A Framework for Solving Surface Partial Differential Equations for Computer Graphics Applications

Marcelo Bertalmio and Guillermo Sapiro*
Electrical and Computer Engineering, University of Minnesota

Li-Tien Cheng and Stanley Osher
UCLA Mathematics Department

Abstract

A novel framework for solving variational problems and partial differential equations for scalar and vector-valued data defined on surfaces is introduced in this paper. The key idea is to implicitly represent the surface as the level set of a higher dimensional function, and solve the surface equations in a fixed Cartesian coordinate system using this new embedding function. This thereby eliminates the need for performing complicated and not-accurate computations on triangulated surfaces, as it is commonly done in the graphics and numerical analysis literature. We describe the framework and present examples in texture synthesis, flow field visualization, as well as image and vector field regularization for data defined on 3D surfaces.

CR Categories: I.3.6 [Computer Graphics]: Methodology and Techniques—; G.1.8 [Numerical Analysis]: Partial Differential Equations—; I.4.10 [Image Processing and Computer Vision]: Image representation—;

Keywords: Partial differential equations, implicit surfaces, pattern formation, natural phenomena, flow visualization, image and vector field regularization.

1 Introduction

In a number of computer graphics applications, variational problems and partial differential equations (PDE's) need to be solved for data defined on arbitrary manifolds, three dimensional surfaces in particular. Examples of this are texture synthesis [37, 41], vector field visualization [10], and weathering [11]. In addition, data defined on surfaces often needs to be regularized, e.g., as part of a vector field computation or interpolation process [28, 38], for inverse problems [15], or for surface parameterization [12]. These last examples can be addressed by solving a variational problem on the surface, or its corresponding gradient-descent flow on the surface, using for example the theory of harmonic maps [14], which has recently been demonstrated to be of use for computer graphics applications as well, e.g., [12, 32, 43]. All these equations are generally solved on triangulated or polygonal surfaces. That is, the surface is given in polygonal (triangulated) form, and the data is discretely defined on it. This involves the non-trivial discretization of the equations in general polygonal grids, as well as the difficult numerical computation of other quantities like projections onto the discretized surface (when computing gradients and Laplacians for example). Although the use of triangulated surfaces is extremely popular in computer graphics, there is still *not* a widely accepted technique to compute differential characteristics such as tangents, normals, principal directions, and curvatures; see for example [9, 23, 34] for a few of the approaches in this direction. On the other hand, it is widely accepted that computing these objects for iso-surfaces (implicit representations) is straightforward and much more accurate and robust. This problem in triangulated surfaces becomes even bigger when we not only have to compute these first and second order differential characteristics of the surface, but also have to use them to solve variational problems and PDE's for data defined on the surface. Moreover, virtually no analysis exists on numerical PDE's on non-uniform grids in the generality needed for computer graphics, making it difficult to understand the behavior of the numerical implementation and its proximity (or lack thereof) to the continuous model.

In this paper we present a new framework to solve variational problems and PDE's for scalar and vector-valued data defined on surfaces. We use, instead of a triangulated/polygonal representation, an implicit representation: our surface will be the zero-level set of a higher dimensional *embedding* function (i.e., a 3D volume with real values, positive outside the surface and negative inside it). Implicit surfaces have been widely used in computer graphics, e.g., [3, 16, 40], as an alternative representation to triangulated surfaces. We smoothly extend the original (scalar or vector-valued) data lying on the surface to the 3D volume, adapt our PDE's accordingly, and then perform all the computations on the Cartesian grid corresponding to the embedding function. These computations are nevertheless intrinsic to the surface. The advantages of using the Cartesian grid instead of a triangulated mesh are many: we can use well studied numerical techniques, with accurate error measures; the topology of the underlying surface it is not an issue; and we can derive simple, accurate, robust and elegant implementations. If the original surface is not already in implicit form, and it is for example triangulated, we can use any of a number of implicitation algorithms that achieve this representation given a triangulated input, e.g., [13, 20, 33, 42]. For example, the public domain software [21] can be used. If the data is just defined on the surface, an extension of it to the whole volume is also easily achieved using a PDE, as we will see. Therefore, the method here proposed works as well for non-implicit surfaces after the preprocessing is performed. This preprocessing is quite simple and no complicated regridding needs to be done to go from one surface representation to another (see below). Finally, we will solve the variational problem or PDE only in a band surrounding the zero level set (a classical approach; see [26]). Therefore, although we will be increasing by one the dimension of the space, the computations remain of the same complexity, while the accuracy and simplicity are significantly improved.

1.1 Our contribution

Representing *deforming* surfaces as level sets of higher dimensional functions was introduced in [24] as a very efficient technique for numerically studying the deformation (see also [40] for studies on the deformation and manipulation of implicit surfaces for graphics applications). The idea is to represent the surface deformation via the embedding function deformation, which adds accuracy, robustness, and, as expected, topological liberty. When the velocity of the deformation is given by the minimization of an energy, the authors in [44] proposed a "variational level set" method, where

*Electrical and Computer Engineering, University of Minnesota, Minneapolis, MN 55455, USA, {marcelo,guille}@ece.umn.edu

they extended the energy (originally defined only on the surface) to the whole space. This allows for the implementation to be in the Cartesian grid. The key of this approach is to go from a "surface energy" to a "volume energy" by using a Dirac's delta function that concentrates the penalization on the given surface.

We will follow this general direction with our fixed, *non deforming* surfaces. In our case, what is being "deformed" is the (scalar or vector-valued) data on the surface. If this deformation is given by an energy-minimization problem (as is the case in data smoothing applications), we will extend the definition of the energy to the whole 3D space, and its minimization will be achieved with a PDE, which despite its being intrinsic to the underlying surface, it is also defined in the whole space. Therefore, it is easily implementable. This is straightforward, as opposed to approaches where one maps the surface data onto the plane, performs the required operations there and then maps the results back onto the triangulated representation of the surface; or approaches that attempt to solve the problem directly on a polygonal surface.

Very interestingly, the new framework proposed here also tells us how to translate into surface terms PDE's that we know that work on the plane but which do not necessarily minimize an energy (e.g., texture synthesis or flow visualization PDE's). Instead of running these PDE's on the plane an then mapping the results onto a triangulated representation of the surface, or running them directly on the triangulated domain, we obtain a 3D straightforward Cartesian grid realization that implements the equation intrinsically on the surface and whose accuracy depends only on the degree of spatial resolution.

Moreover, we consider that for computing differential characteristics and solving PDE's even for triangulated surfaces, it might be appropriate to run an implicitation algorithm as any of the ones used for the examples in this paper and then work on the implicit representation. Current algorithms for doing this, some of them publically available [21], are extremely accurate and efficient.

The contribution of this paper is then a new technique to efficiently solve a common problem in many computer graphics applications: the implementation of PDE's on 3D surfaces. In particular, we show how to transform any intrinsic variational or PDE equation into its corresponding one for implicit surfaces. In this paper we are then proposing a new framework to better solve existent problems and to help in building up the solutions for new ones. To exemplify the technique and its generality, we implement and extend popular equations previously reported in the literature. Here we solve them with our framework, while in the literature were solved with elaborated discretizations on triangulated representations.

2 The general framework

As mentioned before, our approach requires us to have an implicit representation of the given fixed surface, and the data must be defined in a band surrounding it and not just on the surface. The implicit surfaces used in this paper have been derived from public-domain triangulated surfaces via the computation of a (signed) distance function $\psi(x, y, z)$ to the surface $\mathcal{S}$. Arriving at an implicit representation from a triangulated one is not an issue, there are publicly available algorithms that achieve it in a very efficient fashion. To exemplify this, in our paper we have used several of these techniques. For some surfaces the classical Hamilton-Jacobi equation $\parallel \nabla\psi \parallel = 1$ was solved on a pre-defined grid enclosing the given surface via the computationally optimal approach devised in [35]. Accurate implicit surfaces from triangulations of the order of one million triangles are obtained in less than two minutes of CPU-time with this technique. Alternatively we used the implementation of the Closest Point Transform available in [21]. The teapot and knot surfaces were obtained from unorganized data points using the technique devised in [45]. We therefore assume from now on that the three dimensional surface $\mathcal{S}$ of interest is given in implicit form, as the zero level set of a given function $\psi : I\!R^3 \rightarrow I\!R$. This function is negative inside the closed bounded region defined by $\mathcal{S}$, positive outside, Lipschitz continuous a.e., with $\mathcal{S} \equiv \{x \in I\!R^3 : \psi(x) = 0\}$. To ensure that the data, which needs not to be defined outside of the surface originally, is now defined in the whole band, one simple possibility is to extend this data u defined on $\mathcal{S}$ (i.e the zero level set of ψ) in such a form that it is constant normal to each level set of ψ. This means the extension satisfies $\nabla u \cdot \nabla\psi = 0$. (For simplicity, we assume now u to be a scalar function, although we will also address in this paper problems where the data defined on $\mathcal{S}$ is vector-valued. This is solved in an analogous fashion.) To solve this we numerically search for the steady state solution of the Cartesian PDE

$$\frac{\partial u}{\partial t} + \text{sign}(\psi)(\nabla u \cdot \nabla\psi) = 0.$$

This technique was first proposed and used in [7]. Note that this keeps the given data u on the zero level set of ψ (the given surface) unchanged.

Both the implicitation and data extension (if required at all by the given data), need to be done only once off line. Moreover, they will remain for all applications that need this type of data.

We will exemplify our framework with the simplest case, the heat flow or Laplace equation for scalar data defined on a surface. For scalar data u defined on the plane, that is, $u(x, y) : I\!R^2 \rightarrow I\!R$ it is well known that the heat flow

$$\frac{\partial u}{\partial t} = \Delta u \tag{1}$$

where $\Delta := \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ is the Laplacian, is the gradient descent flow of the Dirichlet integral

$$\frac{1}{2}\int_{I\!R^2} \parallel \nabla u \parallel^2 dxdy, \tag{2}$$

where ∇ is the gradient.

Eq. (1) performs smoothing of the scalar data u, and this smoothing process progressively decreases the energy defined in eq. (2). If we now want to smooth scalar data u defined on a surface $\mathcal{S}$, we must find the minimizer of the energy given by

$$\frac{1}{2}\int_{\mathcal{S}} \parallel \nabla_{\mathcal{S}} u \parallel^2 d\mathcal{S}, \tag{3}$$

The equation that minimizes this energy is its gradient descent flow:

$$\frac{\partial u}{\partial t} = \Delta_{\mathcal{S}} u. \tag{4}$$

Here $\nabla_{\mathcal{S}}$ is the intrinsic gradient and $\Delta_{\mathcal{S}}$ the intrinsic Laplacian or Laplace-Beltrami operator. These are classical concepts in differential geometry, and basically mean the natural extensions of the gradient and Laplacian respectively, considering all derivatives intrinsic to the surface. For instance, the intrinsic gradient is just the projection onto $\mathcal{S}$ of the regular 3D gradient.

Classically, both in the computer graphics and numerical analysis communities, eq. (4) would be implemented in a triangulated surface, giving place to sophisticated and elaborated algorithms even for such simple flows. We now show how to simplify this when considering implicit representations.

Recall that $\mathcal{S}$ is given as the zero level set of a function $\psi : I\!R^3 \rightarrow I\!R$, ψ is negative inside the region bounded by $\mathcal{S}$, positive outside with $\mathcal{S} \equiv \{x \in I\!R^3 : \psi(x) = 0\}$. We proceed now to redefine the above energy and compute its corresponding gradient descent flow. Let $\vec{v}$ be a generic three dimensional vector, and $P_{\vec{v}}$

the operator that projects a given three dimensional vector onto the plane orthogonal to $\vec{v}$:

$$P_{\vec{v}} := I - \frac{\vec{v} \otimes \vec{v}}{\| \vec{v} \|^2} \tag{5}$$

It is then easy to show that the harmonic energy (3) is equivalent to (see for example [30])

$$\frac{1}{2} \int_{\mathcal{S}} \| P_{\vec{N}} \nabla u \|^2 \, d\mathcal{S}, \tag{6}$$

where $\vec{N}$ is the normal to the surface $\mathcal{S}$. In other words, $\nabla_{\mathcal{S}} u = P_{\vec{N}} \nabla u$. That is, the gradient intrinsic to the surface ($\nabla_{\mathcal{S}}$) is just the projection onto the surface of the 3D Cartesian (classical) gradient ∇. We now embed this in the function ψ:

$$\begin{aligned} & \frac{1}{2} \int_{\mathcal{S}} \| \nabla_{\mathcal{S}} u \|^2 \, d\mathcal{S} = \frac{1}{2} \int_{\mathcal{S}} \| P_{\vec{N}} \nabla u \|^2 \, d\mathcal{S} \\ = & \frac{1}{2} \int_{\Omega \in \mathbb{R}^3} \| P_{\nabla \psi} \nabla u \|^2 \, \delta(\psi) \| \nabla \psi \| \, dx, \end{aligned}$$

where $\delta(\cdot)$ stands for the delta of Dirac, and all the expressions above are considered in the sense of distributions. Note that first we got rid of intrinsic derivatives by replacing $\nabla_{\mathcal{S}}$ by $P_{\vec{N}} \nabla u$ (or $P_{\nabla \psi} \nabla u$) and then replaced the intrinsic integration ($\int_{\mathcal{S}} d\mathcal{S}$) by the explicit one ($\int_{\Omega \in \mathbb{R}^3} dx$) using the delta function. Intuitively, although the energy lives in the full space, the delta function forces the penalty to be effective only on the level set of interest. The last equality includes the embedding, and it is based on the following simple facts:

1. $\nabla \psi \parallel \vec{N}$.
2. $\int_{\Omega} \delta(\psi) \| \nabla \psi \| \, dx = \int_{\mathcal{S}} d\mathcal{S} =$ surface area.

In Appendix A we show that the gradient descent of this energy is given by

$$\frac{\partial u}{\partial t} = \frac{1}{\| \nabla \psi \|} \nabla \cdot (P_{\nabla \psi} \nabla u \| \nabla \psi \|). \tag{7}$$

In other words, this equation corresponds to the intrinsic heat flow for data on an implicit surface. But all the gradients in this PDE are defined in the three dimensional Cartesian space, not in the surface $\mathcal{S}$ (this is why we need the data to be defined at least on a band around the surface). The numerical implementation is then straightforward. This is the beauty of the approach! Basically, for this equation we use a classical scheme of forward differences in time and a succession of forward and backward differences in space (see Appendix B for details). The other equations in this paper are similarly implemented. This follows techniques as those in [29]. Once again, due to the implicit representation, classic numerics are used, avoiding elaborate projections onto discrete surfaces and discretization on general meshes, e.g., [9, 17].

It is easy to show a number of important properties of this equation:

1. For any second embedding function $\phi = \phi(\psi)$, with $\phi' \neq 0$, we obtain the same gradient descent flow. Since both ψ and ϕ have to share the zero level set, and we are only interested in the flow around this zero level set, this means that the flow is (locally) independent of the embedding function.[1]
2. If ψ is the signed distance function, a very popular implicit representation of surfaces (obtained for example from the implicitation algorithms previously mentioned), the gradient descent simplifies to

$$\frac{\partial u}{\partial t} = \nabla \cdot (P_{\nabla \psi} \nabla u). \tag{8}$$

We note that we could also have derived eq. (7) directly from the harmonic maps flow

$$\frac{\partial u}{\partial t} = \Delta_{\mathcal{S}} u,$$

via the simple geometry exercise of computing $\Delta_{\mathcal{S}} u$ for $\mathcal{S}$ in implicit form. This last property is of particular significance. It basically shows how to solve general PDE's, not necessarily gradient-descent flows, for data defined on implicit surfaces. All that we need to do is to recompute the components of the PDE for implicit representations of the surface. Note that in this way, conceptually, we can re-define classical planar PDE's on implicit surfaces, making them both intrinsic to the underlying surface and defined on the whole space.

From this very simple example we have seen the key point of our approach. If the process that we want to implement comes from the minimization of an energy, we derive a PDE for the whole space by computing the gradient-descent of the whole-space-extension of that energy. Otherwise, given a planar PDE we recompute its components for an implicit representation of the surface. For instance, anisotropic diffusion can be performed on the plane by

$$\frac{\partial u}{\partial t} = \nabla \cdot \left(\frac{\nabla u}{\| \nabla u \|} \right), \tag{9}$$

which minimizes the energy $\frac{1}{2} \int_{\mathbb{R}^2} \| \nabla u \| \, dxdy$ (see [29]). If we now want to perform anisotropic diffusion of scalar data on a surface $\mathcal{S}$, we can either recompute the gradient-descent flow for an extension of the energy or just substitute in eq. (9) the corresponding expressions. Either way, we obtain the same result, the following PDE, which is valid in the Euclidean space:

$$\frac{\partial u}{\partial t} = \frac{1}{\| \nabla \psi \|} \nabla \cdot \left(\frac{P_{\nabla \psi} \nabla u}{\| P_{\nabla \psi} \nabla u \|} \| \nabla \psi \| \right). \tag{10}$$

In the following section more equations will be presented.

3 Experimental examples

We now exemplify the framework just introduced for a number of important cases. The numerical implementation used is quite simple, and requires a few lines of *C++* code. The CPU time required for the diffusion examples is of a few seconds on a PC (512Mb RAM, 1GHz) under Linux. For the texture synthesis examples, the CPU time ranges from a few minutes to one hour, depending on the pattern and parameters chosen. All the volumes used contain roughly 128^3 voxels. Note once again that due to the use of only a narrow band surrounding the zero level set, the order of the algorithmic complexity remains the same. On the other hand, the use of straightforward Cartesian numerics reduces the overall algorithmic complexity, improving accuracy and simplifying the implementation.

3.1 Diffusion of scalar images on surfaces

The use of PDE's for image enhancement has become one of the most active research areas in image processing [6]. In particular, diffusion equations are commonly used for image regularization,

[1] We thank F. Mémoli for helping with this fact.

denoising, and multiscale representations (representing the image simultaneously at several scales or levels of resolution). This started with the works in [19, 39], where the authors suggested the use of the linear heat flow (1) for this task, where u represents the image gray values (the original image is used as initial condition). As we have seen, this is the gradient-descent of (2), and the generalizations of these equations for data on the surface are given by (4) and (3) respectively. In implicit form, the heat flow on surfaces is given by (7). Figure 1 shows a simple example of image diffusion on a surface. Please note that this is *not* equivalent to performing 3D smoothing of the data and then looking to see what happened on $\mathcal{S}$. Our flow, though using extended 3D data, performs smoothing *directly* on the surface, it is an intrinsic heat flow. The complete details of the numerical implementation of this flow are given in Appendix B (this will once again show how the implementation is significantly simplified with the framework here described).

We should note before proceeding that [18] also showed how to regularize images defined on a surface. The author's approach is limited to graphs (not generic surfaces) and only applies to level set based motions. The approach is simply to project the deformation of the data on the surface onto a deformation on the plane.

3.2 Diffusion of directional data on surfaces

A particularly interesting example is obtained when we have unit vectors defined on the surface. That is, we have data of the form $u : \mathcal{S} \rightarrow S^{n-1}$. When $n = 3$ our unit vectors lie on the sphere. Particular examples of this are principal directions (or general directional fields on 3D surfaces) and chromaticity vectors (normalized RGB vectors). This is also one of the most studied cases of the theory of harmonic maps due to its physical relationship with liquid crystals, and it was introduced in [32] for the regularization of directional data, unit vectors, on the plane. This framework of harmonic maps was used in computer graphics for texture mapping and surface parameterization, as pointed out earlier.

We still want to minimize an energy of the form

$$\int_{\mathcal{S}} \parallel \nabla_{\mathcal{S}} u \parallel^p d\mathcal{S},$$

though in this case $\nabla_{\mathcal{S}}$ is the vectorial gradient and the minimizer is restricted to be a unit vector. It is easy to show, e.g., [4, 31], that the gradient descent of this energy is given by the coupled system of PDE's

$$\frac{\partial u_i}{\partial t} = \mathrm{div}_{\mathcal{S}} \left(\parallel \nabla_{\mathcal{S}} u \parallel^{p-2} \nabla_{\mathcal{S}} u_i \right) + u_i \parallel \nabla_{\mathcal{S}} u \parallel^p, \;\; 1 \leq i \leq n.$$

This flow guarantees that the initial unit vector $u(x, y, z, 0)$ remains a unit vector $u(x, y, z, t)$ all the time, thereby providing an equation for isotropic ($p = 2$) and anisotropic ($p = 1$) diffusion and regularization of unit vectors on a surface.

We can now proceed as before, and embed the surface $\mathcal{S}$ into the zero level-set of ψ, obtaining the following gradient descent flows:

$$\frac{\partial u_i}{\partial t} = \frac{1}{\parallel \nabla \psi \parallel} \nabla \cdot \left(\frac{P_{\nabla \psi} \nabla u_i}{\parallel P_{\nabla \psi} \nabla u \parallel^{2-p}} \parallel \nabla \psi \parallel \right) + u_i \parallel P_{\nabla \psi} \nabla u \parallel^p . \tag{11}$$

Note once again that although the regularization is done intrinsically on the surface, this equation only contains Cartesian gradients. An example of this flow for anisotropic diffusion of principal direction vectors is given in Figure 2. On the left, we see the surface of a bunny with its correspondent vector field for the major principal direction. Any irregularity on the surface produces a noticeable alteration of this field, as can be seen in the details *a* and *b*. In the details *a'* and *b'*, we see the result of applying the flow (11). Once again, the implementation of this flow with our framework is straightforward, while it would require very sophisticated techniques on triangulated surfaces (techniques that, in addition, are not supported by theoretical results).

Following also the work [32] for color images defined on the plane, we show in Figure 3 how to denoise a color image painted on an implicit surface. The basic idea is to normalize the RGB vector (a three dimensional vector) to a unit vector representing the chroma, and diffuse this unit vector with the harmonic maps flow (11).[2] The corresponding magnitude, representing the brightness, is smoothed separately via scalar diffusion flows as those presented before (e.g., the intrinsic heat flow or the intrinsic anisotropic heat flow). That is, we have to regularize a map onto S^2 (the chroma) and another one onto $\mathbb{R}$ (the brightness).

3.3 Pattern formation on surfaces via reaction-diffusion flows

The use of reaction-diffusion equations for texture synthesis became very popular in computer graphics following the works of Turk [37] and Witkin and Kass [41]. These works follow original ideas by Turing [36], who showed how reaction diffusion equations can be used to generate patterns. The basic idea in these models is to have a number of "chemicals" that diffuse at different rates and that react with each other. The pattern is then synthesized by assigning a brightness value to the concentration of one of the chemicals. The authors in [37, 41] used their equations for planar textures and textures on triangulated surfaces. By using the framework here described, we can simply create textures on (implicit/implicitized) surfaces, without the elaborated schemes developed in those papers.[3]

Assuming a simple isotropic model with just two chemicals u_1 and u_2, we have

$$\frac{\partial u_1}{\partial t} = F(u_1, u_2) + D_1 \Delta u_1,$$

$$\frac{\partial u_2}{\partial t} = G(u_1, u_2) + D_2 \Delta u_1,$$

where D_1 and D_2 are two constants representing the diffusion rates and F and G are the functions that model the reaction.

Introducing our framework, if u_1 and u_2 are defined on a surface $\mathcal{S}$ implicitly represented as the zero level set of ψ we have

$$\frac{\partial u_1}{\partial t} = F(u_1, u_2) + D_1 \frac{1}{\parallel \nabla \psi \parallel} \nabla \cdot (P_{\nabla \psi} \nabla u_1 \parallel \nabla \psi \parallel), \tag{12}$$

$$\frac{\partial u_2}{\partial t} = G(u_1, u_2) + D_2 \frac{1}{\parallel \nabla \psi \parallel} \nabla \cdot (P_{\nabla \psi} \nabla u_2 \parallel \nabla \psi \parallel). \tag{13}$$

For simple isotropic patterns, Turk [37] selected

$$F(u_1, u_2) = s(16 - u_1 u_2),$$

$$G(u_1, u_2) = s(u_1 u_2 - u_2 - \beta),$$

where s is a constant and β is a random function representing irregularities in the chemical concentration. Examples of this, for implicit surfaces, are given in Figure 4 (the coupled PDE's shown above are run until steady state is achieved). To simulate anisotropic textures, instead of using additional chemicals as in [37], we use

[2] We re-normalize at every discrete step of the numerical evolution to address deviations from the unit norm due to numerical errors [8]. We could also extend the framework in [1] and apply it to our equations.

[3] Note that this is not the scheme proposed in [25], where the texture is created in the full 3D space. Here, the texture is created via reaction-diffusion flows intrinsic to the surface, just the implementation is on the embedding 3D space.

anisotropic diffusion, as suggested in [41]. For this purpose, we replace eq. (12) with:

$$\frac{\partial u_1}{\partial t} = F(u_1, u_2) + D_1 \frac{1}{\parallel \nabla\psi \parallel} \nabla \cdot ((\overrightarrow{d} \cdot P_{\nabla\psi} \nabla u_1)\overrightarrow{d} \parallel \nabla\psi \parallel), \tag{14}$$

where $\overrightarrow{d}$ is a vector field tangent to the surface, e.g., the field of the major principal direction (which for our examples has been also accurately computed directly on the implicit surface, using the technique proposed in [22]). Note how this particular selection of the anisotropic reaction-diffusion flow direction provides a texture that helps on the shape perception of the object. Additional patterns can be obtained with different combinations of the reaction and diffusion parts of the flow.

3.4 Flow visualization on 3D surfaces

Inspired by the work on line integral convolution [5] and that on anisotropic diffusion [27], the authors of [10] suggested to use anisotropic diffusion to visualize flows in 2D and 3D. The basic idea is, starting from a random image, anisotropically diffuse it in the directions dictated by the flow field. The authors presented very nice results both in 2D (flows on the plane) and 3D (flows on a surface), but once again using triangulated surfaces which introduce many computational difficulties. In a straightforward fashion we can compute these anisotropic diffusion equations on the implicit surfaces with the framework here introduced, and some results are presented in Figure 5. Note the complicated topology and how both the inside and outside parts of the surfaces are easily handled with our implicit approach. Also note that, when we choose the vector field to be that of one of the principal directions, the result emphasizes the surface shape.

4 Concluding remarks

In this paper, we have introduced a novel framework for solving variational problems and PDE's for data defined on surfaces. The technique borrows ideas from the level set theory and the theory of harmonic maps. The surface is embedded in a higher dimensional function, and the Euler-Lagrange flow or PDE is solved in the Cartesian coordinate system of this embedding function. The equations are intrinsic to the implicit surface, following the general formulations in harmonic map theory. With this framework we enjoy accuracy, robustness, and simplicity, as expected from the computation of differential characteristics on iso-surfaces (implicit surfaces). In addition to presenting the general approach, we have exemplified it with equations arising in image processing and computer graphics. We are currently investigating other applications of this framework, e.g, image inpainting on surfaces [2], inverse problems as those in [15], and texture mapping for implicit surfaces following [12].

Acknowledgment

We thank Facundo Mémoli for interesting conversations during this work. The implicit data provided by S. Betelu and H. Zhao. This work was partially supported by grants from the Office of Naval Research ONR-N00014-97-1-0509 and ONR-N00014-97-1-0027, the Office of Naval Research Young Investigator Award, the Presidential Early Career Awards for Scientists and Engineers (PECASE), a National Science Foundation CAREER Award, the National Science Foundation Learning and Intelligent Systems Program (LIS), NSF-DMS-9706827, ARO-DAAG-55-98-1-0323 and IIE-Uruguay.

Appendix A: Heat flow on implicit surfaces

Considering

$$E(u) := \frac{1}{2} \int_{\Omega \in I\!R^3} \parallel P_{\nabla\psi} \nabla u \parallel^2 \delta(\psi) \parallel \nabla\psi \parallel dx,$$

and μ a perturbation of u,

$$\begin{aligned}
\tfrac{d}{dt} \mid_{t=0} E(u + t\mu) &= \int_\Omega (P_{\nabla\psi}\nabla u \cdot P_{\nabla\psi}\nabla\mu)\delta(\psi) \parallel \nabla\psi \parallel dx \\
&= \int_\Omega \left(P_{\nabla\psi}\nabla u \cdot \left(\nabla\mu - \frac{\nabla\psi \cdot \nabla\mu}{\parallel \nabla\psi \parallel^2} \nabla\psi \right) \right) \delta(\psi) \parallel \nabla\psi \parallel dx \\
&= \int_\Omega (P_{\nabla\psi}\nabla u \cdot \nabla\mu)\delta(\psi) \parallel \nabla\psi \parallel dx \\
&\quad - \int_\Omega (P_{\nabla\psi}\nabla u \cdot \nabla\psi) \frac{\nabla\psi \cdot \nabla\mu}{\parallel \nabla\psi \parallel^2} \delta(\psi) \parallel \nabla\psi \parallel dx \\
&= \int_\Omega (P_{\nabla\psi}\nabla u \cdot \nabla\mu)\delta(\psi) \parallel \nabla\psi \parallel dx \\
&= -\int_\Omega \nabla \cdot (P_{\nabla\psi}\nabla u \delta(\psi) \parallel \nabla\psi \parallel)\mu dx \\
&= -\int_\Omega \nabla \cdot (P_{\nabla\psi}\nabla u \parallel \nabla\psi \parallel)\delta(\psi)\mu dx \\
&\quad - \int_\Omega (P_{\nabla\psi}\nabla u \cdot \nabla\psi)\delta'(\psi) \parallel \nabla\psi \parallel \mu dx \\
&= -\int_\Omega \nabla \cdot (P_{\nabla\psi}\nabla u \parallel \nabla\psi \parallel)\delta(\psi)\mu dx \\
&= -\int_{\mathcal{S} \equiv \{\psi = 0\}} \frac{1}{\parallel \nabla\psi \parallel} \nabla \cdot (P_{\nabla\psi}\nabla u \parallel \nabla\psi \parallel)\mu d\mathcal{S}.
\end{aligned}$$

Since the above has to be zero for all μ, we conclude that at the zero level set of ψ,

$$\frac{1}{\parallel \nabla\psi \parallel} \nabla \cdot (P_{\nabla\psi}\nabla u \parallel \nabla\psi \parallel) = 0,$$

and we make a natural extension to the whole domain Ω by considering this to hold on it.[4] We then obtain that the gradient descent for the "implicit harmonic energy" is given by

$$\frac{\partial u}{\partial t} = \frac{1}{\parallel \nabla\psi \parallel} \nabla \cdot (P_{\nabla\psi}\nabla u \parallel \nabla\psi \parallel). \tag{15}$$

Appendix B: Numerical implementation of the heat flow on implicit surfaces

We now provide details on the numerical implementation of the intrinsic heat flow on implicit surfaces. All other equations reported in this paper are similarly implemented. Recall that all equations are on Cartesian grids, thereby permitting the use of classical numerics. We work on a cubic grid ($\Delta x = \Delta y = \Delta z = 1$), using an explicit scheme, where we compute the value of $u^n_{i,j,k} = u(i\Delta x, j\Delta y, k\Delta z, n\Delta t)$ based only in previous values ($t = (n-1)\Delta t$) of its neighbors, i.e., forward time differences.

[4] We have assumed that $\parallel \nabla\psi \parallel \neq 0$, at least on a band surrounding the zero level set. This assumption is valid since we can make the embedding function to be a distance function ($\parallel \nabla\psi \parallel = 1$), or simply multiply ψ by another function that guarantees that the zero-level set $\mathcal{S}$ is preserved and that the gradient of the new embedding function is not zero.

Firstly, we compute the 3D gradient of u using forward differences:

$\overrightarrow{v}^n_{i,j,k} = \nabla_+ u^n_{i,j,k} = (u^n_{i+1,j,k} - u^n_{i,j,k}, u^n_{i,j+1,k} - u^n_{i,j,k}, u^n_{i,j,k+1} - u^n_{i,j,k})$.

We compute the vector $\overrightarrow{N}(i,j,k)$, which gives the direction of the (outward) normal to the isosurface of ψ at the point (i,j,k):

$\overrightarrow{N}_{i,j,k} = \nabla\psi_{i,j,k} = \frac{1}{2}(\psi_{i+1,j,k} - \psi_{i-1,j,k}, \psi_{i,j+1,k} - \psi_{i,j-1,k}, \psi_{i,j,k+1} - \psi_{i,j,k-1})$.

Here we have used central differences. Note that, since ψ is fixed, $\overrightarrow{N}(i,j,k)$ does not change in time and we need only to compute it once. Its norm is: $\| \overrightarrow{N}_{i,j,k} \| = (\sum_{m=1}^{3} (N_{i,j,k}[m])^2)^{\frac{1}{2}}$. The square brackets denote the components of the vector. Then we compute the intrinsic gradient, i.e., we project ∇u onto the plane normal to $\overrightarrow{N}$:

$(P_{\overrightarrow{N}} \overrightarrow{v})^n_{i,j,k} = \overrightarrow{v}^n_{i,j,k} - (\frac{\sum_{m=1}^{3} N_{i,j,k}[m] \cdot v_{i,j,k}[m]}{\| \overrightarrow{N}_{i,j,k} \|^2}) \overrightarrow{N}_{i,j,k}$.

Finally, we use a backward-differences implementation of the divergence:

$\nabla_- \overrightarrow{w}_{i,j,k} = w_{i,j,k}[1] - w_{i-1,j,k}[1] + w_{i,j,k}[2] - w_{i,j-1,k}[2] + w_{i,j,k}[3] - w_{i,j,k-1}[3]$.

With forward time differences, the numerical implementation of the heat flow on implicit surfaces (eq. (7)) is then:

$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \Delta t (\frac{1}{\| \overrightarrow{N}_{i,j,k} \|} \nabla_- \cdot ((P_{\overrightarrow{N}} \overrightarrow{v})^n_{i,j,k} \| \overrightarrow{N}_{i,j,k} \|))$.

If the embedding function is a signed distance function, obtaining eq. (8), this expression for the numerical implementation of the intrinsic heat flow is simplified even further, making it virtually trivial.

References

[1] F. Alouges, "An energy decreasing algorithm for harmonic maps," in J.M. Coron *et al.*, Editors, *Nematics*, Nato ASI Series, Kluwer Academic Publishers, Netherlands, pp. 1-13, 1991.

[2] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," *Computer Graphics (SIGGRAPH)*, pp. 417-424, New Orleans, July 2000.

[3] J. Bloomenthal, *Introduction to Implicit Surfaces*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1997.

[4] H. Brezis, J. M. Coron, and E. H. Lieb, "Harmonic maps with defects," *Communications in Mathematical Physics* **107**, pp. 649-705, 1986.

[5] B. Cabral and C. Leedom. "Imaging vector fields using line integral convolution," *ACM Computer Graphics (SIGGRAPH '93)* **27:4**, pp. 263-272, 1993.

[6] V. Caselles, J. M. Morel, G. Sapiro, and A. Tannenbaum, Editors, "Special Issue on Partial Differential Equations and Geometry-Driven Diffusion in Image Processing and Analysis," *IEEE Trans. Image Processing* **7**, March 1998.

[7] S. Chen, B. Merriman, S. Osher, and P. Smereka, "A simple level set method for solving Stefan problems," *Journal of Computational Physics* **135**, pp. 8, 1995.

[8] R. Cohen, R. M. Hardt, D. Kinderlehrer, S. Y. Lin, and M. Luskin, "Minimum energy configurations for liquid crystals: Computational results," in J. L. Ericksen and D. Kinderlehrer, Editors, *Theory and Applications of Liquid Crystals*, pp. 99-121, IMA Volumes in Mathematics and its Applications, Springer-Verlag, New York, 1987.

[9] M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr, "Discrete differential-geometry operators in nD," *Multi-res modeling group TR*, Caltech, September 2000 (obtained from www.multires.caltech.edu).

[10] U. Diewald, T. Preufer, and M. Rumpf, "Anisotropic diffusion in vector field visualization on Euclidean domains and surfaces," *IEEE Trans. Visualization and Computer Graphics* **6**, pp. 139-149, 2000.

[11] J. Dorsey and P. Hanrahan, "Digital materials and virtual weathering," *Scientific American* **282:2**, pp. 46-53, 2000.

[12] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. "Multi-resolution analysis of arbitrary meshes," *Computer Graphics (SIGGRAPH '95 Proceedings)*, pp. 173-182, 1995.

[13] M. Eck and H. Hoppe, "Automatic reconstruction of B-spline surfaces of arbitrary topological type," *Computer Graphics*, 1996.

[14] J. Eells and L. Lemarie, "A report on harmonic maps," *Bull. London Math. Soc.* **10:1**, pp. 1-68, 1978.

[15] O. Faugeras, F. Clément, R. Deriche, R. Keriven, T. Papadopoulo, J. Gomes, G. Hermosillo, P. Kornprobst, D. Lingrad, J. Roberts, T. Viéville, F. Devernay, "The inverse EEG and MEG problems: The adjoint state approach I: The continuous case," *INRIA Research Report* **3673**, June 1999.

[16] S. F. Frisken, R. N. Perry, A. Rockwood, and T. Jones, "Adaptively sampled fields: A general representation of shape for computer graphics," *Computer Graphics (SIGGRAPH)*, New Orleans, July 2000.

[17] G. Huiskamp, "Difference formulas for the surface Laplacian on a triangulated surface," *Journal of Computational Physics* **95**, pp. 477-496, 1991.

[18] R. Kimmel, "Intrinsic scale space for images on surfaces: The geodesic curvature flow," *Graphical Models and Image Processing* **59**, pp. 365-372, 1997.

[19] J. J. Koenderink, "The structure of images," *Biological Cybernetics* **50**, pp. 363-370, 1984.

[20] V. Krishnamurthy and M. Levoy, "Fitting smooth surfaces to dense polygon meshes," *Computer Graphics*, pp. 313-324, 1996.

[21] S. Mauch, "Closest point transform," www.ama.caltech.edu/∼seanm/software/cpt/cpt.html.

[22] O. Monga, S. Benayoun and O. Faugeras, "From partial derivatives of 3D density images to ridge lines," *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 354-359, 1992.

[23] H. P. Moreton and C. H. Séquin, "Functional minimization or fair surface design," *Computer Graphics (SIGGRAPH)*, pp. 167-176, 1992.

[24] S. J. Osher and J. A. Sethian, "Fronts propagation with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulations," *Journal of Computational Physics* **79**, pp. 12-49, 1988.

[25] K. Perlin, "An image synthesizer," *Computer Graphics* **19**, pp. 287-296, 1985.

[26] D. Peng, B. Merriman, S. Osher, H. Zhao, M. Kang, "A PDE-based fast local level set method," *Journal of Computational Physics* **155**, pp. 410-438, 1999.

[27] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern. Anal. Machine Intell.* **12**, pp. 629-639, 1990.

[28] E. Praun, A. Finkelstein, and H. Hoppe, "Lapped textures," *ACM Computer Graphics (SIGGRAPH)*, New Orleans, July 2000.

[29] L. I. Rudin, S. Osher, and E. Fatemi, "Nonlinear total variation based noise removal algorithms," *Physica D* **60**, pp. 259-268, 1992.

[30] L. Simon, *Lectures on Geometric Measure Theory*, Australian National University, Australia, 1984.

[31] M. Struwe, "On the evolution of harmonic mappings of Riemannian surfaces," *Comment. Math. Helvetici* **60**, pp. 558-581, 1985.

[32] B. Tang, G. Sapiro, and V. Caselles, "Diffusion of general data on non-flat manifolds via harmonic maps theory: The direction diffusion case," *Int. Journal Computer Vision* **36:2**, pp. 149-161, February 2000.

[33] G. Taubin, "Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation," *IEEE Trans. PAMI* **13:11**, pp. 1115-1138, 1991.

Figure 1: Intrinsic isotropic diffusion. Left: original image. Middle: after 60 diffusion steps. Right: after 160 diffusion steps.

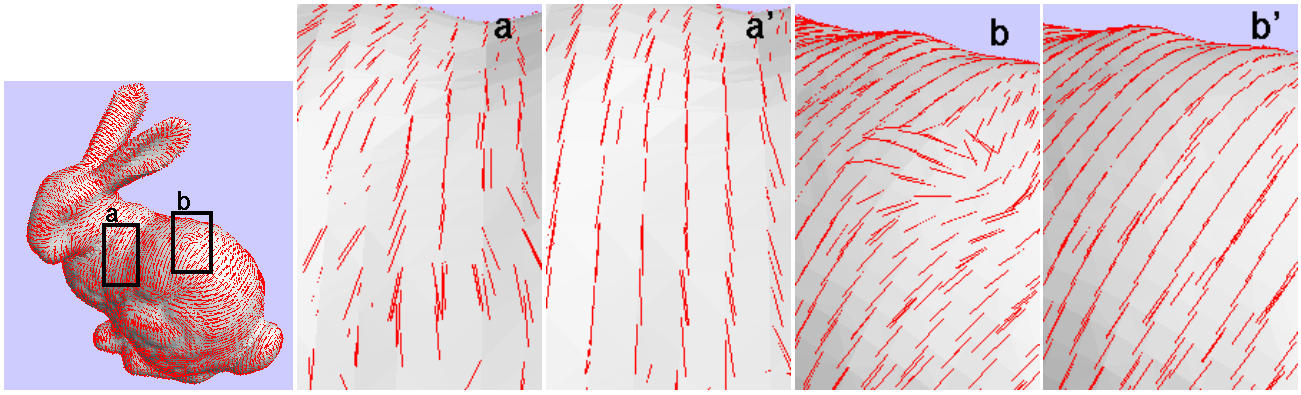

Figure 2: Intrinsic vector field regularization. Left: original field of major principal direction of the surface. Details *a* and *b*: original field. Details *a'* and *b'*: after anisotropic regularization.

[34] G. Taubin, "Estimating the tensor of curvature of a surface from a polyhedral approximation," *IEEE International Conference Computer Vision*, pp. 902-907, Boston, MA, 1995.

[35] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control* **40** pp. 1528-1538, 1995.

[36] A. Turing, "The chemical basis of morphogenesis," *Philosophical Transactions of the Royal Society B* **237**, pp. 37-72, 1952.

[37] G. Turk, "Generating textures on arbitrary surfaces using reaction-diffusion," *Computer Graphics* **25:4**, pp. 289-298, , July 1991.

[38] G. Winkenbach and D. H. Salesin, "Rendering parametric surfaces in pen and ink," *Computer Graphics (SIGGRAPH 96)*, pp. 469-476, 1996.

[39] A. P. Witkin, "Scale-space filtering," *Int. Joint. Conf. Artificial Intelligence* **2**, pp. 1019-1021, 1983.

[40] A. Witkin and P. Heckbert, "Using particles to sample and control implicit surfaces," *Computer Graphics (SIGGRAPH)*, pp. 269-278, 1994.

[41] A. Witkin and M. Kass, "Reaction-diffusion textures," *Computer Graphics (SIGGRAPH)* **25:4**, pp. 299-308, July 1991.

[42] G. Yngve and G. Turk, "Creating smooth implicit surfaces from polygonal meshes," *Technical Report GIT-GVU-99-42, Graphics, Visualization, and Usability Center. Georgia Institute of Technology*, 1999 (obtained from www.cc.gatech.edu/gvu/geometry/publications.html).

[43] D. Zhang and M. Hebert, "Harmonic maps and their applications in surface matching," *Proc. CVPR '99*, Colorado, June 1999.

[44] H. K. Zhao, T. Chan, B. Merriman, and S. Osher, "A variational level set approach to multi-phase motion," *J. of Computational Physics* **127**, pp. 179-195, 1996.

[45] H. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit, nonparametric shape reconstruction from unorganized points using a variational level set method," *Comp. Vision and Image Understanding* **80**, pp. 295-314, 2000.

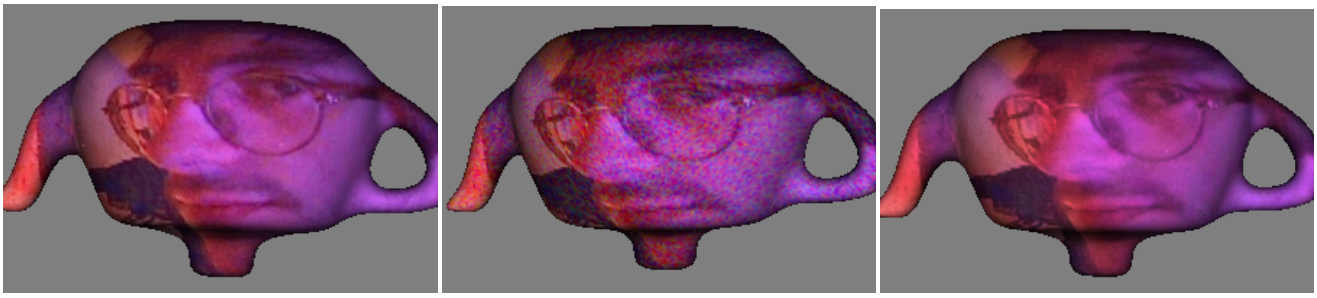

Figure 3: Intrinsic vector field regularization. Left: original color image. Middle: heavy noise has been added to the 3 color channels. Right: color image reconstructed after 20 steps of anisotropic diffusion of the chroma vectors.

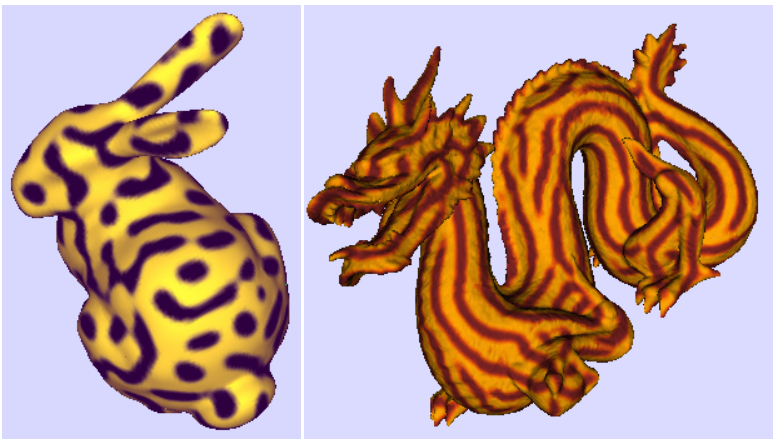

Figure 4: Texture synthesis via intrinsic reaction-diffusion flows on implicit surfaces. Left: isotropic. Right: anisotropic. Pseudo-color representation of scalar data is used. The numerical values used in the computations were $D_1 = 1.0, D_2 = 0.0625, s = 0.025, \beta = 12.0 \pm 0.1, u_1(0) = u_2(0) = 4.0$.

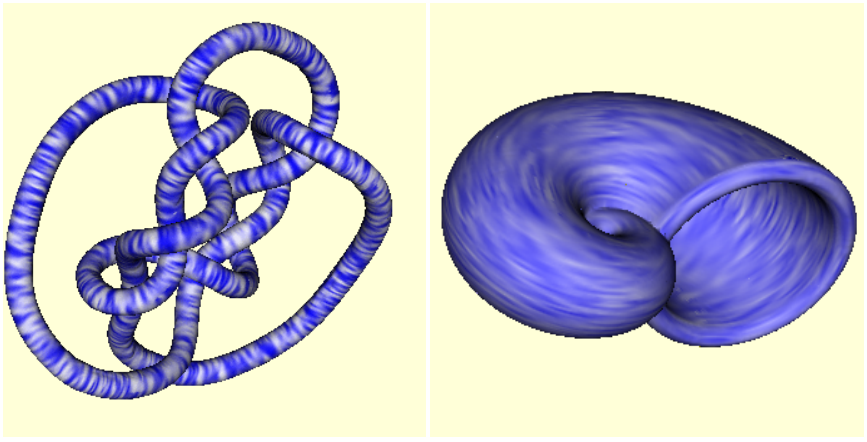

Figure 5: Flow visualization on implicit 3D surfaces via intrinsic anisotropic diffusion flows. Left: flow aligned with the major principal direction of the surface. Right: flow aligned with the minor principal direction of the surface. Pseudo-color representation of scalar data is used.