A VOLUME-BASED METHOD FOR DENOISING ON CURVED SURFACES

Harry Biddle
Double Negative Visual Effects
London, UK
hb@dneg.com

Ingrid von Glehn, Colin B. Macdonald, Thomas M"arz
Mathematical Institute
University of Oxford
Oxford, UK
vonglehn|macdonald|maerz@maths.ox.ac.uk

ABSTRACT

We demonstrate a method for removing noise from images or other data on curved surfaces. Our approach relies on in-surface diffusion: we formulate both the Gaussian diffusion and Perona–Malik edge-preserving diffusion equations in a surface-intrinsic way. Using the Closest Point Method, a recent technique for solving partial differential equations (PDEs) on general surfaces, we obtain a very simple algorithm where we merely alternate a time step of the usual Gaussian diffusion (and similarly Perona–Malik) in a small 3D volume containing the surface with an interpolation step. The method uses a closest point function to represent the underlying surface and can treat very general surfaces. Experimental results include image filtering on smooth surfaces, open surfaces, and general triangulated surfaces.

Index Terms— Image denoising, Surfaces, Partial differential equations, Numerical analysis.

1. INTRODUCTION

Denoising is an important tool in image processing and often forms a crucial step in image acquisition and image analysis. On general curved surfaces, denoising and the corresponding scale space analysis of images, as well as other image processing tasks have seen some interest [1]. For example, Kimmel [2] studies an intrinsic scale space for images on parametrically-defined surfaces via the geodesic curvature flow. More recently, Spira and Kimmel [3] use the same flow for segmentation of images painted on parametrically-defined manifolds. Bogdanova et al. [4] perform scale space analysis as well as segmentation for omnidirectional images defined on various shapes. Our aim here is to provide simple numerical realizations of the Gaussian diffusion filter and the nonlinear Perona–Malik [5, 6, 7] edge-preserving diffusion filter for images on general surfaces. Fig. 1 shows an example comparing the two approaches.

In applications, image processing on curved surfaces can occur even when data is acquired in three-dimensional volumes. For example, Lin and Seales [8] propose CT scanning of scrolls in order to non-destructively read text written on rolled up documents. In digital image-based elasto-tomography (DIET), a new methodology for non-invasive breast cancer screening, the surface and the surface data are measured from strobed lighting with several calibrated digital cameras [9]. In such applications, it may be useful to perform denoising and other techniques on surfaces embedded in three-dimensional volumes. Denoising on surfaces is also related to surface fairing if the surface data is a height-field which gives a perturbed surface of interest as a function on a reference surface [10]. In this context the Perona-Malik model and variations of it have already been considered in the literature [10, 11, 12].

A common theme in many techniques (e.g., [2, 3, 4] above) is the use of models based on partial differential equations (PDEs). Applying PDE-based models on surfaces requires some representation of the geometry and this choice is significant in the complexity of the resulting algorithm. A common choice is a parameterization of the surface; however this introduces artificial distortions and singularities into the model, even for the simplest surfaces such as spheres. On more complex geometry, joining up small patches is typically required. On a triangulated surface, PDEs can be discretized using finite element methods, e.g., [13]. Alternatively, implicit representations such as level sets (e.g., [14]) or the closest point representation employed here can be used. These tend to be flexible with respect to the geometry although level sets do not give an obvious way to treat open surfaces. Using level set methods in a 3D neighbourhood of the surface also requires introducing artificial boundary conditions at the boundary of the computational band.

In this work we use the Closest Point Method [15] for solving both the Gaussian and Perona–Malik diffusion models on surfaces. Using this technique, we keep the resulting evolution as simple as possible by alternating between two straightforward steps:

1. A time step of the model in three dimensions using entirely standard finite difference methods.
2. An interpolation step, which encodes surface geometry and makes the 3D calculation consistent with the surface problem.

One benefit is that the Closest Point Method uses a closest point function [15, 16] to represent the surface, and therefore does not require modification of the model via a parameterization nor does
Suppose we have a surface function $u$ into the surrounding volume by defining \( v : \mathbb{R}^3 \to \mathbb{R} \) by

\[
v(x) := u(cp(x)).
\]

Notably, \( v \) will be constant in the direction normal to the surface. This feature is crucial because it implies that an application of a Cartesian differential operator (such as the gradient or Laplacian) to \( v \) is equivalent to applying the corresponding intrinsic surface differential operator to \( u \). These mathematical principles are established in [15, 16].

2.3. The Closest Point Method

We want to solve the three-dimensional diffusion equation \( \frac{\partial u}{\partial t} = \Delta v \) subject to the constraint that \( v(x) = v(cp(x)) \) (i.e., that \( v \) remains constant in the normal direction). One simple approach to solving this constrained problem is to discretize in time and alternate between advancing the three-dimensional PDE in time and re-extending the results [15]. Using the forward Euler time-stepping algorithm with step size \( \tau \), this gives the following two-step process

\[
\begin{align*}
\tilde{v}^{n+1} &= v^n + \tau \Delta v^n = v^n + \tau \left( v_{xx}^n + v_{yy}^n + v_{zz}^n \right), \\
v^{n+1} &= \tilde{v}^{n+1}(cp(x)).
\end{align*}
\]

Note that the temporary \( \tilde{v}^{n+1} \) contains the result of a forward Euler step using the Cartesian Laplacian, but this might not be constant in the normal direction, so we re-extend to obtain \( v^{n+1} \).

For additional accuracy, this can easily be extended to higher-order Runge–Kutta or multistep methods [15]. Implicit methods are also practical [19, 20] and useful if the problem is stiff.

2.4. The role of voxels

We discretize the three-dimensional embedding volume, typically with a uniform Cartesian grid as this makes the algorithm straightforward to implement and analyze. We will think of this three-dimensional voxel grid as our discrete image with the voxel size parameter \( h \) specifying scale. This could be appropriate if the surface and image data were acquired in three dimensions using e.g., CT scans [8]. In that case, the grid parameter \( h \) could be naturally chosen by the resolution of the CT machine. In other cases, \( h \) may simply be a numerical parameter.

Acquiring closest point representations is a research area in its infancy and for now we assume such a representation is given. However, it is straightforward to convert triangulations into this formation [21] or to perform steepest descent on level-set representations [16].

2.5. Interpolation

Because we are working on a discrete grid, \( cp(x) \) is likely not a grid point and thus we need a way to approximate the extension operation of (3b). One approach is to use interpolation on a stencil consisting of the surrounding grid points nearby \( cp(x) \) [15]. We use a standard tri-cubic interpolant which gives good accuracy and stability; a schematic of the extension is shown in Fig. 2c.

The numerical properties of the Closest Point Method are fairly well-understood [15, 16, 19, 20, 22] and highly accurate computations have been done up to fifth-order accuracy [21]. In some cases, stability analysis has also been performed [22, 23]. The operations can also be accelerated using a multigrid strategy [23] although we do not do so here.
2.6. Banding

We have described the algorithm as taking place on the voxel grid of a full cube. This will work fine in practice but many of the grid points are not required and thus the code can be made more efficient by working on a narrow band of points surrounding the surface. For any particular discretization, the minimum bandwidth is easily computed as a small multiple of \( h \) and using a wider band will have no effect on the solution. Notably, no artificial boundary condition need be imposed at the edges of the band \([15]\).

3. PERONA–MALIK DENOISING

The Perona–Malik equation \([5, 6]\) is a classic modification of Gaussian diffusion that employs edge preservation by varying the diffusion coefficient across the image, penalising it at edges and structures that should be preserved in the image. The equation in its general form is

\[
\frac{\partial u}{\partial t} = \text{div}(g(\nabla u) \nabla u),
\]

(4)

where the image is defined on a finite rectangle \( \Omega \subset \mathbb{R}^2 \) via a function \( u : \mathbb{R}^2 \to \mathbb{R} \) that describes the pixel intensity at any given point, for example where one is white and zero is black. If the diffusion coefficient \( g : \mathbb{R} \to \mathbb{R} \) is taken to be a constant, we recover the Gaussian diffusion equation. Instead, we take the popular choice

\[
g(s) = \frac{1}{1 + s^2}.
\]

(5)

Edges are detected via the gradient: a large gradient \( |\nabla u| \gg \lambda \) means that we are close to an edge and the diffusion almost stops, since \( g \approx 0 \). A small gradient \( |\nabla u| \ll \lambda \) means that we are away from edges, hence we filter (locally) with Gaussian diffusion, since \( g \approx 1 \). Here \( \lambda \) is a tunable parameter that controls the sensitivity of the scheme to visual edges; it gives a threshold to separate noise from edges.

3.1. Perona–Malik on surfaces

To formulate Perona–Malik diffusion on a surface \( \mathcal{S} \), we simply replace operators with in-surface operators:

\[
\frac{\partial u}{\partial t} = \text{div}_S(g(|\nabla_S u|)\nabla_S u).
\]

(6)

The original noisy surface image is the initial condition \( u_0 : \mathcal{S} \to \mathbb{R} \).

We solve this equation for \( t \in [0, t_{\text{final}}] \), where the final time is a second parameter which controls the amount of denoising.

3.2. Numerical discretization

We start with the three-dimensional Cartesian form of (4)

\[
\frac{\partial v}{\partial t} = \frac{\partial}{\partial x}\left[g\left(\sqrt{v_x^2 + v_y^2 + v_z^2}\right)v_x\right] + \frac{\partial}{\partial y}\left[g(\ldots)v_y\right] + \frac{\partial}{\partial z}\left[g(\ldots)v_z\right],
\]

and discretize with a textbook-standard finite difference scheme for parabolic equations. Combined with forward Euler, the time-evolution in 3D is thus given by

\[
\tilde{v}^{n+1}_{i,j,k} = v^{n}_{i,j,k} + \tau \left[ \frac{g^n_{i+1/2,j,k} D^x v^{n}_{i,j,k} - g^n_{i-1/2,j,k} D^x v^{n}_{i,j,k}}{h} + \frac{g^n_{i,j+1/2,k} D^y v^{n}_{i,j,k} - g^n_{i,j-1/2,k} D^y v^{n}_{i,j,k}}{h} + \frac{g^n_{i,j,k+1/2} D^z v^{n}_{i,j,k} - g^n_{i,j,k-1/2} D^z v^{n}_{i,j,k}}{h} \right].
\]

(7a)

Here \( D^x \) and \( D^y \) indicate forward and backward external differences, in the direction indicated by the superscript \( \alpha \), on a grid spacing of \( h \). The expressions involving \( g \) between grid points are calculated as the average of the values at the two neighbouring grid points, e.g.,

\[
g^n_{i+1/2,j,k} = \frac{1}{2}(g^n_{i+1,j,k} + g^n_{i,j,k}),
\]

\[
g^n_{i-1/2,j,k} = \frac{1}{2}(g^n_{i-1,j,k} + g^n_{i,j,k}),
\]

(7b)

where the nodal \( g^n_{i,j,k} \) are computed using central finite differences \( D^c \) applied to \( g(\nabla u) \)

\[
g^n_{i,j,k} = g(\sqrt{(D^c_x v^n_{i,j,k})^2 + (D^c_y v^n_{i,j,k})^2 + (D^c_z v^n_{i,j,k})^2}).
\]

(7c)

As in (3), after advancing the solution in time using the above scheme, we perform a re-extension of the surface values

\[
v^{n+1}_{i,j,k} = \tilde{v}^{n+1}(c\langle x_{i,j,k} \rangle),
\]

(8)

where \( \tilde{v}^{n+1}(c\langle x_{i,j,k} \rangle) \) is approximated with tri-cubic interpolation on the surrounding grid points (see Fig. 2c).

The numerical scheme (7) uses a stencil shown in Fig. 2b. The extension (8) uses the tri-cubic interpolation stencil shown in Fig. 2c.

To determine the minimal computational band \( \Sect{2.6} \), we must be able to apply former stencil at each point in the latter \([15]\). In 3D, this results in a band with a width of \( 4.9h \). We note that the computational band scales with the area of the surface and not with the volume of the embedding space. In practical computations, we find this band contains less than 10% of the (theoretical) voxels of the bounding box of the surface (and the difference is even more significant in highly-resolved calculations). For a chosen number of time-steps, the running time scales linearly with the number of voxels in the computational band.

4. RESULTS

The results of our algorithm are demonstrated in Fig. 1, which shows denoising on the surface of a triangulated fish \([24]\). Gaussian diffusion blurs the image, while Perona–Malik diffusion removes noise while preserving sharp edges. Our implementation uses MATLAB.

Parameters used were \( h = 0.002, \lambda = 5 \), and 100 time steps of size \( \tau = 0.15h^2 = 6 \times 10^{-7} \). The fish is about 1 unit long. The computational band consists of 2 837 600 voxels surrounding the fish. Each of the 100 time-steps takes about 1 second on an Intel Core i7 CPU running at 3.20 GHz.

Our surface images have a range of \([0, 1]\). In our experiments, we apply additive noise in the embedding volume, normally distributed with amplitude 0.3, mean 0 and standard deviation 1. However, Perona–Malik is not limited to this noise model.

Fig. 3 shows an example of computing on the surface of a globe. In Fig. 3c we show that our approach results in a uniform application of denoising over the surface. We also demonstrate that if one first maps the surface image data onto a plane (easy enough here with a sphere but quite hard in general), and naively applies the standard
In this paper, we describe a numerical realization of nonlinear diffusion filters for images on general surfaces. We obtain our filters by combining the in-surface PDE models of Gaussian or Perona–Malik edge-preserving diffusion with the Closest Point Method. The resulting filter is simple: it alternates a time step of the corresponding diffusion PDE in a 3D volume surrounding the surface with a re-extension step.

In this approach we use the closest point function to represent the surface. Notably, only the re-extension step evaluates the closest point function, and the PDE need not be transformed as in other approaches to surface PDEs. The fully discrete algorithm uses standard finite difference techniques on a voxel grid to approximate the diffusion operators and tri-cubic interpolation to do the re-extension. Our experiments demonstrate that the filter works well and that we can handle complex geometries.

Here we have considered Gaussian and Perona–Malik diffusion on surfaces but the Closest Point Method is not limited to these two models. Indeed, the Closest Point Method applies generally to surface PDEs and could be useful in other surface image processing tasks. Applications other than denoising include deblurring and inpainting and are part of ongoing research.

6. REFERENCES


