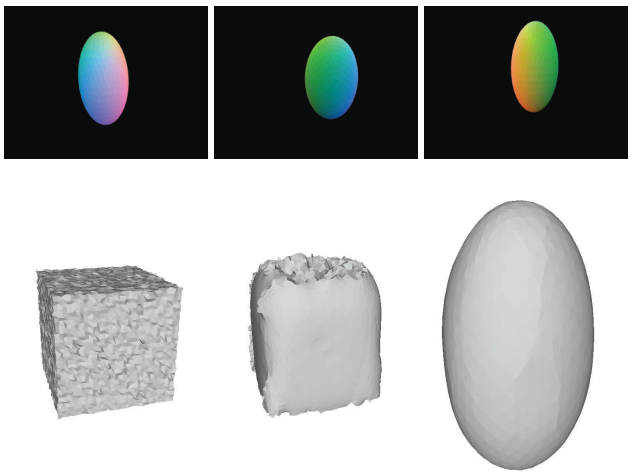

Fig. 10 The ellipse sequence. Top row: 3 of 24 input images showing normal maps of the object of interest; Bottom row: Initial surface; Intermediate result during the evolution; Reconstructed surface.

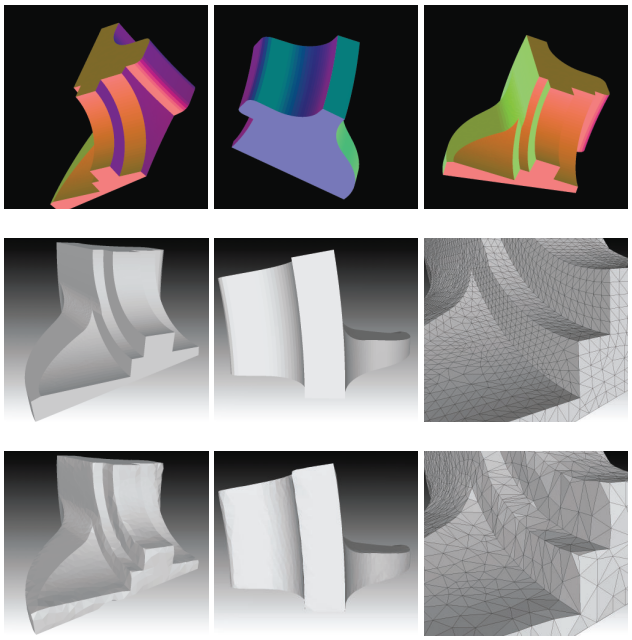

Fig. 11 The Fandisk sequence. From Top to Bottom rows: 3 of 24 input images showing normal maps of the object of interest; original CAD model; reconstructed mesh. The last column shows details of the meshes with the associated triangulation. It shows that the coherent gradient flow makes triangle edges match with the data.

The third experiment (Figure 12) shows the efficiency of the proposed method for handling complex surfaces. The initial surface is the visual hull and a coarse to fine approach is used for the evolution. Details of the result are well recovered, and the final shape is very similar to the ground truth even though the input images have low resolution (640 $\times$

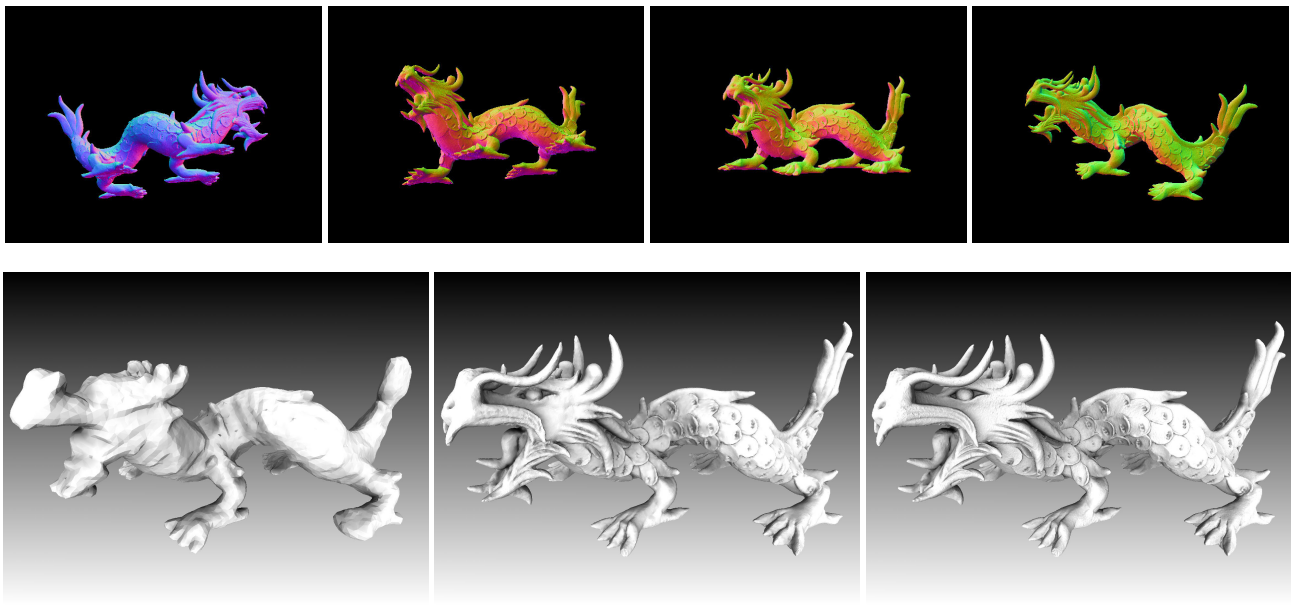

Fig. 12 The dragon sequence (mesh obtained from Stanford repository [1]). Top row: 4 of 24 input images (640×480) showing normal maps of the object of interest; Bottom row: Initial surface; Reconstructed surface; Ground truth shape. Rendering is performed using ambient vertex occlusion and flat shading so that it displays the surface without missing any details.

480). Using photometric information, our method can then be used in order to obtain high quality meshes.

In order to compare ours results to state-of-the-art methods, we tested our approach on the dynamic photometric stereo dataset provided by Vlasic et al. [43]. It is composed of 8 images associated with 8 normal maps. The presented approach can directly be applied to their dataset by performing normal integration. Figure 13 illustrates those results. One of the images shows the shape obtained without using the *horizon* term. In this case the surface shrinks toward the empty set. The result on the right is obtained using the same flow plus the one of the *horizon* term, which yields the expected result. Integrating the normal field gives good high frequency and details, but is poor for low frequency due to the integration. In these conditions, mixing multiview stereo and multiview normal field integration will provide powerful 3D reconstruction algorithms [31,43]. Figure 14 shows the results obtained by [43] along with ours. In their paper, the authors compute several normal maps from each view, and then register and merge the different integrations in order to obtain the final mesh. In this context they have troubles in recovering parts where the normal maps contain occluding contours. Since our approach is surface-based, we can better exploit the multi-view system in the reconstruction process and the 3D position of the surface is more accurate even though both methods nicely recover shape details (note that there are *only* 8 images). In fact, Figure 14 also shows the textured meshes obtained by reprojecting camera images onto the mesh. This emphasize the fact that even though the recovered surface of [43] visually looks really nice, they suffer from the integration bias and registration errors which leads to slightly incorrect 3D positions, as well as incomplete surface recovery. For example, the two images seeing the right ear reproject in different locations, creating a non-coherent textured surface - other problems are shown in red. This might result in wrong visual artifacts for relighting purposes. In contrast, our approach naturally takes advantage of the multi-view information. Also, our approach is purely image-based and does not use pre or post-processing such as re-estimating (and smoothing) the normal maps or performing hole filling like in [43]. Some part like the cap are not well recovered in our case, mainly due to noise and missing normals in the input images, but also by the fact we use a closed surface.

5.6 Discussion

The presented approach offers a general framework to solve different vision reconstruction problems using deformable meshes. However the way it is optimized might be improved since it uses simple L^2 gradient descent. One may change the gradient metric, make the functional convex, change the optimization algorithm or simply define a more robust cost measure to improve robustness or speed. In particular the initialization is important here, and there might be cases were the minimization fails because of local minima, even though coarse-to-fine approach (in both images and mesh

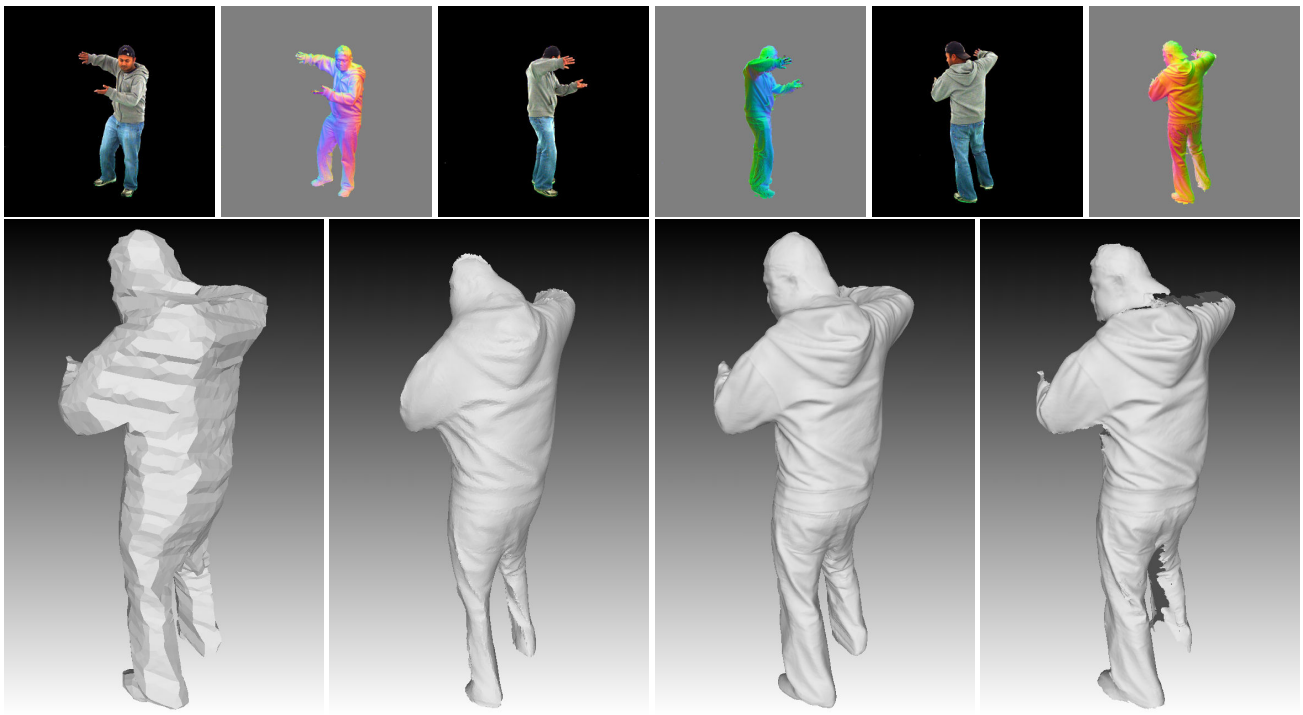

Fig. 13 MIT Sequence (Courtesy of [43]). Top row: 3 of 8 input images and normals (1024×1024). Bottom row, from left to right: visual hull; result using Section 5.5 *wihtout* using the horizon term; result using Section 5.5 *with* the horizon term; result of [43].

resolution) significantly help the algorithm. A study of possible improvements is out the scope of this work but this paper still provides tools and inspiration for that (for example changing the gradient metric for deformable meshes is a straight forward extension of [12], and [45] already proposed a global error metric based on cross-correlations for triangle meshes based on [8,34]). This work focuses more on the modeling part, by explaining the shape directly from images, rather than reconstructing surfaces under constraints (i.e. processing time, fast convergence, etc...). The paper however shows that it can be successfully applied for mesh refinement in a variety of cases, including geometric flows that depends on the surface's normal and/or visibility. Moreover one may adapt its error metric $\mathbf{g}$ for concerned applications - for example adding volumetric flows such as silhouettes constraints or edge attachment weight as done in geodesic active contour methods. In all cases adapting the metric is straight forward and its minimization via L^2 gradient descent is a direct application of the presented approach. Finally, in this work we used deformable meshes which imply remeshing (in particular for topology changes) during the optimization. In this context, adding or removing points might slightly change the objective functional. One additional improvement would be to change the remeshing algorithm in order to completely ensure the spatial consistency in the optimization process, but this is out of the scope of this paper and is still an open research area.

6 Conclusion

In this paper we compute the shape gradient of general energy functionals which account for normals and visibility changes and which embody a number of energies used in mesh processing and computer vision. Gradient computation is done directly with respect to the discrete representation of the surface based on triangular meshes. This allows for coherent gradient flows that tend to place the mesh vertices to their correct locations and make triangle edges match with the data. To illustrate the presented approach and show the advantage of having a coherent gradient flow, we apply our results to several applications.

In particular we presented a mesh evolution technique for 3D vision problems, when the cost functional depends on an image based score and a camera model. For instance, we show the multiview stereo problem based on the *discrete* representation. Contrary to previous works, during the evolution, *we correctly deal with visibility changes* by expressing the exact gradient of the reprojection error functional. In particular, exactly as in the continuous case [15], this forces the contour generators of the surface to appear at their correct location in the images and reduces the minimal surface bias from which some variational methods suffer.

Acknowledgements The authors want to thank J.-P. Pons for sharing his implementation of the topologically-adaptive deformable meshes and for the useful discussions and help; S.M. Seitz, B. Curless, J. Diebel,

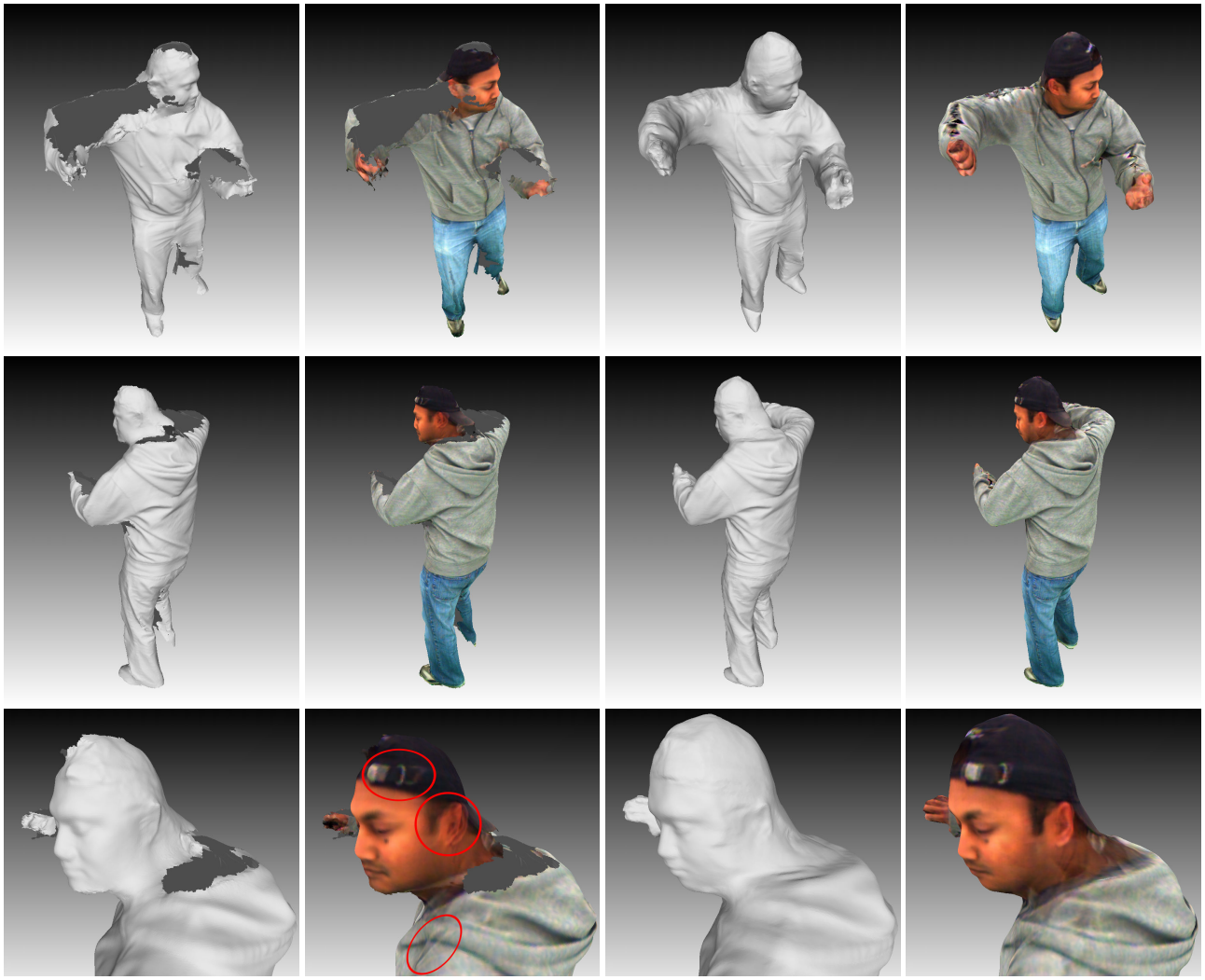

Fig. 14 Results and comparison with [43]. From left to right: final shape of [43]; corresponding mean texture from visible cameras; result with the proposed approach; corresponding mean texture from visible cameras.

D. Scharstein and R. Szeliski for the temple and dino datasets and evaluations; D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz and W. Matusik for the dynamic multi-view photometric stereo data; the Stanford repository team for their data; P. Gargallo for the usefull discussions and help and the ball dataset; and the anonymous reviewers for their work that helped to improve the quality of this paper. This work was supported by the Agence Nationale pour la Recherche within the Flamenco project (Grant ANR-06-MDCA-007).

References

1. The Stanford 3D Scanning Repository. http://graphics.stanford.edu/data/3dscanrep/.
2. M. Bertalmio, G. Sapiro, Li-Tien Cheng, and S. Osher. Variational problems and pdes on implicit surfaces. *Variational and Level Set Methods in Computer Vision*, 2001.
3. N. Birkbeck, D. Cobzas, P. Sturm, and M. Jägersand. Variational shape and reflectance estimation under changing light and viewpoints. In *Proceedings of European Conference on Computer Vision*, volume 1, pages 536–549, 2006.
4. CGAL, Computational Geometry Algorithms Library. http://www.cgal.org/.
5. J. Y. Chang, K. M. Lee, and S. U. Lee. Multiview normal field integration using level set methods. In *IEEE Conference in Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007.
6. G. Charpiat, P. Maurel, J.-P. Pons, R. Keriven, and O. Faugeras. Generalized gradients: Priors on minimization flows. *International Journal of Computer Vision*, 73(3):325–344, Jul 2007.
7. É. Debreuve, M. Gastaud, M. Barlaud, and G. Aubert. Using the shape gradient for active contour segmentation: from the continuous to the discrete formulation. *Journal of Mathematical Imaging and Vision*, 2007.
8. A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons, and P. Sturm. Minimizing the multi-view stereo reprojection error for triangular surface meshes. In *British Machine and Vision Conference, Leeds, UK*, 2008.
9. M. Desbrun, M. Meyerm, P. Schröder, and A. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99*, pages 317–324, 1999.
10. Y. Duan, L. Yang, H. Qin, and D. Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In *Eu-*

ropean Conference on Computer Vision, pages 238–251, Prague, Czech Republic, 2004.
11. G. Dziuk and C.M. Elliott. Finite elements on evolving surfaces. *IMA journal of numerical analysis*, 27(2):262–292, 2007.
12. I. Eckstein, J.-P. Pons, Y. Tong, C.-C. J. Kuo, and M. Desbrun. Generalized surface flows for mesh processing. In *Eurographics Symposium on Geometry Processing*, 2007.
13. O.D. Faugeras and R. Keriven. Variational-principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE TIP*, 7(3):336–344, 1998.
14. P. Gargallo. *Contributions to the Bayesian approach to Multi-view Stereo*. PhD thesis, Institut National Polytechique de Grenoble, France, February 2008.
15. P. Gargallo, E. Prados, and P. Sturm. Minimizing the reprojection error in surface reconstruction from images. In *Proceedings of the International Conference on Computer Vision, Rio de Janeiro, Brazil*. IEEE Computer Society Press, 2007.
16. B. Goldlucke, I. Ihrke, C. Linz, and M. Magnor. Weighted minimal hypersurface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1194–1208, 2007.
17. B. Goldlücke and M. A. Magnor. Weighted minimal hypersurfaces and their applications in computer vision. In T. Pajdla and J. Matas, editors, *European Conference on Computer Vision*, volume 3022 of *Lecture Notes in Computer Science*, pages 366–378. Springer, 2004.
18. R. Gupta and R. I. Hartley. Linear pushbroom cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):963–975, 1997.
19. C. Hernandez and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, 2004.
20. C. Hernandez, G. Vogiatzis, and R. Cipolla. Multiview photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):548–554, 2008.
21. H. Jin, D. Cremers, D. Wang, E. Prados, A. Yezzi, and S. Soatto. 3-d reconstruction of shaded objects from multiple images under unknown illumination. *International Journal of Computer Vision*, 76(3), March 2008.
22. H. Jin, S. Soatto, and A. J. Yezzi. Multi-view stereo reconstruction of dense shape and complex appearance. *International Journal of Computer Vision*, 63(3):175–189, 2005.
23. H. Jin, A.J. Yezzi, and S. Soatto. Variational multiframe stereo in the presence of specular reflections. In *3DPVT*, pages 626–630, 2002.
24. K. Kolev, D. Cremers Integration of multiview stereo and silhouettes via convex functionals on convex domains In *European Conference on Computer Vision*, oct 2008.
25. K. Kolev, D. Cremers Continuous ratio optimization via convex relaxation with applications 3D multiview reconstruction In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.
26. K. Kolev, M. Klodt, T. Brox, D. Cremers Continuous global optimization in multiview 3D reconstruction In *International Journal of Computer Vision*, 2009.
27. K. Kolev, T. Pock, D. Cremers Anisotropic Minimal Surfaces Integrating Photoconsistency and Normal Information for Multiview Stereo In *European Conference on Computer Vision*, 2010.
28. Kolmogorov and Zabih. What energy functions can be minimized via graph cuts. *IEEETPAMI: IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26, 2004.
29. P. Labatut, R. Keriven, and J.-P. Pons. Fast level set multi-view stereo on graphics hardware. In *3DPVT '06: Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*, pages 774–781, Washington, DC, USA, 2006. IEEE Computer Society.
30. M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. 2002.
31. D. Nehab, S. Rusinkiewicz, J. Davis and R. Ramamoorthi Efficiently combining positions and normals for precise 3D geometry In *SIGGRAPH*, 536–543, 2005.
32. S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.
33. J.-P. Pons and J.-D. Boissonnat. A lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space. *Computer Graphics Forum*, 26(2):227–239, Aug 2007.
34. J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 822–827, San Diego, USA, Jun 2005.
35. J.-P. Pons, R. Keriven, and O. Faugeras. Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score. *The International Journal of Computer Vision*, 72(2):179–193, Apr 2007.
36. S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 519–528, 2006.
37. S. N. Sinha and M. Pollefeys. Multi-view reconstruction using photo-consistency and exact silhouette constraints: a maximum-flow formulation. In *IEEE International Conference on Computer Vision*, pages 349–356, 2005.
38. G. Slabaugh and G. Unal. Active polyhedron: surface evolution theory applied to deformable meshes. *IEEE Conference on Computer Vision and Pattern Recognition*, 2:84–91 vol. 2, 2005.
39. S. Soatto, A. J. Yezzi, and H. Jin. Tales of shape and radiance in multi-view stereo. In *IEEE International Conference on Computer Vision*, pages 974–981, 2003.
40. J. E. Solem, H. Aanaes, and A. Heyden. A variational analysis of shape from specularities using sparse data. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 26–33, Washington, DC, USA, 2004. IEEE Computer Society.
41. J. E. Solem and A. Heyden. Reconstructing open surfaces from image data. *Int. J. Comput. Vision*, 69(3):267–275, 2006.
42. J. E. Solem and N. Chr. Overgaard. A geometric formulation of gradient descent for variational problems with moving surfaces. In R. Kimmel, N. Sochen, and J. Weickert, editors, *Scale-Space 2005*, volume 3459 of *LNCS*, pages 419–430. Springer Verlag, 2005.
43. D. Vlasic, P. Peers, I. Baran, P. Debevec, J. Popović, S. Rusinkiewicz and W. Matusik Dynamic Shape Capture using Multi-View Photometric Stereo In *ACM Transactions on Graphics*, 28(5):174, 2009.
44. G. Vogiatzis, C. Hernández Esteban, P. H. S. Torr, and R. Cipolla. Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(12):2241–2246, 2007.
45. H. Vu, R. Keriven, P. Labatut, and J.-P Pons. Towards high-resolution large-scale multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2009.
46. R. T. Whitaker. A level-set approach to 3d reconstruction from range data. *International Journal of Computer Vision*, 29(3):203–231, 1998.
47. A. Yezzi and S. Soatto. Stereoscopic segmentation. *International Journal of Computer Vision*, 53(1):31–43, 2003.
48. A.J. Yezzi and S. Soatto. Stereoscopic segmentation. *IEEE International Conference on Computer Vision*, 2001.
49. K-J. Yoon, E. Prados, and P. Sturm. Joint estimation of shape and reflectance using multiple images with known illumination conditions. *International Journal of Computer Vision*, 2009. to appear.
50. T. Yu, N. Xu, and N. Ahuja. Shape and view independent reflectance map from multiple views. *International Journal of Computer Vision*, 73(2):123–138, 2007.

51. A. Zaharescu, E. Boyer, and R.P. Horaud. Transformesh: a topology-adaptive mesh-based approach to surface evolution. In *Proceedings Asian Conference on Computer Vision*, Tokyo, Japan, November 2007.

A Computation details

A.1 Expression of $A'_j[0]$

We have to explicit $A'_j[0]$ linearly in function of the $\mathbf{V}$. The area $A_j[t]$ of a triangle $\mathcal{S}_j[t]$ is $A_j[t] = \frac{1}{2} \parallel \overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k1}[t]} \wedge \overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k2}[t]} \parallel$ where $\mathbf{x}_k[t]$, $\mathbf{x}_{k1}[t]$, $\mathbf{x}_{k2}[t]$ are the vertices of triangle i at time t. For more convinience, we express the squared area A_j^2 to avoid squared root while computing the differential at $t = 0$. Then we have :

$$A_j^2[t] = \frac{1}{4} \left(\overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k1}[t]} \wedge \overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k2}[t]} \right) \cdot \left(\overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k1}[t]} \wedge \overrightarrow{\mathbf{x}_k[t]\mathbf{x}_{k2}[t]} \right)$$

$$\frac{d}{dt} A_j^2[t]\Big|_{t=0} = \frac{1}{2} \left(\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}} \right) \cdot \left(\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k + \overrightarrow{\mathbf{x}_{k2}\mathbf{x}_k} \wedge \mathbf{V}_{k1} + \overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \mathbf{V}_{k2} \right).$$

Using $\frac{d}{dt} A_j^2[t]\Big|_{t=0} = 2\, A_j[0]\, \frac{d}{dt} A_j[t]\Big|_{t=0}$, we get

$$A'_j[0] = \sum_{k \in K_j} \mathbf{V}_k \cdot \left(\frac{1}{2} \mathbf{n}_j \wedge \mathbf{e}_{j,k} \right) . \tag{70}$$

If we move only one vertex at once (meaning V_{k1} & V_{k2} are null for vertex k), we have :

$$A'_j[0] = \frac{1}{2} \left(\mathbf{n}_j \wedge \overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \right) \cdot \mathbf{V}_k . \tag{71}$$

A.2 Expression of $\mathbf{n}'_j[0]$

We have to explicit $\mathbf{n}'_j[0]$ linearly in function of the $\mathbf{V}$. Considering $\mathbf{n}_j = \frac{\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}}}{2\, A_j}$, we have :

$$\mathbf{n}'_j[0] = \frac{1}{2\, A_j[0]^2} \left((\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}})'[0]\, A_j[0] - \left(\overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \overrightarrow{\mathbf{x}_k\mathbf{x}_{k2}} \right) A'_j[0] \right).$$

$$\mathbf{n}'_j[0] = \frac{1}{2\, A_j} \left(\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k + \overrightarrow{\mathbf{x}_{k2}\mathbf{x}_k} \wedge \mathbf{V}_{k1} + \overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \mathbf{V}_{k2} - \left((\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k + \overrightarrow{\mathbf{x}_{k2}\mathbf{x}_k} \wedge \mathbf{V}_{k1} + \overrightarrow{\mathbf{x}_k\mathbf{x}_{k1}} \wedge \mathbf{V}_{k2}) \cdot \mathbf{n}_j \right) \mathbf{n}_j \right).$$

So

$$\mathbf{n}'_j[0] = \frac{1}{2\, A_j} \left(\left(\sum_{k \in K_j} \mathbf{e}_{j,k} \wedge \mathbf{V}_k \right) - \left(\left(\sum_{k \in K_j} \mathbf{e}_{j,k} \wedge \mathbf{V}_k \right) \cdot \mathbf{n}_j \right) \mathbf{n}_j \right). \tag{72}$$

Therefore $\mathbf{n}'_j[0]$ is the projection of $\sum_{k \in K_j} \mathbf{e}_{j,k} \wedge \mathbf{V}_k$ on the orthogonal plane to $\mathbf{n}_j$, divided by $2\, A_j$.

In the case where we consider moving only one vertex at once (meaning $\mathbf{V}_{k1}$ & $\mathbf{V}_{k2}$ are null for vertex k), we have :

$$\mathbf{n}'_j[0] = \frac{\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k - \left((\overrightarrow{\mathbf{x}_{k1}\mathbf{x}_{k2}} \wedge \mathbf{V}_k) \cdot \mathbf{n}_j \right) \mathbf{n}_j}{2\, A_j} . \tag{73}$$

A.3 Details on the Lambertian Case Using Illumination

All the terms in Equation (62) are explicit at the exception of $\nabla_{\mathbf{x}} R(\mathbf{x}, \mathbf{n}_j)$. In fact $\nabla_{\mathbf{x}} R(\mathbf{x}, \mathbf{n}_j) = \nabla\rho(\mathbf{x}) \mathrm{L}(\mathbf{x}, \mathbf{n}_j) + \rho(\mathbf{x}) \nabla_{\mathbf{x}} \mathrm{L}(\mathbf{x}, \mathbf{n}_j)$ where we denote

$$\mathrm{L}(\mathbf{x}, \mathbf{n}_j) = \sum_{l=1}^{n_L} L_l \nu_l(\mathbf{x}) (\mathbf{n}_j \cdot \mathbf{l}_l(\mathbf{x})) + E_0 .$$

In the case of a homogeneous albedo (typically in shape from shading) we have $\nabla\rho(\mathbf{x}) = 0$.

According to (54), for $\mathbf{x}$ in $\mathcal{S}_j$, we have $\nabla\rho(\mathbf{x}) = \dfrac{b\, \nabla a - a\, \nabla b}{b^2}$, where a and b are defined by

$$\begin{aligned} a &= \sum_i I_i(\pi_i(\mathbf{x})) \nu_{\mathcal{S},i}(\mathbf{x}) , \\ b &= \sum_i \left(\sum_{l=1}^{n_L^i} L_l^i \nu_{l^i,\mathcal{S}}(\mathbf{x}) (\mathbf{n}_j \cdot \mathbf{l}_l^i(\mathbf{x})) + E_0 \right) \nu_{\mathcal{S},i}(\mathbf{x}) \end{aligned} \tag{74}$$

and where ∇a and ∇b are

$$\begin{aligned} \nabla a &= \sum_i D\pi(\mathbf{x})^T \nabla I_i(\pi_i(\mathbf{x})) \nu_{\mathcal{S},i}(\mathbf{x}) , \\ \nabla b &= \sum_i \left(\sum_{l=1}^{n_L^i} L_l^i \nu_{l^i,\mathcal{S}}(\mathbf{x}) (\mathbf{n}_j \cdot \nabla \mathbf{l}_l^i(\mathbf{x})) \right) \nu_{\mathcal{S},i}(\mathbf{x}). \end{aligned} \tag{75}$$

(We assume here that visibilities are the same for all the points $\mathbf{x}$ on the triangle $\mathcal{S}_j$, or we neglect their variations). For scenes illuminated by far light sources we have $\nabla \mathbf{l}_l^i(\mathbf{x}) \approx 0$, and so $\nabla\rho \approx a'/b$. Moreover, if the light sources are same for all the image, then

$$b = (\sum_i \nu_{\mathcal{S},i}(\mathbf{x})) \left(\sum_{l=1}^{n_L} L_l (\mathbf{n}_j \cdot \mathbf{l}_l) \nu_{l,\mathcal{S}}(\mathbf{x}) + E_0 \right) .$$

Finally, neglecting the variations of the light visibility, we have

$$\nabla_{\mathbf{x}} \mathrm{L}(\mathbf{x}, \mathbf{n}_j) = \sum_{l=1}^{n_L} L_l \nu_l(\mathbf{x}) D\mathbf{l}_l(\mathbf{x})^T \mathbf{n}_j,$$

where $D\mathbf{l}_l(\mathbf{x})^T$ is the transposition of the differential of $\mathbf{l}_l$ (3×3 matrix). In the case of far light sources, we have $\nabla_{\mathbf{x}} \mathrm{L}(\mathbf{x}, \mathbf{n}_j) \approx 0$.