International Journal of Computer Mathematics
Vol. 90, No. 1, Jan 2013, 92–110

On Fast Iterative Algorithms For Solving The Minimisation Of Curvature-Related Functionals In Surface Fairing

Carlos Brito-Loeza and Ke Chen

A number of successful variational models for processing planar images have recently been generalized to three dimensional (3D) surface processing. With this new dimensionality, the amount of numerical computations to solve the minimization of such new 3D formulations naturally grows up dramatically. Though the need of computationally fast and efficient numerical algorithms able to process high resolution surfaces is high, much less work has been done in this area. Recently a two step algorithm for the fast solution of the *total curvature* model was introduced in [T. Tasdizen, R. Whitaker, P. Burchard and S. Osher, ACM Transactions on Graphics, 22, 2003]. In this paper, we generalise and modify this algorithm to the solution of analogues of the *mean curvature* model of [M. Droske and Martin Rumpf, Interfaces and free boundaries, 6, 2004] and the *Gaussian curvature* model of [Matthew Elsey and Selim Esedoḡlu, Multiscale Model. Simul., 7, 2009]. Numerical experiments are shown to illustrate the good performance of the algorithms and test results.

1 Introduction

In recent years it has become evident that some variational models developed for manipulating and transforming digital images in popular tasks such as image reconstruction, image denoising, image segmentation and others are able to succeed where other computational strategies struggle to deliver good results. Due to this success some researches have worked on the generalization of such models to 3D processing.

One of the hot topics has been the study of models for surface fairing also known as surface denoising or surface smoothing [9,17,18]. In this application, the selection of the regularization term which is the one in charge of capturing the high frequency oscillations it is a critical designing step. It is also highly desirable to provide such models with strong geometric foundations in such a way that all the theory and mathematical tools from the fields of geometry and differential geometry can be at hand to properly analyze such models. Similar to what has been done in image processing, geometric entities such as the gradient or the curvature of the surface may be used. In deciding what to use, one may consider generalizing the regularizers for planar images, e.g., total variation (first order information) and curvature (second order information). It turns out that unless the underlying objects are special, [7, 10, 15] have already presented clear evidence that, for surface fairing, second order information (curvature) is preferable over first order information (gradient) in order to obtain the best results. In the literature we can find among others, the approach of Welch and Witkin [21,22] for mesh fairing which minimizes a functional based on total curvature (TC), the model of Schneider and Kobbelt [15] using the intrinsic Laplacian of mean curvature also on meshes, the work of Droske and Rumpf [9] using a level set formulation for the Willmore or mean curvature (MC) energy, the Tasdizen *et al.* work [17] also using total curvature and the model of Elsey and Esedoglu [10] minimizing an energy function based on the Gaussian curvature (GC) of the surface.

In the above five models, the last three [9, 10, 17] making use of level sets to obtain an implicit surface representation are considered in this work. These

Facultad de Matemáticas, Universidad Autónoma de Yucatán Mexico. Email: carlos.brito@uady.mx
Centre for Mathematical Imaging Techniques and Department of Mathematical Sciences, The University of Liverpool, Peach Street, Liverpool L69 7ZL, United Kingdom. Email: k.chen@liv.ac.uk, cmit@liv.ac.uk

ISSN 1029-4937 print; ISSN 0020-7160 online
DOI:

three models differ from each other in the way the curvature is defined and the way the numerical solution is obtained. About the latter and more critical than 2D models where the fast numerical solution is needed, the numerical algorithm for 3D plays a very important role in the realisation of a fairing model due to the large amount of data to be processed. When moving from 2D to 3D, the number of computations increases in nonlinear proportion to the new dimensionality. Therefore new and fast numerical algorithms are necessary and in high demand.

Each model from [9,10,17] has as first-order optimality condition a fourth order nonlinear partial differential equation (PDE). It is well known that fourth order PDEs do not obey a maximum principle and, in parabolic form, are usually very stiff. Besides, accurate discretisation of the high order derivatives is computationally expensive and probably oscillatory. These are some of the reasons that commonly motivate researchers to avoid direct numerical solution of such difficult equations. Recently for 2D images, there has been some improvement in this field with the development of algorithms such as the stabilized fixed point (SFP) method reported in [2] and based on convexity splitting ideas, the nonlinear multigrid method of [6], the augmented Langragian method of [23] and the fixed point homotopy method of [24]. All those are very promising results but unless convergence is obtained in a short number of iterations, they can still be computationally expensive for commercial use.

Also for 2D images, there is the two-step (TS) algorithm that has already been successfully tested in a variety of high order variational problems; see for instance [1, 17] and references therein. In a TS algorithm, an irregular normal field obtained from the noisy data is first smoothed and then new data fitted to the smoothed field. This method has already been used in [17] for the TC surface fairing model. It is known that the TS algorithm, which we will review in detail shortly, does not always give the right solution to the original problem but a close approximation to it.

In this paper, we will follow similar arguments to those presented in [17] to show that by using a TS in a different way we can indeed find the accurate solution of analogous models to [9, 10, 17]. The rest of this paper is organized as follows. Section 2 reviews the different definitions for surface curvature. Section 3 presents the three curvature based models from [9, 10, 17]. Section 4 will be committed to introduce and analyse the TS algorithm for a general formulation. In Section 5 we will show how to use a TS for the solution of the MC model [9] and the GC model [10]. Section 8 will present some results, the complexity analysis and a performance comparison of the TS algorithm over the analogue and original models. Finally, in Section 9, we will discuss the results and the challenges still to overcome.

2 A review of surface curvature

In geometry the curvature, here represented by $\kappa(\boldsymbol{x})$, at a point $\boldsymbol{x} = (x, y, z)$ living on a 3D surface S is related to the amount of deviation of $S(\boldsymbol{x})$ from being flat. There are two universally accepted curvature definitions for a given surface S and they are:

(1) Gaussian curvature is defined as the product of the two principal curvatures [8], i.e. $\kappa_G(\boldsymbol{x}) = \kappa_1(\boldsymbol{x})\kappa_2(\boldsymbol{x})$. This curvature is an intrinsic measure since its value does not depend on the way S is embedded in the space. The value of κ_G represents the element of area of spherical element of surface area.

(2) Mean curvature which is computed by averaging the two principal curvatures [8], i.e. $\kappa_M(\boldsymbol{x}) = (\kappa_1(\boldsymbol{x}) + \kappa_2(\boldsymbol{x}))/2$, represents the rate of change of surface area under small deformations in the normal direction.

When constructing variational models for surface fairing, either Gaussian curvature [10] or Mean curvature [9] have been used to define energy func-

tionals. These functionals need to be minimised in order to reduce the sharp oscillations constituting the noise or imperfections on the surface. The use of curvature-based functionals to capture the energy of the corrupted data to be processed has proved to deliver remarkable results in other topics of signal and image processing, see for instance [2,5,6] and references therein. Therefore its generalisation to 3D surface processing comes naturally.

We remark that since surfaces are commonly defined parametrically and hence curvature formulas are often represented by local parameters [8]. Although models using this kind of representation suit very well to mesh smoothing on one hand, it is however well known that, with the parametric representation, any severe change in the topology of the surface (such as splitting up) is very difficult to handle. Then again, implicit surface representation by using level set functions, does not suffer from this problem and is a technique widely tested in many applications. Curvature formulas for implicitly defined surfaces, on the other hand, are much harder to locate. In this sense, the monograph of Goldman [12] contains an invaluable source of information.

Here we will use ϕ to represent the level set surface representation and will adopt the following notation for the gradient vector $\nabla\phi$, the Hessian matrix $H(\phi)$ and its adjoint $H^*(\phi)$

$$\nabla\phi = (\phi_x, \phi_y, \phi_z) \tag{1}$$

$$H(\phi) = \begin{pmatrix} \phi_{xx} & \phi_{xy} & \phi_{xz} \\ \phi_{yx} & \phi_{yy} & \phi_{yz} \\ \phi_{zx} & \phi_{zy} & \phi_{zz} \end{pmatrix} \tag{2}$$

$$H^*(\phi) = \begin{pmatrix} \phi_{yy}\phi_{zz} - \phi_{yz}\phi_{zy} & \phi_{yz}\phi_{zx} - \phi_{yx}\phi_{zz} & \phi_{yx}\phi_{zy} - \phi_{yy}\phi_{zx} \\ \phi_{zx}\phi_{zy} - \phi_{xy}\phi_{zz} & \phi_{xx}\phi_{zz} - \phi_{xz}\phi_{zx} & \phi_{xy}\phi_{zx} - \phi_{xx}\phi_{zy} \\ \phi_{xy}\phi_{yz} - \phi_{xz}\phi_{yy} & \phi_{yx}\phi_{xz} - \phi_{xx}\phi_{yz} & \phi_{xx}\phi_{yy} - \phi_{xy}\phi_{yx} \end{pmatrix} \tag{3}$$

Following [12], the Mean and Gaussian curvatures are defined as follows

$$\kappa_M = \nabla \cdot \left(\frac{\nabla\phi}{|\nabla\phi|} \right), \tag{4}$$

$$\kappa_G = \frac{\nabla\phi \cdot H^*(\phi) \cdot \nabla\phi^T}{|\nabla\phi|^4} \tag{5}$$

where $|\nabla\phi| = \sqrt{\phi_x^2 + \phi_y^2 + \phi_z^2}$. Further they are related by the following equality

$$\kappa_G = \frac{1}{2}\left(\kappa_M^2 - \kappa_T^2\right) \tag{6}$$

where $\kappa_T = \sqrt{\kappa_1^2 + \kappa_2^2}$ is the *length* of the second fundamental form [4]. The integral $\int_S \kappa_T^2 dS$ has been called by some researchers the *total curvature* of the surface. The expression in level set form for κ_T is given by

$$\kappa_T = |(\nabla \boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})|, \quad \boldsymbol{P} = \frac{\nabla\phi \otimes \nabla\phi}{|\nabla\phi|^2}, \quad \boldsymbol{N} = \frac{\nabla\phi}{|\nabla\phi|} \tag{7}$$

where $\otimes$ is the tensor product, $\boldsymbol{I}$ the identity matrix,, $\boldsymbol{I} - \boldsymbol{P}$ the 3×3 tangential projection matrix, $\boldsymbol{N}$ is the unit outward normal vector to the surface and $\nabla\boldsymbol{N}$ represents a matrix whose rows are the gradient vectors of the components of $\boldsymbol{N}$.

3 Curvature-based surface fairing models

In this section, we shall review three variational surface fairing models before we consider how to solve them effectively in next sections.

3.1 *The total curvature model – TC [17]*

In [17] the authors filtered out the noise of the surface by constructing an energy functional based on κ_T and proposed the following minimisation problem

$$\min_{\phi} \int_{\Omega} G(\kappa_T^2)|\nabla\phi| d\boldsymbol{x} \tag{8}$$

where $\Omega \subset R^3$ is the domain where the level set function ϕ lives, G may be chosen as the identity function, i.e. $G(\kappa_T^2) = \kappa_T^2$, yielding isotropic diffusion or as

$$G(\kappa_T^2) = 2\mu^2(1 - e^{-\frac{\kappa_T^2}{2\mu^2}}) \tag{9}$$

leading to anisotropic diffusion, where μ is a positive parameter.

To avoid solving directly the associated fourth order Euler-Lagrange PDE and noticing that κ_T^2 can be expressed in terms of $\boldsymbol{N}$, the authors of [17] found a simply but elegant way to solve (8) by decoupling $\boldsymbol{N}$ from ϕ. They applied a two-step process by first minimising (8) with respect $\boldsymbol{N}$ to obtain a smoothed new $\boldsymbol{N}$ and second fitting a new ϕ to the this vector field . This process although computationally very efficient, does not give the exact solution to (8) but only a very good approximation as it was showed in [17]. We will show more details of this two step process in solving the analogue models to be introduced below.

Aiming to simplify even more the problem, in [17] the term $|\nabla\phi|$ was removed from (8) to create an easier to solve *analogue model*. By using the definition of κ_T in (7) this model can be written as

$$\min_{\phi} \int_{\Omega} G(\|(\nabla\boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})\|) d\boldsymbol{x}. \tag{10}$$

This *analogue model* which also has associated a fourth order PDE was solved using the two-step process mentioned above. In details, first the normals to the noisy surface are computed using the available ϕ_{old} and processed until smoothing them. This means using (7) to compute $\boldsymbol{P}(\phi_{old})$ and minimising

$$\min_{\boldsymbol{N}} \int_{\Omega} G(\|(\nabla\boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})\|) d\boldsymbol{x} \tag{11}$$

through solving

$$\frac{\boldsymbol{N}^{k+1} - \boldsymbol{N}^k}{\Delta t} = -(\boldsymbol{I} - \boldsymbol{N}^k \otimes \boldsymbol{N}^k)\nabla \cdot g(\|(\nabla\boldsymbol{N}^k)(\boldsymbol{I} - \boldsymbol{P})\|^2)\nabla\boldsymbol{N}^k(\boldsymbol{I} - \boldsymbol{P}) \tag{12}$$

where g is the derivative of G with respect to its argument and $\boldsymbol{P}$ is kept constant across all iterations. The second step involves finding new data ϕ_{new} which fits the best the recently smoothed normal field i.e., the resulting $\boldsymbol{N}$ from the solution of (12). This is obtained by now minimising

$$\min_{\boldsymbol{\phi}} \int_{\Omega} (|\nabla\phi| - \nabla\phi \cdot \boldsymbol{N})\, d\boldsymbol{x} \tag{13}$$

through solving

$$\frac{\phi^{k+1}-\phi^k}{\Delta t} = -\nabla \cdot \left(\frac{\nabla \phi^k}{\|\nabla \phi^k\|} - \boldsymbol{N} \right). \tag{14}$$

This two step process can be cyclically repeated and the solution from 14 taken as the true minimizer of (10). Details of the aforementioned process are shown later on in Algorithm 4.1.

The authors in [17] did not go further in analysing the two-step method of Algorithm 4.1 when applied to the *analogue model* of (10). In this paper, we will closely follow the steps of [17] and will show that Algorithm 4.1 indeed finds the true minimiser of the *analogue model*. Even more we will show how to extend this method to the other two curvature based models to be reviewed here.

3.2 *The mean curvature model – MC [9]*

Our second model to review involves the Willmore energy

$$\min_{\phi} \int_{\Omega} \kappa_M^2 |\nabla \phi| d\boldsymbol{x} \tag{15}$$

as introduced in [9], where the numerical solution was obtained by evolving the following fourth order parabolic equation

$$\frac{\partial \phi}{\partial t} = -|\nabla \phi| \nabla \cdot \left(\frac{\kappa_M^2 \nabla \phi}{2|\nabla \phi|} + (\boldsymbol{I} - \boldsymbol{P}) \frac{\nabla(\kappa_M |\nabla \phi|)}{|\nabla \phi|} \right). \tag{16}$$

The boundary conditions are

$$\frac{\nabla \phi}{|\nabla \phi|} \cdot \nu = 0, \qquad \nabla \left(|\nabla \phi| \kappa_M \right) \cdot \nu = 0 \tag{17}$$

where ν is the unit outward normal on $\partial\Omega$

3.3 *The Gaussian curvature model – GC [10]*

In [10] the authors used the Gaussian curvature κ_G and proposed the model

$$\min_{\phi} \int_{\Omega} |\kappa_G| |\nabla \phi| d\boldsymbol{x}. \tag{18}$$

They argued that this model is the analogue, in the context of geometry processing, to the total variation based image denoising model of Rudin, Osher and Fatemi (ROF) [14]. This model, as the total variation regularization does for monotone functions, treats convex shapes as noise-free assigning to them the least possible energy [9]. To solve the above minimisation, the authors avoided the numerical treatment of a very stiff fourth order PDE which arises from taking the first variation of (18) and went for a simpler explicit representation of the surface using triangulation.

4 The TS algorithm

As mentioned, the TS algorithm has been already successfully tested on a number of different problems. This algorithm is a good option when a direct fast solution method for a high order PDE is not at hand. Here we will review

with detail its foundations and will show that for analogue models to those from Section 3 delivers the right solution.

We first present a general two step algorithm for the minimisation of high order variational functionals of the type

$$\min_{\phi} F_{\phi} = \int_{\Omega} G(\phi) d\boldsymbol{x}. \tag{19}$$

Here ϕ may represent image intensities as in [1, 13, 19], surface values as in [17] or any other type of data. When (19) involves second order information, such as any of the curvature definitions introduced above, its associated Euler-Lagrange equation or first order condition results to be a fourth order PDE.

Trying to solve directly such a high order PDE represents many challenges. For instance, fourth order PDEs do not obey a maximum principle so level sets can easily collide during the iterative evolution of the data causing instabilities in the algorithm; efficient and accurate discretisation is also hard to obtain and computationally very costly; non-standard stable numerical algorithms, as in [2, 6, 23, 24], have to be implemented.

The idea of simplifying the numerical solution by splitting the process in two steps, each one involving solving simpler PDEs, is by no means new and has been recursively used by other authors. We recommend consulting [1,13,17,19] and references therein for extensive treatment of these methods.

Two-step methods work as follows: in the first step, the normal field is smoothed to remove high oscillations (denoising) or to interpolate the normal field vectors (inpainting). Once this step has finished, the second involves reconstructing the data from the new field.

On the following, we will follow the steps of [17] to show that provided $G(\phi)$ can be expressed as a function of $\boldsymbol{N}$, i.e. $G = G(\phi, \boldsymbol{N})$, (19) can be minimised in a two step process by first solving

$$\min_{\boldsymbol{N}} F_{\phi} = \int_{\Omega} G(\phi) d\boldsymbol{x} \tag{20}$$

and then

$$\min_{\phi} E_{\phi} = \int_{\Omega} D(\phi) d\boldsymbol{x} \tag{21}$$

where $D(\phi) = |\nabla\phi| - \nabla\phi \cdot \boldsymbol{N}$.

Remark 1 It is important to point out that for functionals of the form $F_{\phi} = \int_{\Omega} G(\phi) d\boldsymbol{x}$, as the one presented above, the TS algorithm succeeds in finding the true solution of the minimisation problem (19). Indeed, the rest of this section will be devoted to prove this fact.

This however does not happen, as it was shown in [17], when the functional has the form $\int_{\Omega} G(\phi)|\nabla\phi| d\boldsymbol{x}$ where the TS algorithm only finds an approximate solution.

4.1 *The first step*

To minimise (19), the first condition $\frac{dG(\phi)}{d\phi} = 0$ needs to be satisfied. However, and on the contrary to normal procedures, the equation given at the end of this section will show that $\frac{dG(\phi)}{d\phi}$ can be expressed as a function of $\frac{dG(\phi)}{d\boldsymbol{N}}$. Hence the name *vector field smoothing* given to this step is appropriate since energy reduction of (19) is accomplished by minimising G with respect to $\boldsymbol{N}$ as in (20). The exact formula of $\frac{dG(\phi)}{d\boldsymbol{N}}$ for each model will be given until Section 5.

We now proceed to obtain the aforementioned expression for $\frac{dG(\phi)}{d\phi}$. This can be done by taking the following steps:

$$\frac{dF_\phi}{d\phi} = \int_\Omega \frac{dG(\phi)}{d\phi} d\boldsymbol{x}. \tag{22}$$

Using the above equation and the identity

$$\frac{dG(\phi)}{d\phi} d\phi = \frac{dG(\phi)}{d\boldsymbol{N}} \cdot d\boldsymbol{N} \tag{23}$$

we can express dF_ϕ as follows

$$\begin{aligned} dF_\phi &= \int_\Omega \frac{dG(\phi)}{d\phi} d\phi d\boldsymbol{x} \\ &= \int_\Omega \frac{dG(\phi)}{d\boldsymbol{N}} \cdot d\boldsymbol{N} d\boldsymbol{x}. \end{aligned} \tag{24}$$

The term $d\boldsymbol{N}$ can be rewritten as follows

$$d\boldsymbol{N} = d\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = d\left(\frac{\nabla\phi}{(\nabla\phi\cdot\nabla\phi)^{1/2}}\right) = \frac{d(\nabla\phi)}{|\nabla\phi|} - \frac{d(\nabla\phi\cdot\boldsymbol{N})}{|\nabla\phi|}\boldsymbol{N}. \tag{25}$$

This let us rewrite (24) as follows

$$dF_\phi = \int_\Omega \frac{dG(\phi)}{d\boldsymbol{N}} \cdot \left(\frac{d(\nabla\phi)}{|\nabla\phi|} - \frac{d(\nabla\phi\cdot\boldsymbol{N})}{|\nabla\phi|}\boldsymbol{N}\right) d\boldsymbol{x}$$

$$= \int_\Omega \frac{dG(\phi)}{d\boldsymbol{N}} \cdot \frac{d(\nabla\phi)}{|\nabla\phi|} d\boldsymbol{x} - \int_\Omega \frac{d(\nabla\phi\cdot\boldsymbol{N})}{|\nabla\phi|}\frac{dG(\phi)}{d\boldsymbol{N}} \cdot \boldsymbol{N} d\boldsymbol{x}. \tag{26}$$

The second term in (26) can be neglected if the constraint $dG/d\boldsymbol{N}\cdot\boldsymbol{N} = 0$ is imposed. Requiring $dG/d\boldsymbol{N}\cdot\boldsymbol{N} = 0$ means no changes of the normal map in the normal direction so if we start with a normal map with unit length constraint, this feature will remain within the evolution of the surface. In our algorithms, this constraint will be enforced numerically by using a projection operator as it will be shown in Section 5.

Thus after neglecting the second term in (26) we get

$$\begin{aligned} dF_\phi &= \int_\Omega \frac{dG(\phi)}{d\boldsymbol{N}} \cdot \frac{d(\nabla\phi)}{|\nabla\phi|} d\boldsymbol{x} \\ &= \int_\Omega \left(\frac{1}{|\nabla\phi|}\right) \frac{dG(\phi)}{d\boldsymbol{N}} \cdot \nabla(d\phi)\, d\boldsymbol{x} \\ &= -\int_\Omega \nabla\cdot\left(\frac{1}{|\nabla\phi|}\frac{dG(\phi)}{d\boldsymbol{N}}\right) d\phi d\boldsymbol{x} + \int_\Gamma \left(\frac{1}{|\nabla\phi|}\frac{dG(\phi)}{d\boldsymbol{N}}\right) d\phi\cdot\boldsymbol{n} d\Gamma \end{aligned} \tag{27}$$

leading to

$$\frac{dF_\phi}{d\phi} = -\int_\Omega \nabla\cdot\left(\frac{1}{|\nabla\phi|}\frac{dG(\phi)}{d\boldsymbol{N}}\right) d\boldsymbol{x} \tag{28}$$

where to drop the boundary term, it has been assumed that the energy flow across the boundary Γ is zero. Hence, Neumann boundary conditions for Ω

have been considered. Finally we have the required expression for $\frac{dG(\phi)}{d\phi}$ and the first optimality condition

$$\frac{dG(\phi)}{d\phi} \equiv -\nabla \cdot \left(\frac{1}{|\nabla\phi|} \frac{dG(\phi)}{d\boldsymbol{N}} \right) = 0. \tag{29}$$

4.2 *The second step*

This step involves minimising (21) with respect to ϕ, hence the first condition $\frac{dD(\phi)}{d\phi} = 0$ have to be satisfied. This is

$$\begin{aligned} \frac{dE_\phi}{d\phi} &= \int_\Omega \frac{dD(\phi)}{d\phi} d\boldsymbol{x} \\ &= \int_\Omega \frac{d}{d\phi} \left(|\nabla\phi| - \nabla\phi \cdot \boldsymbol{N} \right) d\boldsymbol{x} \\ &= -\int_\Omega \nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} d\boldsymbol{x} + \int_\Omega \nabla \cdot \boldsymbol{N} d\boldsymbol{x} \end{aligned} \tag{30}$$

where also Neumann boundary condition was used to drop the boundary term. Hence the expression for $\frac{dD(\phi)}{d\phi}$ and first condition are given by

$$\frac{dD(\phi)}{d\phi} \equiv -\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} + \nabla \cdot \boldsymbol{N} = 0. \tag{31}$$

4.3 *Analysis*

The algorithm just described would fail if the second step would not be reducing the energy in the surface. Being a two step process, both steps must be tuned to perform the same task: minimisation of (19). Therefore to guarantee this, the following equation must be true

$$\frac{dG(\phi)}{d\phi} = \frac{dD(\phi)}{d\phi}. \tag{32}$$

In Section 4.1, equation (29) states that a reduction of the energy can be obtained by minimising G with respect to $\boldsymbol{N}$. Again from (29), it can be seen that a feasible way to achieve this would be to smooth the vector field by solving $\frac{dG(\phi)}{d\boldsymbol{N}} = 0$. In this fashion, [17] proposed to use the iterative procedure shown in Algorithm 4.1. However we shall show that equation (32) cannot be accomplished by Algorithm 4.1, and a slightly modified algorithm (Algorithm 4.2) can lead to satisfying (32).

Algorithm 4.1 The algorithm of [17]

Require: ϕ^0, IN1, IN2, OUT
  $n = 0$
  Compute the normal vector field $\boldsymbol{N}^0$ and the projection operator $\boldsymbol{P}$ using the available ϕ^0
  while $n <$ OUT **do**
    ***** The First Step *****
    for k=0 to IN1 **do**
      With $\boldsymbol{N}^n$ as initial guess, and using a gradient descent method, solve $\frac{dG}{d\boldsymbol{N}} = 0$ keeping ϕ^n lagged i.e. not updating $\boldsymbol{P}$
      $\boldsymbol{N}^{k+1} = \boldsymbol{N}^k - \Delta t \frac{dG}{d\boldsymbol{N}}$
    end for

Update $\boldsymbol{N}$ by doing $\boldsymbol{N}^{n+1} = \boldsymbol{N}^{\mathrm{IN1}}$
***** The Second Step *****
for k=0 to IN2 **do**
With ϕ^n as initial guess, and using a gradient descent method, solve $\frac{dD}{d\phi} = 0$ this time keeping $\boldsymbol{N}^{n+1}$ lagged
$\phi^{k+1} = \phi^k - \Delta t \frac{dD}{d\phi}$
end for
Update ϕ^n by doing $\phi^{n+1} = \phi^{\mathrm{IN2}}$
Update $\boldsymbol{P}$ using ϕ^{n+1}
$n = n + 1$
end while
OUT, IN1, IN2 are the maximum number of iterations of each loop

The whole first step in Algorithm 4.1 may be seen as

$$\boldsymbol{N}^{n+1} = \boldsymbol{N}^n - \frac{dG(\phi)}{d\boldsymbol{N}}, \tag{33}$$

if we take $\Delta t = 1/\mathrm{IN1}$, where the second term is evaluated at some $\boldsymbol{N}$. At the start of the second step, we see that the quantity from (31) is

$$\frac{dD(\phi)}{d\phi} = -\nabla \cdot \frac{\nabla \phi^n}{|\nabla \phi^n|} + \nabla \cdot \boldsymbol{N}, \tag{34}$$

where $\boldsymbol{N} = \boldsymbol{N}^{n+1}$. Then from (33), we have

$$\frac{dD(\phi)}{d\phi} = -\nabla \cdot \frac{\nabla \phi^n}{|\nabla \phi^n|} + \nabla \cdot \left(\boldsymbol{N}^n - \frac{dG(\phi)}{d\boldsymbol{N}} \right) \tag{35}$$

where we note that $\nabla \cdot \frac{\nabla \phi^n}{|\nabla \phi^n|} = \nabla \cdot \boldsymbol{N}^n$ from a previous iteration and therefore

$$\frac{dD(\phi)}{d\phi} = -\nabla \cdot \left(\frac{dG(\phi)}{d\boldsymbol{N}} \right). \tag{36}$$

Then we have the undesirable result

$$-\nabla \cdot \left(\frac{dG(\phi)}{d\boldsymbol{N}} \right) \neq -\nabla \cdot \left(\frac{1}{|\nabla \phi|} \frac{dG(\phi)}{d\boldsymbol{N}} \right) \quad \Rightarrow \quad \frac{dD(\phi)}{d\phi} \neq \frac{dG(\phi)}{d\phi} \tag{37}$$

since $|\nabla \phi| \neq 1$ in general, implying that Algorithm 4.1 will not be successful for solving (19).

From the above simple analysis, we see that if we slightly modify the evolution in (33) as follows

$$\boldsymbol{N}^{n+1} = \boldsymbol{N}^n - \frac{1}{|\nabla \phi|} \frac{dG(\phi)}{d\boldsymbol{N}} \tag{38}$$

we have the desired result

$$\frac{dD(\phi)}{d\phi} = -\nabla \cdot \left(\frac{1}{|\nabla \phi|} \frac{dG(\phi)}{d\boldsymbol{N}} \right) = \frac{dG(\phi)}{d\phi}. \tag{39}$$

This leads to a new algorithm as shown in Algorithm 4.2.

Algorithm 4.2 A modified algorithm

Require: ϕ^0, IN1, IN2, OUT
$n = 0$

Compute the normal vector field $\boldsymbol{N}^0$ and the projection operator $\boldsymbol{P}$ using the available ϕ^0
while $n <$ OUT **do**
***** The First Step *****
for k=0 to IN1 **do**
With $\boldsymbol{N}^n$ as initial guess, and using a gradient descent method, solve $\frac{dG}{d\boldsymbol{N}} = 0$ keeping ϕ^n lagged i.e. not updating $\boldsymbol{P}$
$\boldsymbol{N}^{k+1} = \boldsymbol{N}^k - \frac{\Delta t}{|\nabla \phi^n|} \frac{dG}{d\boldsymbol{N}}$
end for
Update $\boldsymbol{N}$ by doing $\boldsymbol{N}^{n+1} = \boldsymbol{N}^{\mathrm{IN1}}$
***** The Second Step *****
for k=0 to IN2 **do**
With ϕ^n as initial guess, and using a gradient descent method, solve $\frac{dD}{d\phi} = 0$ this time keeping $\boldsymbol{N}^{n+1}$ lagged
$\phi^{k+1} = \phi^k - \Delta t \frac{dD}{d\phi}$
end for
Update ϕ^n by doing $\phi^{n+1} = \phi^{\mathrm{IN2}}$
Update $\boldsymbol{P}$ using ϕ^{n+1}
$n = n + 1$
end while
OUT, IN1, IN2 are the maximum number of iterations of each loop

In the Algorithms 5.1, 5.2 and 5.3 to be introduced for each model in the next section, the iterative scheme (38) will be used in the first step, specifically over (42), (46) and (48).

5 The TS method for curvature models

In what follows, we will derive for each one of models reviewed in Section 3, the second order PDE that has to solved in the first leg of the two-step process. More specifically we will derive the correspondent $\frac{dG(\phi)}{d\boldsymbol{N}} = 0$ to each model. We note that from each model we drop the term $|\nabla\phi|$ from the functional to guarantee obtaining the correct solution from the two-step algorithm as shown before. Removing this term simplifies computations and has been done before in other applications [17, 25] without modifying the model properties so we expect to be the same here. The second step is common for all models and was already been derived in (31) but for sake of completeness we repeat it here. It consists on solving $-\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|} + \nabla \cdot \boldsymbol{N} = 0$ with appropriate boundary conditions.

Remark 2 In each model, it is important to ensure that the solution satisfies the unit length constraint. This can be done either by brute force or by using the projection operator $\boldsymbol{I} - \boldsymbol{N} \otimes \boldsymbol{N}$ at every iteration forcing the changes of $\boldsymbol{N}$ to be perpendicular to itself. We selected the second option.

5.1 *The smoothing step*

This step is sometimes named as the smoothing step in denoising algorithms

Total curvature model - isotropic case The isotropic case of the total curvature model writen in the level set form is the given by

$$\min_{\boldsymbol{N}} F_{(\phi,\boldsymbol{N})} = \int_\Omega |(\nabla \boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})|^2 \, d\boldsymbol{x}. \tag{40}$$

To compute the first order condition we proceed as follows

$$\begin{aligned}\frac{dF}{d\boldsymbol{N}} &= \frac{d}{d\epsilon}\left[\int_\Omega \left|\nabla(\boldsymbol{N}+\epsilon\boldsymbol{\psi})\left(\boldsymbol{I}-\boldsymbol{P}\right)\right|^2 d\boldsymbol{x}\right]_{\epsilon=0}\\ &= \int_\Omega 2\left(\boldsymbol{I}-\boldsymbol{P}\right)\nabla\boldsymbol{N}\cdot\nabla\boldsymbol{\psi}\; d\boldsymbol{x}\\ &= -2\int_\Omega \nabla\cdot\left(\left(\nabla\boldsymbol{N}\right)\left(\boldsymbol{I}-\boldsymbol{P}\right)\right)\boldsymbol{\psi} d\boldsymbol{x} + \int_\Gamma \left(\nabla\boldsymbol{N}\right)\left(\boldsymbol{I}-\boldsymbol{P}\right)\cdot\boldsymbol{n} d\Gamma. \end{aligned} \tag{41}$$

Then selecting as boundary condition $\left(\nabla\boldsymbol{N}\right)\left(\boldsymbol{I}-\boldsymbol{P}\right)\cdot\boldsymbol{n}=0$ the boundary term is dropped resulting in

$$-\nabla\cdot\left(\left(\nabla\boldsymbol{N}\right)\left(\boldsymbol{I}-\boldsymbol{P}\right)\right)=0 \tag{42}$$

Algorithm 5.1 TS algorithm for the Total Curvature Model

Require: ϕ^0
  for n=0 to OUT **do**
    Compute the normal vector field $\boldsymbol{N}$ and the projection operator $\boldsymbol{P}$ using the available ϕ^n
    for k=0 to IN1 **do**
      Solve (42) by $\frac{\partial\boldsymbol{N}}{\partial t}=\frac{1}{|\nabla\phi|}(\boldsymbol{I}-\boldsymbol{N}\otimes\boldsymbol{N})\nabla\cdot\left(\left(\nabla\boldsymbol{N}\right)\left(\boldsymbol{I}-\boldsymbol{P}\right)\right)$ for $\boldsymbol{N}$ keeping ϕ lagged i.e. not updating $\boldsymbol{P}$
    end for
    Using the recently updated $\boldsymbol{N}$ in the previous loop iterate to find new ϕ
    for k=0 to IN2 **do**
      Solve $-\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}+\nabla\cdot\boldsymbol{N}=0$ this time keeping $\boldsymbol{N}$ lagged
    end for
  end for
  OUT, IN1, IN2 are the maximum number of iterations of each loop

Mean curvature model The level set expression for the mean curvature model is given by

$$\min_{\boldsymbol{N}} F_{(\phi,\boldsymbol{N})}=\int_\Omega\left(\nabla\cdot\frac{\nabla\phi}{|\nabla\phi|}\right)^2 d\boldsymbol{x} \tag{43}$$

which can be rewritten in terms of $\boldsymbol{N}$ as

$$\min_{\boldsymbol{N}} F_{(\phi,\boldsymbol{N})}=\int_\Omega\left(\nabla\cdot\boldsymbol{N}\right)^2 d\boldsymbol{x}. \tag{44}$$

The first order condition is computed as follows

$$\begin{aligned}\frac{dF}{d\boldsymbol{N}} &= \frac{d}{d\epsilon}\left[\int_\Omega \left(\nabla\cdot\left(\boldsymbol{N}+\epsilon\boldsymbol{\psi}\right)\right)^2 d\boldsymbol{x}\right]_{\epsilon=0}\\ &= \int_\Omega 2\left(\nabla\cdot\boldsymbol{N}\right)\left(\nabla\cdot\boldsymbol{\psi}\right) d\boldsymbol{x}\\ &= -\int_\Omega 2\nabla\left(\nabla\cdot\boldsymbol{N}\right)\boldsymbol{\psi} d\boldsymbol{x} + \int_\Gamma \boldsymbol{N}\cdot\boldsymbol{n} d\Gamma \end{aligned} \tag{45}$$

where by selecting as boundary condition $\boldsymbol{N}\cdot\boldsymbol{n}=0$ let us to drop the boundary term resulting in

$$-\nabla\left(\nabla\cdot\boldsymbol{N}\right)=0 \tag{46}$$

Algorithm 5.2 TS algorithm for the Mean Curvature Model

Require: ϕ^0
  for n=0 to OUT **do**
    Compute the normal vector field $\boldsymbol{N}$ and the projection operator $\boldsymbol{P}$ using the available ϕ^n
    for k=0 to IN1 **do**
      Solve (46) by $\frac{\partial \boldsymbol{N}}{\partial t} = \frac{1}{|\nabla \phi|}(\boldsymbol{I} - \boldsymbol{N} \otimes \boldsymbol{N})\nabla (\nabla \cdot \boldsymbol{N})$
    end for
    Using the recently updated $\boldsymbol{N}$ in the previous loop iterate to find new ϕ
    for k=0 to IN2 **do**
      Solve $-\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} + \nabla \cdot \boldsymbol{N} = 0$ this time keeping $\boldsymbol{N}$ lagged
    end for
  end for
  OUT, IN1, IN2 are the maximum number of iterations of each loop

Gaussian curvature model By using the identity, $\kappa_G = \frac{1}{2}\left(\kappa_M^2 - \kappa_T^2\right)$ we can rewrite the analogue model as follows

$$\min_{\phi} \int_{\Omega} \frac{1}{2} \left|\kappa_M^2 - \kappa_T^2\right| d\boldsymbol{x}. \tag{47}$$

From the previous two models it is easy to write the first order condition for the Gaussian-curvature-based model i.e.

$$-\nabla \cdot ((\nabla \boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})) - \nabla (\nabla \cdot \boldsymbol{N}) = 0 \tag{48}$$

with corresponding boundary conditions.

Algorithm 5.3 TS algorithm for the Gaussian Curvature Model

Require: ϕ^0
  for n=0 to OUT **do**
    Compute the normal vector field $\boldsymbol{N}$ and the projection operator $\boldsymbol{P}$ using the available ϕ^n
    for k=0 to IN1 **do**
      Solve (48) by $\frac{\partial \boldsymbol{N}}{\partial t} = (\frac{1}{|\nabla \phi|}\boldsymbol{I} - \boldsymbol{N} \otimes \boldsymbol{N})\nabla \cdot ((\nabla \boldsymbol{N})(\boldsymbol{I} - \boldsymbol{P})) - \nabla (\nabla \cdot \boldsymbol{N})$ for $\boldsymbol{N}$ keeping ϕ lagged i.e. not updating $\boldsymbol{P}$
    end for
    Using the recently updated $\boldsymbol{N}$ in the previous loop iterate to find new ϕ
    for k=0 to IN2 **do**
      Solve $-\nabla \cdot \frac{\nabla \phi}{|\nabla \phi|} + \nabla \cdot \boldsymbol{N} = 0$ this time keeping $\boldsymbol{N}$ lagged
    end for
  end for
  OUT, IN1, IN2 are the maximum number of iterations of each loop

6 Numerical implementation

We proceed to outline the discretisation scheme we use for the differential operators involved in each one of the TS algorithms just described. From now on, we assume a continuous domain $\Omega = [0, m] \times [0, n] \times [0, p]$ and let (h_x, h_y, h_z) to represent a vector of finite mesh sizes, then we define the infinite grid by

$$G_h = \{(x, y, z) : x = x_i = ih_x, \;\; y = y_j = jh_y; \;\; z = z_k = kh_z; \;\; i, j, k \in \mathbb{Z}\} \tag{49}$$

$\Omega_h = \Omega \cap G_h$ and $u_h = \phi_h(x,y,z) = \phi_h(x_i,y_j,z_k) = \phi_h(ih_x, jh_y, kh_z)$ the discrete version of any function ϕ defined on Ω_h. For simplicity we will assume that $m = n = p$ and $h = h_x = h_y = h_z$. We will use $(\cdot)^{\Lambda}_{\psi}$ to denote the derivative with respect to any variable ψ, where the symbol $\Lambda = \{+,-,c\}$ is used to differentiate among forward, backward and central differences. For instance,

$$
\begin{aligned}
(\phi_x^+)_{i,j,k} &= (\phi_{i+1,j,k} - \phi_{i,j,k})/h, \\
(\phi_x^-)_{i,j,k} &= (\phi_{i,j,k} - \phi_{i-1,j,k})/h, \\
(\phi_x^c)_{i,j,k} &= (\phi_{i+1,j,k} - \phi_{i-1,j,k})/2h,
\end{aligned}
$$

denote the forward, backward and central differences of ϕ on the direction x at point (i,j,k) respectively.

6.1 *Discretisation of the first step*

To begin with, discretisation of the first step involves computing the projection operator $\boldsymbol{I} - \boldsymbol{P}$ which will remain lagged along the iterations. The expression for $\boldsymbol{P}$ at every (i,j,k) point is given by

$$
\boldsymbol{P} = \frac{1}{|\nabla\phi|}\begin{pmatrix} (\phi_x^c)^2 & \phi_x^c\phi_y^c & \phi_x^c\phi_z^c \\ \phi_y^c\phi_x^c & (\phi_y^c)^2 & \phi_y^c\phi_z^c \\ \phi_z^c\phi_x^c & \phi_z^c\phi_y^c & (\phi_z^c)^2 \end{pmatrix}. \tag{50}
$$

The norm $|\nabla\phi|$ is computed as follows

$$
|\nabla\phi| = \sqrt{(\phi_x^c)^2 + (\phi_y^c)^2 + (\phi_z^c)^2 + \beta}
$$

where β is an small parameter to avoid division by zero.

The next operator that needs to be approximated is $\nabla\boldsymbol{N}$. To this end, first $\boldsymbol{N} = \left[N^1, N^2, N^3\right]$ is discretized as follows

$$
\boldsymbol{N} = \left[\phi_x^c, \phi_y^c, \phi_z^c\right]/|\nabla\phi|, \tag{51}
$$

and the matrix

$$
\nabla\boldsymbol{N} = \begin{pmatrix} (N^1)_x^+ & (N^1)_y^+ & (N^1)_z^+ \\ (N^2)_x^+ & (N^2)_y^+ & (N^2)_z^+ \\ (N^3)_x^+ & (N^3)_y^+ & (N^3)_z^+ \end{pmatrix}. \tag{52}
$$

To approximate the 3×1 vector $\boldsymbol{A} = \left[A^1, A^2, A^3\right]$ which results from $\boldsymbol{A} = \nabla\cdot((\nabla\boldsymbol{N})(\boldsymbol{I}-\boldsymbol{P}))$, first the matrix product $\boldsymbol{B} = (\nabla\boldsymbol{N})(\boldsymbol{I}-\boldsymbol{P})$ is computed and then, the divergence operator applied row-wise to $\boldsymbol{B}$ using backward differences.

Now, to approximate $\nabla(\nabla\cdot\boldsymbol{N})$, first forward differences are applied to each entry of $\boldsymbol{N}$ and the divergence computed as

$$
\nabla\cdot\boldsymbol{N} = (N^1)_x^+ + (N^2)_y^+ + (N^3)_z^+ \tag{53}
$$

then the gradient vector $\nabla(\nabla\cdot\boldsymbol{N})$ is computed using backward differences.

Finally, an explicit time marching scheme is used to evolve the parabolic PDE. For instance,

$$
\frac{\boldsymbol{N}^{n+1} - \boldsymbol{N}^n}{\Delta t} = -\nabla\cdot((\nabla\boldsymbol{N})(\boldsymbol{I}-\boldsymbol{P})) - \nabla(\nabla\cdot\boldsymbol{N}) \tag{54}
$$

is used for the Gaussian curvature model at every point. The implementation for the other two models is straightforward from here.

6.2 *Discretisation of the second step*

Here we use a staggered discretization. For instance, to approximate $\nabla \cdot \boldsymbol{V} = (V^1)_x + (V^2)_y + (V^3)_z$ for any $\boldsymbol{V} = [V^1, V^2, V^3]$ at some pixel (i, j, k) we use central differences between ghost half-points as follows

$$(\nabla \cdot V)_{i,j} = \frac{\left(V^1_{i+\frac{1}{2},j} - V^1_{i-\frac{1}{2},j}\right)}{h} + \frac{\left(V^2_{i,j+\frac{1}{2}} - V^2_{i,j-\frac{1}{2}}\right)}{h} + \frac{\left(V^3_{i,j,k+\frac{1}{2}} - V^3_{i,j,k-\frac{1}{2}}\right)}{h}. \tag{55}$$

This way

$$h\left(\nabla \cdot \frac{\nabla\phi}{|\nabla\phi|}\right)_{i,j,k} = \frac{(\phi_x)_{i+\frac{1}{2},j,k}}{|\nabla\phi|_{i+\frac{1}{2},j,k}} - \frac{(\phi_x)_{i-\frac{1}{2},j,k}}{|\nabla\phi|_{i-\frac{1}{2},j,k}} + \frac{(\phi_y)_{i,j+\frac{1}{2},k}}{|\nabla\phi|_{i,j+\frac{1}{2},k}} \tag{56}$$

$$- \frac{(\phi_y)_{i,j-\frac{1}{2},k}}{|\nabla\phi|_{i,j-\frac{1}{2},k}} + \frac{(\phi_z)_{i,j,k+\frac{1}{2}}}{|\nabla\phi|_{i,j,k+\frac{1}{2}}} - \frac{(\phi_z)_{i,j,k-\frac{1}{2}}}{|\nabla\phi|_{i,j,k-\frac{1}{2}}}. \tag{57}$$

When appropriate we use *min-mod* derivatives as defined below since as noted in [2, 14] they help to recover sharp edges. The *min-mod* derivative is defined as

$$\textit{min-mod}\ (a, b) = \left(\frac{sgn\ a + sgn\ b}{2}\right) \min(|a|, |b|). \tag{58}$$

We note that *average* derivatives were also tested and indeed we use them to compute the computational complexity of the algorithm in Section 7. For instance to compute the terms at the $(i + \frac{1}{2}, j, k)$ and $(i - \frac{1}{2}, j, k)$ points we proceed as follows:

Partial derivatives in x by the central differencing of two adjacent *whole* pixels

$$(\phi_x)_{i+\frac{1}{2},j,k} = (\phi_{i+1,j,k} - \phi_{i,j,k})/h,$$
$$(\phi_x)_{i-\frac{1}{2},j,k} = (\phi_{i,j,k} - \phi_{i-1,j,k})/h,$$

Partial derivatives in y by the *min-mod* of $(\cdot)_y$'s at two adjacent *whole* points

$$(\phi_y)_{i+\frac{1}{2},j,k} = \textit{min-mod}\left(\frac{1}{2h}(\phi_{i+1,j+1,k} - \phi_{i+1,j-1,k}), \frac{1}{2h}(\phi_{i,j+1,k} - \phi_{i,j-1,k})\right),$$
$$(\phi_y)_{i-\frac{1}{2},j,k} = \textit{min-mod}\left(\frac{1}{2h}(\phi_{i,j+1,k} - \phi_{i,j-1,k}), \frac{1}{2h}(\phi_{i-1,j+1.k} - \phi_{i-1,j-1,k})\right),$$

Partial derivatives in z by the *min-mod* of $(\cdot)_z$'s at two adjacent *whole* points

$$(\phi_z)_{i+\frac{1}{2},j,k} = \textit{min-mod}\left(\frac{1}{2h}(\phi_{i+1,j,k+1} - \phi_{i+1,j,k-1}), \frac{1}{2h}(\phi_{i,j,k+1} - \phi_{i,j,k-1})\right),$$
$$(\phi_z)_{i-\frac{1}{2},j,k} = \textit{min-mod}\left(\frac{1}{2h}(\phi_{i,j,k+1} - \phi_{i,j,k-1}), \frac{1}{2h}(\phi_{i-1,j,k+1} - \phi_{i-1,j,k-1})\right),$$

TS algorithm

Step	TC model	MC model	GC model
1st	116 flops	38 flops	134 flops
2nd	80 flops	80 flops	80 flops

Table 1.: Complexity analysis of the algorithms.

Norm of the gradient vector

$$|\nabla u|_{i+\frac{1}{2},j,k} = \sqrt{((\phi_x)_{i+\frac{1}{2},j,k})^2 + ((\phi_y)_{i,j+\frac{1}{2},k})^2 + ((\phi_z)_{i,j,k+\frac{1}{2}})^2 + \beta},$$
$$|\nabla \phi|_{i-\frac{1}{2},j,k} = \sqrt{((\phi_x)_{i-\frac{1}{2},j,k})^2 + ((\phi_y)_{i,j-\frac{1}{2},k})^2 + ((\phi_z)_{i,j,k-\frac{1}{2}})^2 + \beta}.$$

By a similar procedure we can obtain the approximations for $V^2_{i,j+\frac{1}{2}}$, $V^2_{i,j-\frac{1}{2}}$, $V^3_{i,j,k+\frac{1}{2}}$, and $V^3_{i,j,k-\frac{1}{2}}$. Finally, the Neumann's boundary condition on $\partial\Omega$ is treated as

$$\phi_{i,0} = \phi_{i,1}, \quad \phi_{i,n+1} = \phi_{i,n}, \quad \phi_{0,j} = \phi_{1,j}, \quad \phi_{m+1,j} = \phi_{m,j}. \tag{59}$$

7 Complexity analysis

In this section we shall review the computational cost of each algorithm in order to give an idea of the low cost involved in each one of them. We will use flops as comparison measure. For instance, addition, subtraction and multiplication will be assigned 1 flop and division and square root will be assigned 8 flops. In Table 1 we present a summary of the cost of one iteration for each algorithm. It is very clear that the most costly algorithm is the GC model and the cheapest the MC model. The cost for each model is however very small if compared against discretisation of the fourth order PDEs which turns to be at least three times. This is the main strength of the TS algorithm.

8 Results

In this section we present and discuss the results obtained through running the TS algorithm, for each model, from Section 3, over two *test problems*. Our first test problem is the noisy cube shown in Figure 1(a) of size $32 \times 32 \times 32$. The second test problem is the noisier sharp-sphere, containing both sharp and smooth regions, in Figure 1(b) of size $106 \times 113 \times 113$. In both cases, Gaussian noise was added to create a noisy surface and then the noisy surface processed using the publicly available level set toolbox [16] to force the implicit surface to be a signed distant function.

In the TS algorithm both steps were solved using an explicit gradient descent method with manually optimized time-step. In all cases, the selected regularization parameter was $\beta = 10^{-3}$. To stop the algorithms the following stopping criteria was used: in the first leg, stop if the reduction of the relative residual between two consecutive iterations is less than 10^{-4}; in the second leg stop if the change in the energy decreasing is less than one ten thousandth the initial energy value. A maximum of 250 iterations is setup for each leg. As illustrated in Algorithms 5.1, 5.2 and 5.3, the TS algorithm is also ran in cycles. This is, after finishing the second leg, we recomputed the lagged terms and ran the algorithm again. Our observations indicate that no more than 5 cycles are required to have a good surface reconstruction.

As can be appreciated from Table 2, we can see that the TS algorithm required more cycles and CPU time to solving the TC model while it worked

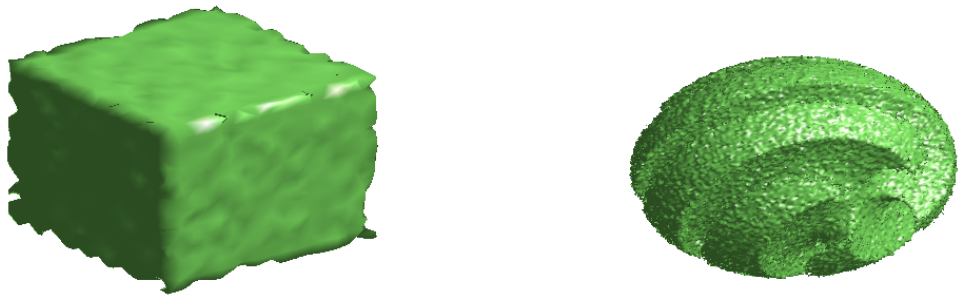
TS Algorithm

Size	TC model		MC model		GC model	
Cube	O	3/73.98 s	A	2/15.71 s	A	3/17.84 s
	A	3/73.03 s	O	2/16.83 s	O	3/19.43 s
Sharp-Sphere	O	3/2898 s	O	2/1373 s	O	3/1381 s
	A	3/2867 s	A	2/1438 s	A	3/1447 s

Table 2.: Number of cycles and CPU times for the model problems. A stands for Analogue and O for Original

pretty well for the other two models. The CPU-times obtained for the second test problem also show the need to improve the TS algorithm. This maybe done by using a nonlinear multigrid algorithm for each leg and maybe will be our future work. From Table 2, we also can deduce that solving the analogue or original model with TS involves almost the same amount of computational effort and time.

To illustrate the quality of reconstruction and to have a reference for comparing the original models against the analogue models, we present the results for the first test problem in Figure 2. These results must be interpreted with caution since the TS algorithm does not solve accurately the original model, see [17]. From Figure 2 we can see that solving any, original or analogue, model with TS delivers visually almost the same kind of reconstruction.



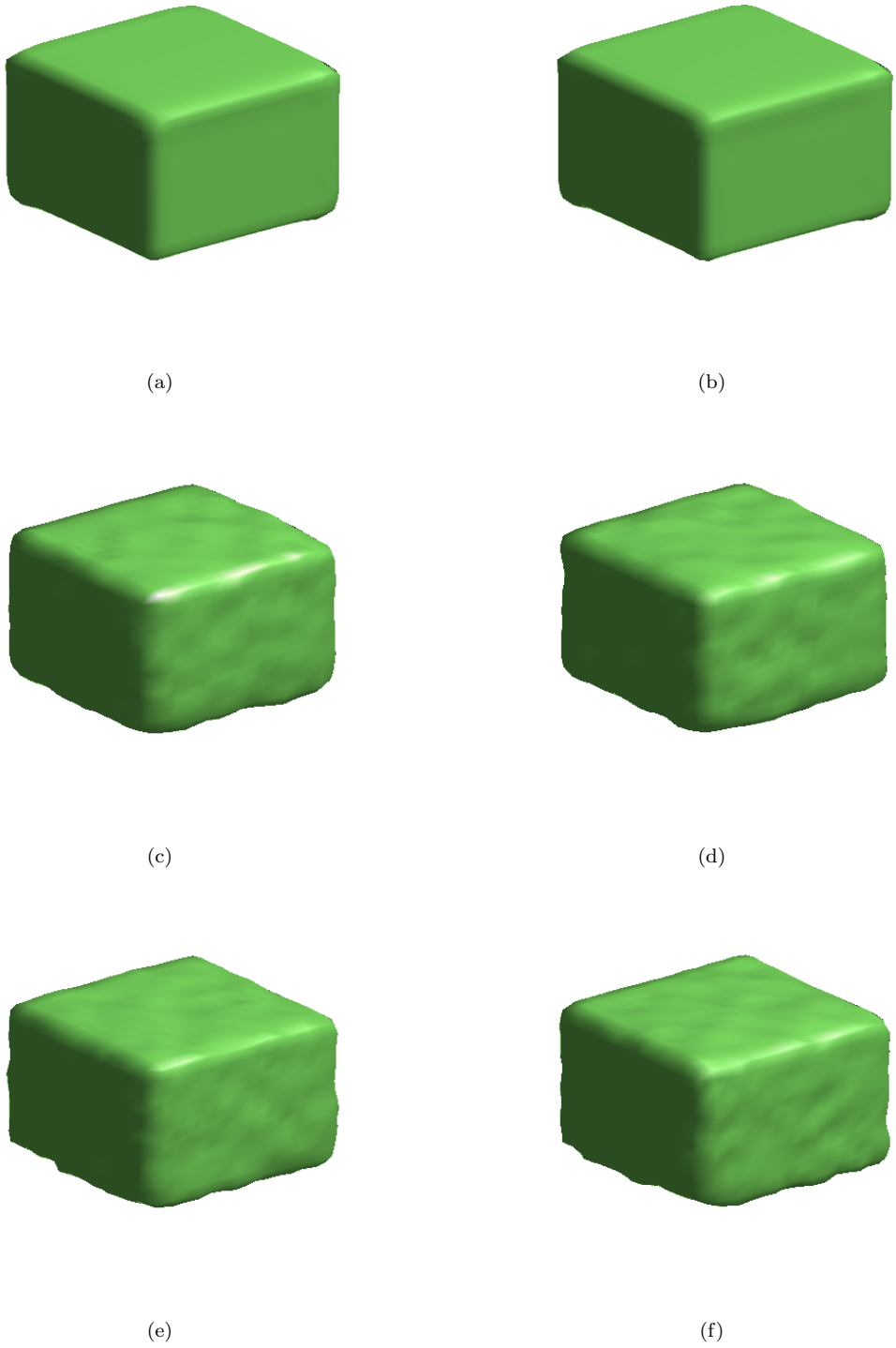
(a) (b)

Figure 1.: (a) Noisy Cube. (b) Noisy Sphere (courtesy of Dr. Rongjie Lai, University of Southern California).

Finally in Figure 3, we show the results of TS solving the analogue model for the second test problem . In this case, the MC model result in Figure 3 (a), shows that this model as expected delivers better restoration in the smooth regions but tends to round the sharp edges. On the contrary, is apparent from Figures 3 (b) and (c) that the GC and TC models preserve better the salient sections while showing some kind of staircase effect in the smooth ones.

9 Discussion

In this paper we have presented a generalisation of a well known method for the numerical solution of high order PDEs. We have shown that the presented two-step method it is a good option for the solution of the analogues of the curvature models . In the context of surface fairing models and up to our knowledge, the TS method had been only applied to the solution of the total curvature model. Here we have shown how to use it for the solution of the MC and GC models as well. By doing this, we are providing a unified algorithm which can be used to make fair comparisons among the three models. We have



(a) (b)

(c) (d)

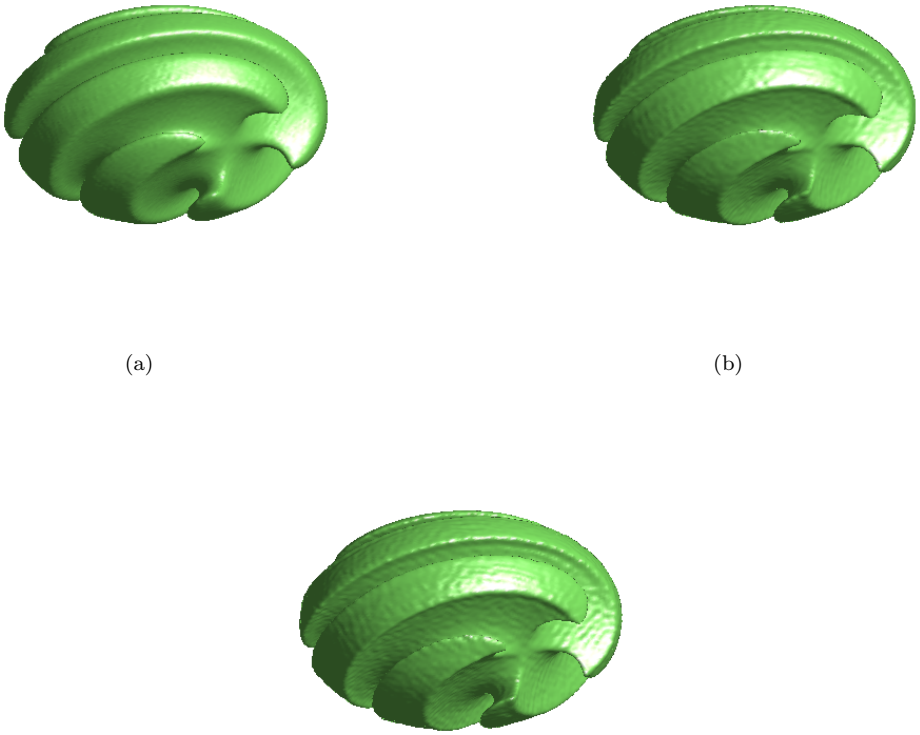
(e) (f)

Figure 2.: (a) analogue MC result. (b) original MC result. (c) analogue GC result. (d) original GC result. (e) analogue TC result. (f) original TC result.

also shown that the TS algorithm proposed is computationally very efficient and cheap compared to direct solvers of the fourth order problem. For the solution of each second order PDE we have used very simple Explicit time marching methods so there is still room for improvement here. For instance, nonlinear multigrid methods maybe used to speed up convergence and this maybe part of our future work.

Reference

[1] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro and J. Verdera, *Filling-in by joint interpolation of vector elds and gray levels*, IEEE Trans. Image Process. 10(8):12001211, 2001.



(a) (b)

(c)

Figure 3.: Results for the *analogue models* (a) MC . (b) GC . (c) TC.

[2] C. Brito-Loeza and K. Chen , *Multigrid Algorithm for High Order Denoising*, SIAM J. on Imaging Sciences , 3(3):363-389, 2004.

[3] C. Brito-Loeza and K. Chen, *Fast Numerical Algorithms for Eulers Elastica Inpainting Model*, International Journal of Modern Mathematics, 5(2):157182, 2010.

[4] Y. Burago and V.A. Zalgaller, *Geometric Inequalities*, Springer-Verlag (February 1988).

[5] T. F. Chan, S. H. Kang, J. Shen, *Euler's Elastica and Curvature-Based Inpainting*, SIAM J. of Appl. Math., 63(2):564-592, 2003.

[6] N. Chumchob, K. Chen, and C. Brito-Loeza , *A Fourth-Order Variational Image Registration Model and Its Fast Multigrid Algorithm*, SIAM J. on Multiscale Model. Simul., 9(1):89-128

[7] M. Desbrun, M Meyer, P. Schröder and A. Barr, *Implicit fairing of irregular meshes using diffusion and curvature flow*, In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (SIGGRAPH '99). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 317-324.

[8] M. Do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall; 1 edition (February 11, 1976)

[9] M. Droske and Martin Rumpf, *A level set formulation for Willmore flow*, Interfaces and free boundaries, 6(3):361-378, 2004.

[10] M. Elsey and S. Esedoḡlu, *Analogue of the Total Variation Denoising Model in the Context of Geometry Processing*, Multiscale Model. Simul. 7(4):1549-1573, 2009.

[11] D. J. Eyre, *Unconditionally gradient stable time marching the Cahn-Hilliard equa- tion*, Computational and Mathematical Models of Microstructural Evolution 53:16861712, 1998.

[12] R. Goldman, *Curvature formulas for implicit curves and surfaces*, Computer aided geometric design, 22:632-658, 2005.

[13] M. Lysaker, S. Osher, and X-C. Tai, *Noise Removal Using Smoothed Normals and Surface Fitting*, IEEE Trans. Image Process. 13(10):13451357, 2004.

[14] L.I. Rudin, S. Osher, E. Fatemi, *Nonlinear total variation based noise removal algorithms*, Physica D, 60:259268, 1992.

[15] R. Schneider and L. Kobbelt, *Geometric fairing of irregular meshes for free-form surface design*, Comput. Aid. Geom. Des. 18:359-379, 2001.

[16] B. Sumengen, *A Matlab toolbox implementing Level Set Methods*, Vision Research Lab at UC Santa Barbara, `http://barissumengen.com/level_set_methods/` .

[17] T. Tasdizen, R. Whitaker, P. Burchard and S. Osher, *Geometric Surface Processing via Normal Maps*, ACM Transactions on Graphics, 22(4):1012-1033, 2003.

[18] T. Tasdizen and R. Whitaker, *Anisotropic diffusion of surface normals for feature preserving surface reconstruction*, 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on, pp. 353-360, 2003.

[19] L. A. Vese and S. Osher, *Numerical methods for p-harmonic ows and applications to image*

processing, SIAM J. Numer. Anal., 40(6):20852104, 2002.
[20] B. P. Vollmayr-Lee and A. D. Rutenberg, *Fast and accurate coarsening simula- tion with an unconditionally stable time step*, Phys. Rev. E 68: 066703.1 066703.13, 2003.
[21] W. Welch and A Witkin, *Variational surface modeling*, SIGGRAPH Comput. Graph., 26(2):157-166, 1992.
[22] W. Welch and A Witkin, *Free-form shape design using triangulated surfaces*, In Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94). ACM, New York, NY, USA, 247-256.
[23] C. Wu, X.-C. Tai, *Augmented Lagrangian method, dual methods, and split bregman iteration for ROF, vectorial TV, and high order models*,SIAM J. on Imaging Sciences, 3(3):300339, 2010.
[24] F. Yang, K. Chen, B. Yu, *Homotopy method for a mean curvature-based denoising model*, Preprint submitted to Applied Numerical Mathematics.
[25] W. Zhu and T. F. Chan, *Image Denoising Using Mean Curvature*, preprint, `http://www.math.nyu.edu/$\sim$wzhu/`.