Partial Differential Equations for Interpolation and Compression of Surfaces

Egil Bae[1] and Joachim Weickert[2]

[1] Department of Mathematics, University of Bergen, Norway
(Egil.Bae@math.uib.no).
[2] Mathematical Image Analysis Group, Faculty of Mathematics and Computer Science Building E1.1, Saarland University, 66041 Saarbrücken, Germany
(weickert@mia.uni-saarland.de).

Abstract. Partial differential equations (PDEs) have recently shown to be very promising for image interpolation and compression. Inspired from this work, we present a PDE based approach to interpolation of surfaces from scattered point sets using the geometric diffusion equation. Triangulated surfaces are considered in the discrete setting, and the geometric diffusion equation is discretized by the finite element method directly on the triangular mesh. Furthermore, a PDE based method for lossy compression of triangulated surfaces is presented. The idea is to store only a few relevant vertex coordinates in the encoding step. In the decoding step, the remaining vertices are reconstructed by solving the geometric diffusion equation. Finally, two modified reconstruction methods are proposed that are shown to improve the compression quality for both images and surfaces. These reconstruction methods approximate instead of interpolating, and have links to Hopscotch methods for the numerical solution of PDEs. Experiments are presented illustrating that results of high quality can be obtained using simple geometric diffusion without any information on surface normals.

1 Introduction

Compression of mesh geometry is getting increasingly more important. Surfaces of millions, and even billions of vertices can get acquired. In order to handle such large amount of data, compression is necessary.

Until recently, the most competitive image and surface compression methods are based on decorrelating the data in the frequency domain, using methods such as wavelets or the cosine transform [1,2,3]. For image compression, the work of [4] proposed a very promosing alternative method, based on interpolation in the spatial domain with PDEs.

Inspired from this work, we propose a PDE based method for surface interpolation from sparse scattered point sets, and a PDE based method for lossy compression of triangulated surfaces. The interpolation method is formulated in the continuous setting as finding the steady state of a time dependent geometric PDE. In the discrete setting, triangulated surfaces are considered, and the

PDE is discretized by the finite element method, yielding a discrete flow for each vertex coordinate.

The compression method encodes the surface by storing only a suitable small subset of the vertices. The surface is reconstructed in the decoding step by interpolating from the scattered set of points by solving a geometric PDE.

In this work, we particularly study the simple geometric diffusion equation, and show that it has surprisingly good interpolation properties when the interpolation points are selected properly. Furthermore, an improvement is presented based on this equation, which links approximation and numerical methods for PDEs. A similar improvement is also presented for image compression, showing that the linear diffusion equation can yield equally good results as more complicated nonlinear PDEs.

The goal of this paper is to prove the concept of the new surface compression method. We aim to make comparisons with other methods at later stage of development. At the current stage it is hard to assess the exact compression rate of our method. Also, in contrary to wavelets which require semi-regular meshes, our method is applied directly to irregular meshes. This makes a direct comparison difficult.

The structure of the paper is as follows: In Section 2 we describe the continuous framework behind our surface interpolation. A finite element discretization is presented in Section 3. In the fourth section we give details on our baseline method for surface compression, while in Section 5 we introduce two modifications that avoid singularities at the interpolation points. Numerical experiments are performed in Section 6, and the paper is concluded with a summary in Section 7.

Relation to previous work. PDEs have been used for a growing number of image and surface processing problems in the recent years. They have shown to be very powerful for problems such as inpainting, where one wants to fill in regions of missing information using information at the boundary of the regions [5,6,7,8,9]. Interpolation is a special type of inpainting problem, where the data is only known on a scattered set of points. Until recently, PDE based methods have been little studied for this problem.

For surface interpolation, one method based on solving the Poisson problem was proposed in [10]. Contrary to this work, we don't need any information about the normal vectors. From a compression point of view this is an advantage, since normal vectors are expensive to store. For unoriented dense point clouds, a level set approach was proposed in [11], based on finding the surface minimizing a functional depending on the distance to the interpolation points. In [12,13], Kobbelt et. al. proposed a method for interpolating a fine polygon from the vertices of a coarse polygon by minimizing a function of linearized curvature.

For image interpolation, the recent work of [4] demonstrated that certain PDEs are very promising. Especially edge enhancing anisotropic diffusion [14], where an anisotropic diffusion tensor suppresses diffusion across edges and allows diffusion along them, performed well. Also the linear diffusion equation yielded good results, provided that the interpolation points were chosen carefully. The image was reconstructed by solving a Dirichlet boundary value problem with the interpolation data as boundary conditions. Furthermore, a new image compression method was developed based on these results, shown to outperform the well-known JPEG standard at high compression rates. The idea was to store only pixels that were hard to reconstruct by the PDE in the encoding step. In the decoding step, the image was interpolated from this sparse set of pixels. Selection of the best interpolation points is a difficult combinatorial optimization problem. For the linear diffusion equation, [15] showed that the exact solution could be computed. For other PDEs, suboptimal solutions could be estimated by B-tree triangular coding, or stochastic approaches [4].

We wish to extend the image interpolation ideas to surfaces. Locally, the coordinate maps of a surface are very similar to an image. We therefore formulate PDEs, such that the differential operators act directly on the coordinate maps. In this sense, the geometric diffusion equation can be seen as the surface equivalent of the linear diffusion equation.

The geometric diffusion equation and the Laplace-Beltrami operator have been extensively used in geometric modelling. Typically for the purpose of surface fairing or smoothing [16,17,18,19,20]. In [21], a mesh encoding technique is developed based on quantization of the discrete Laplace-Beltrami operator applied to the mesh.

2 PDE based surface interpolation

In this section, we present the continuous formulation for our PDE based method for surface interpolation from sparse scattered point sets.

We begin with some basic notation. Let S denote a smooth, compact, orientable surface embedded in $\mathbb{R}^3$. S can be parameterized by $\{x^\alpha, \Omega^\alpha\}_\alpha$, where $x^\alpha(\xi_1^\alpha, \xi_2^\alpha) : \Omega_\alpha \mapsto S$ are the coordinate maps (patches). For ease of notation, we from now on drop the subscript α. We let ∇_S denote the intrinsic gradient operator of S. Applied to a function $u \in \mathcal{C}^1(S)$, this operator can be written in local coordinates

$$\nabla_S u = \sum_{i,j} {g^{-1}}_{ij} \frac{\partial (u \circ x)}{\partial \xi_j} \frac{\partial}{\partial \xi_i}, \tag{1}$$

where $g_{ij} = I(\frac{\partial x}{\partial \xi_i}, \frac{\partial x}{\partial \xi_j}) = \frac{\partial x}{\partial \xi_i} \cdot \frac{\partial x}{\partial \xi_j}$ are the metric coefficients with respect to x. $I(\cdot,\cdot)$ is the first fundamental form, which at each $p \in S$ is the inner product on the tangent plane T_pS. The Laplace-Beltrami operator Δ_S is a generalization of the Laplace operator for manifolds. Applied to a function $u \in C^2(S)$, it is defined by the duality

$$\int_S \Delta_S u \, \phi \, dx = -\int_S I(\nabla_S u, \nabla_S \phi) + \int_{\partial S} \partial_{n^{co}} u \, \phi \, d\sigma, \qquad \forall \phi \in C^\infty(S), \tag{2}$$

where n^{co} is the co-normal, which at each $p \in \partial S$ is the normal vector of the curve ∂S lying in the tangent plane T_pS. Linear diffusion can now be formulated for a function u defined on a surface

$$\frac{\partial u}{\partial t} = \Delta_S u \qquad \text{on} \quad S \times [0, \infty) \tag{3}$$

Our interpolation method is based on applying these differential operators directly to the coordinate maps x. We will particularly focus on the geometric diffusion equation, defined as

$$\frac{\partial x}{\partial t} = \Delta_{S(t)} x \qquad \text{on} \quad S(t) \times [0, \infty). \tag{4}$$

Note that this equation is non-linear, as the differential operators depend on the unknown x. It can be shown that this equation describes motion by mean curvature (see [22] page 151). A disadvangate of this equation is certain shrinking effects on the volume enclosed by the surface [23]. Incorporation of boundary conditions will counteract most of these shrinking effects. An even better countermeasure is presented in Section 5.

Our interpolation method can now be formulated. We let x^{known} denote the coordinates of the scattered set of points. The surface S is reconstructed by solving the geometric diffusion equation with x^{known} as boundary conditions.

$$\frac{\partial x}{\partial t} = \Delta_{S(t)} x \qquad \text{on} \quad S(t) \times [0, \infty) \tag{5}$$

$$x = x^{known} \qquad \text{on} \quad \partial S(t) \times [0, \infty), \tag{6}$$

$$S(0) = S_0 \tag{7}$$

where S_0 is some initial guess. In order to be well posed in the continuous setting, x^{known} is assumed to be patches of non-zero measure. In the discrete setting, points always have a non-zero measure, so we avoid this problem.

As previously stated, the geometric diffusion equation can locally be seen as the surface equivalent of the linear diffusion equation, since the Laplace-Beltrami operator is applied to the coordinate maps. A different philosophy is to process the normal vectors, which yields higher order PDEs. See [24] for total curvature minimization in the level set framework. See [25] for minimization of Willmore energy and approximations on discrete meshes.

It could interesting to formulate the interpolation method with higher PDEs, however in this paper we focus on second order for a number of reasons: Higher order PDEs requires information of the normal vectors at the boundary in order to have a unique solution. From a compression point of view, storage of normal vectors at the interpolation points would require twice as much space. Numerical solution of higher order PDEs is computationally harder. Finally, we expect oscillation problems, since the boundary is so sparsely scattered. On the order hand, higher order PDEs have many attractive properties such as less shrinking effects, preservation of edges etc. They should be explored for the interpolation problem in the future. A generalization we have strongest belief in is anisotropic geometric diffusion [19].

3 Finite element discretization

Discretization of our continuous surface interpolation model is done by the finite element method. Using linear elements results in linear coordinate patches, which suits very well with triangulated surfaces. A finite element discretization for the Laplace-Beltrami operator was first proposed in [18], and for the geometric diffusion equation in [17]. We express equation (5) - (7) in weak form. That is, find $x(t)$ such that

$$\int_{S(t)} \frac{\partial x}{\partial t}\phi \, dx = -\int_{S(t)} I(\nabla_{M(t)} x, \nabla_{S(t)}\phi)\, dx \qquad t > 0 \tag{8}$$

$$S(0) = S_0, \tag{9}$$

for all $\phi \in C^\infty(S(t))^3 = V^3$ with $\phi = 0$ on $\partial S(t)$. Note that the boundary term in (2) vanishes, because the boundary is held fixed.

Spatial discretization is made by reducing V to a finite dimensional subspace $V_h \subset V$. The surface coordinate maps are thus reduced to a finite dimensional subspace. This will result in a discrete surface S_h. Letting $X(t)$ denote the discrete coordinate map, we then obtain

$$\int_{S_h(t)} \frac{\partial X}{\partial t}\phi \, dx = -\int_{S_h(t)} I(\nabla_{S_h(t)} X, \nabla_{S_h(t)}\phi)\, dx \qquad t > 0 \tag{10}$$

$$M_h(0) = M_{h_0} \tag{11}$$

for all $\phi \in V_h^3$ with $\phi = 0$ on ∂S_h. We choose the subspace consisting of linear polynomials on each triangle.

$$V_h = \{\phi \in C^0(M_h)\,\text{s.t.}\,|\phi|_T \text{ is a linear polynomial on each triangle } T\}.$$

Choosing time step τ and letting $X^n = X(n\tau)$ and $S_h^n = S_h(n\tau)$, we discretize semi-implicitly in time [17] by

$$\int_{S_h^n} \frac{X^{n+1} - X^n}{\tau}\phi \, dx = -\int_{S_h^n} I(\nabla_{S_h^n} X^{n+1}, \nabla_{S_h^n}\phi)\, dx \qquad n \in [0, 1, ...) \tag{12}$$

$$M_h^0 = M_{h_0}, \tag{13}$$

for all $\phi \in V_h^3$ such that $\phi = 0$ on ∂S_h^n. We thus observe that the surface S_h^{n+1} at time step $n+1$ is parameterized over the surface at the previous time step S_h^n. For each time step n, we choose the nodal basis $\{\Phi_i^n\}_{i=1}^m$ for V_h, where m is the number of vertices. That is, for each vertex $\bar{X}_i^n$ we associate a piecewise linear basis function Φ_i^n such that

$$\Phi_j^n(\bar{X}_i^n) = \delta_{ij}, \qquad i, j = 1, ..., m$$

The coordinate map can now be written in terms of the basis as $X^{n+1} = \sum_{i=1}^m \bar{X}_i^{n+1}\Phi_i^n$ and $X^n = \sum_{i=1}^m \bar{X}_i^n \Phi_i^n$. Using these expressions, equation (12) can be written as a linear system for the new vertex coordinates $\bar{X}^{n+1}$.

$$(M + \tau L)\bar{X}^{n+1} = M\bar{X}^n \qquad n \in [0, 1, ...) \tag{14}$$

where $M_{ij} = \int_{S_h^n} \Phi_i^n \Phi_j^n \, dx$ and $L_{ij} = \int_{S_h^n} I(\nabla_{S_h^n} \Phi_i^n, \nabla_{S_h^n} \Phi_j^n) \, dx$. At each time step, this linear system must be solved for each of the three vertex coordinates. Note that the stiffness matrix $(M + \tau L)$ is identical for each of the three linear systems, only the right hand side differs.

By the choice of the basis, we observe that $(M + \tau L)$ is very sparse, with non-zeroes mainly concentrated around the diagonal. To numerically solve the linear systems, iterative methods like the conjugate gradient method (CG) or SOR can be used. At each time step n, $\bar{X}^n$ can be used for initialization, since we expect $\bar{X}^{n+1}$ to be close to $\bar{X}^n$.

4 Surface compression

The new compression method can now be formulated. A triangular mesh is encoded by deleting all vertices that are reconstructed in acceptable quality by the PDE. Only the remaining vertices are stored on the computer. In the decoding step, the unknown vertices are reconstructed via PDE based interpolation.

4.1 Encoding

To achieve a compression rate of p percent, we aim to store only p percent of the vertices. The vertices should be selected such that the reconstruction error when using them for interpolation points is minimized. Intuitively, these points should be located at areas of high curvature. To efficiently find exact or suboptimal solutions of this combinatorial optimization problem is of interest in its own. Since the goal of this paper is not efficiency, but to demonstrate the potential of the method, we present a stochastic approach. The idea is to progressively reduce the number of vertices. This is achieved by interpolating in small randomly selected subsets of the vertices, and permanently removing the vertices with least reconstruction error:

```
Encoding Algorithm:
Set $V = \{\text{all vertices in the mesh}\}$.
Repeat until desired sparsification is reached:
    1. Randomly remove set $R$ of vertices (e.g. $|R| = 5\%$ of $|V|$).
    2. Interpolate in $V \setminus R$ with geometric diffusion.
    3. Permanently remove the set $P$, where reconstruction error is
       smallest (e.g. $|P| = 5\%$ of $|R|$)

                         $V \leftarrow V \setminus P$
```

In the interpolation step (step 2), the vertices in the known surface are used for initialization. Finally, a coarse triangulation topologically equivalent to the original mesh should be defined for the sparse set of vertices. This triangulation should be stored in addition to the set of vertices.

It is hard to assess the exact compression rate of our method at the current stage. We assume that if p percent of the vertices are to be stored, the coarse triangulation requires p percent as much storage as the original triangulation. Therefore, the compression rate is also p percent. Since the coarse triangulation can potentially be stored with much less space, the actual compression should be even higher.

4.2 Decoding

The surface is decoded by interpolating from the stored set of vertices V. Note that we do not aim to reconstruct the exact locations of the original vertex positions. Our aim is rather to reconstruct a surface which as close as possible resembles the original surface. Within that surface, we do not care how the vertices are distributed relative to the original surface.

Since only a small amount of vertices are known, the interpolation method requires an initialization of all the unknown vertices. Furthermore an initial triangulation must be defined on the set of all known and initial vertices. Note that such an initialization, also determines an initialization of the metric coefficients of the discrete Laplace-Beltrami operator.

The initialization can be computed in many ways. Starting with the coarse mesh, a possibility is to iteratively refine and solve the PDE, introducing more vertices in each iteration. At each step, the mesh should be topologically equivalent to the stored mesh. In our experiments, we have simply used the known mesh for initialization. This will not have any influence on the final result, since under Dirichlet boundary conditions, the steady state of the geometric diffusion equation is unique.

5 Surface and image approximation

It is well known that the diffusion equation results in a smooth functions everywhere, except at the boundary where we in general cannot say anything about the smoothness. In fact, as the area of the boundary points goes to zero, a logarithmic singularity will arise at each boundary point. For the interpolation problem, this can be observed as a sharp cusp at each interpolation point, although in the discrete setting the singularities are not fully developed. The phenomena happens both for surfaces and images, but is much more visible for surfaces.

We now present a method to counteract this problem, which is based on approximation instead of interpolation. Afterwards, a further improvement is presented which is also able to prevent shrinking effects, and yields a sharper reconstruction at edges.

The idea is to iteratively exchange the role of interpolation domain and boundary domain. Letting S_1 denote the set of unknown vertices, and S_2 the set of known vertices, the algorithm can be sketched as follows.

```
Modification I:
for n=1,...,N:
    1. Interpolate $S_1^{n+1}$ with $S_2^n$ fixed
    2. Interpolate $S_2^{n+1}$ with $S_1^{n+1}$ fixed
```

where S_i^0 is the input and S_i^N is the output for $i = 1, 2$.

This process may be interpreted as Hopscotch iteration for the geometric diffusion equation in the whole domain. Hopscotch iteration was developed in the 1960's as a numerical method for solving time dependent PDEs [26,27]. The idea is to decompose the spatial domain into two or more subdomains consisting of scattered regions. The PDE is solved numerically by alternatingly taking one time step in each subdomain, with function values in the inactive subdomain as boundary conditions. Attempts to analyze the numerical error has been made in the above mentioned references. For a given compression rate, S_1 and S_2 each contain a certain percentage of vertices. Therefore, this method is independent on the resolution of the mesh.

Note that this method achieves smoothness at the expense of accuracy, as the surface will be progressively diffused in each time step. However, the smoothness will be significantly improved after only one iteration.

A further improvement can be made. For decompression, we observe that the interpolation points are located at locations of sharp edges and corners, where the curvature is large. We therefore propose the same iterative scheme, but instead project the displacements of the interpolation points back in their opposite direction. This can be seen as inverse diffusion applied to the interpolation points. The motivation is to imitate the behaviour of edge enhancing nonlinear PDEs, which perform inverse diffusion at locations of sharp corners. In order to maintain a smooth result, backward diffusion is only applied every second iteration. The algorithm is sketched below.

```
Modification II:
for n=1,2,...,N:
    1. Interpolate $S_1^{n+1}$ with $S_2^n$ fixed.
    2. Interpolate $S_2^{n+1}$ with $S_1^{n+1}$ fixed.
    3. if $n$ is odd: $S_2^{n+1} \leftarrow S_2^n - 2S_2^{n+1}$
```

We experienced that usually the best results were obtained with one iteration of modification I, or two iterations of modification II. All experiments in this paper are made with these number of iterations.

We have also applied this method for image compression with linear diffusion. In this case S_1 and S_2 can be replaced by Ω_1 and Ω_2, where Ω_1 is the domain of unknown pixels, and Ω_2 is the domain of known pixels. Analyzing the L1 error, we are able to obtain equally good reconstruction results based on linear diffusion as the more expensive nonlinear anisotropic diffusion.

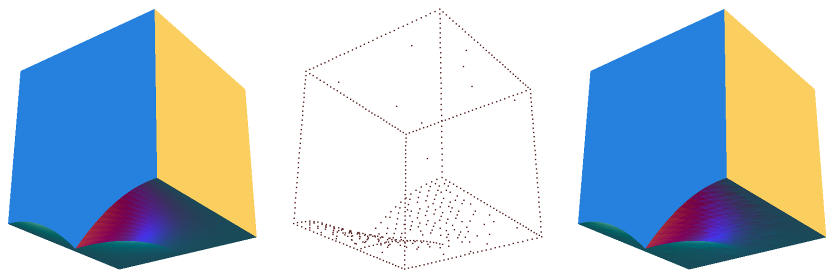

(a) Ground truth (b) interpolation points (c) Geometric diffusion

Fig. 1. Reconstruction from 10% of vertices

6 Numerical experiments

This section presents numerical results. The mean L2 error measures for all surface experiments, computed by the metro tool [28], are shown in Table 1. An illustrative example is shown in Figure 1, where 10 percent of the interpolation vertices have been selected. The coordinates of these vertices are depicted as dark dots in Figure 1(b). This demonstrates that the encoding step tends to select vertices at locations of high curvature, such as sharp edges. For this particular example, the compression is not lossy, as the exact surface can be reconstructed.

The next example, Figure 4, aims to show the differences between the interpolation method and the different approximation modifications. For this low resolution mesh, the sharp cusps at each interpolation point are particularly visible, Figure 4 (b). As seen in Figure 4(c)(d), the two modifications significantly reduces the problem. In addition, modification II yields a sharper reconstruction with less shrinking effects. Although it may be hard to see this difference when the surfaces are placed next to each other, the effect is clear from the error measures in Table 1, Moai. As expected, modification I yields slightly higher errors than geometric diffusion, while modification II is significantly better than both geometric diffusion and modification I.

Some more realistic examples are depicted next. In Figure 5, we compare interpolation with geometric diffusion, modification I, and modification II. For this example, 90 percent of the vertex coordinates have been deleted in the encoding step. We observe that the overall shape can be reconstructed quite well with geometric diffusion, Figure 5(b). A disadvantage is the sharp cusps at each interpolation point. The result of one iteration of modification I is shown in Figure 5(c). The surface is smoother, but has the drawback of smooth corners and edges, and slight shrinking effects. Figure 5(d) is the result of two iterations of modification II, which we observe yields both a sharper results at edges, while removing singularities at the boundary points. See also Table 1, Armadillo.

Another example is shown in Figure 6, where 95 percent of the vertex coordinates have been removed. This example demonstrates that quite amazing results

Table 1. Mean L2 errors (metro) for surfaces

	Compression rate	Geometric diffusion	Modification I	Modification II
Moai	2 percent	-	-	-
	5 percent	0.047	0.050	0.024
	10 percent	0.021	0.024	0.009
Armadillo	2 percent	0.218	0.226	0.171
	5 percent	0.128	0.138	0.082
	10 percent	0.065	0.079	0.049
Max-Planck	2 percent	0.107	0.112	0.091
	5 percent	0.074	0.082	0.056
	10 percent	0.045	0.059	0.033

can be obtained with geometric diffusion. The improved result with modification II is also shown. At such high compression rate some small scale detail, such as wrinkles, may become less prominent.

We also demonstrate the approximation method presented in Section 5 for image compression. In Figure 2 and 3, the reconstruction results of linear diffusion, anisotropic diffusion and modification II are shown. In Table 2, we compare the L1 reconstruction error of the different PDEs. We observe that modification II yields equally good results as the more sophisticated anisotropic diffusion, although it seems to have a slightly lesser ability of preserving edges. Anisotropic diffusion was shown to outperform the well known JPEG standard in [4]. Since modification II is based on linear diffusion, it is several times faster than anisotropic diffusion.

7 Conclusions

We have presented new PDE based methods for interpolation, approximation and compression of triangulated surfaces. Experiments show that amazingly good results can be obtained with the simple geometric diffusion equation acting on the coordinate map from unoriented points (i.e. without any normal vectors) when the interpolation points are selected properly.

This work may only be considered a proof of concept, as further developments are necessary for a practical method. In the future, we will study anisotropic geometric diffusion equations [19] and develop a faster algorithm for selecting the

Table 2. Average absolute errors (L1 errors) for images

	Linear diffusion	Anisotropic diffusion	Modification II
Trui	14.614	7.697	8.037
Peppers	12.967	10.591	9.471

best interpolation points in the encoding step. Comparisons with other methods will also be made.

Acknowledgements

Thanks to Martin Welk, Alexander Belyaev and Michael Breuss for interesting discussions. We also thank Alexander Belyaev for providing useful software and database of meshes.

References

1. Khodakovsky, A., Schroeder, P., Sweldens, W.: Progressive geometry compression. In: Proceedings of Siggraph. (2000) 95–102
2. DeVore, R.A., Jawerth, B., Lucier, B.J.: Surface compression. Computer Aided Geometric Design **9** (1992) 219–239
3. Davis, G.M., Nosratinia, A.: Wavelet-based image coding: An overview. In Datta, B.N., ed.: Applied and Computational Control, Signals, and Circuits. Volume 1. Birkhäuser, Boston (1999) 205–269
4. Galić, I., Weickert, J., Welk, M., Bruhn, A., Belyaev, A., Seidel, H.P.: Image compression with anisotropic diffusion. Journal of Mathematical Imaging and Vision **31** (2008) 255–269
5. Masnou, S., Morel, J.M.: Level lines based disocclusion. In: 5th IEEE International Conference on Image Processing. (1998) 259–263
6. Bertalmio, M., Sapiro, G., Ballester, C.: Image inpainting. In: Proc. SIGGRAPH 2000. (2000) 417–424
7. Chan, T.F., Shen, J.: Non-texture inpainting by curvature-driven diffusions (CDD). Journal of Visual Communcation and Image Representation **12** (2001) 436–449
8. Chan, T.F., Ha, S., Kang, Shen, J.: Euler's elastica and curvature based inpaintings. SIAM Journal of Applied Mathematics **63** (2002) 564–592
9. Verdera, J., Bertalmio, M., Sapiro, G.: Inpainting surface holes. In: In Int. Conference on Image Processing. (2003) 903–906
10. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing, Aire-la-Ville, Switzerland, Eurographics Association (2006) 61–70
11. Zhao, H.K., Osher, S., Fedkiw, R.: Fast surface reconstruction using the level set method. In: VLSM '01: Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01), Washington, DC, USA, IEEE Computer Society (2001) 194–201
12. Kobbelt, L.: Discrete fairing. In: In Proceedings of the Seventh IMA Conference on the Mathematics of Surfaces. (1997) 101–131
13. Schneider, R., Kobbelt, L.: Discrete fairing of curves and surfaces based on linear curvature distribution. In: In Curve and Surface Design: Saint-Malo, University Press (2000) 371–380
14. Weickert, J.: Theoretical foundations of anisotropic diffusion in image processing. Computing, Suppl **11** (1996) 221–236
15. Belhachmi, Z., Bucur, D., Burgeth, B., Weickert, J.: How to choose interpolation data in images. Technical Report No. 205, Department of Mathematics, Saarland University, Saarbrücken, Germany (2008)

16. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co. (1999) 317–324
17. Dziuk, G.: An algorithm for evolutionary surfaces. Numerische Mathematik **58** (1991) 603–611
18. Dzuik, G.: Finite elements for the Beltrami operator on arbitrary surfaces. Partial Differential Equations and Calculus of Variations **1357** (1988) 142–155
19. Clarenz, U., Diewald, U., Rumpf, M.: Anisotropic geometric diffusion in surface processing. In: VIS '00: Proceedings of the conference on Visualization '00, Los Alamitos, CA, USA, IEEE Computer Society Press (2000) 397–405
20. Taubin, G.: A signal processing approach to fair surface design. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1995) 351–358
21. Sorkine, O., Cohen-Or, D., Toledo, S.: High-pass quantization for mesh encoding. In: SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing, Aire-la-Ville, Switzerland, Switzerland, Eurographics Association (2003) 42–51
22. Carmo, M.P.D.: Riemannian geometry. Birkhauser, Bosten-Basel-Berlin (1993)
23. Sapiro, G.: Geometric partial differential equations and image analysis. Cambridge University Press (2001)
24. Tasdizen, T., Whitaker, R., Burchard, P., Osher, S.: Geometric surface processing via normal maps. ACM Trans. Graph. **22**(4) (2003) 1012–1033
25. Bobenko, A.I., Schröder, P.: Discrete willmore flow. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Courses, New York, NY, USA, ACM (2005) 5
26. Gordon, P.: Nonsymmetric difference schemes. Journal of the Society of Industrial and Applied Mathematics **13**(3) (1965) 667–673
27. Gourlay, A.: Hopscotch: a fast second-order partial differential equation solver. IMA Journal of Applied Mathematics **6**(4) (1969) 375–390
28. P. Cignoni, C.R., Scopigno, R.: Metro: measuring error on simplified surfaces. Computer Graphics Forum, Blackwell Publishers, vol. 17(2) (1998) 167–174

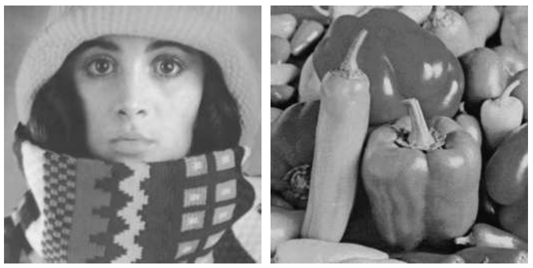


Fig. 2. Input images. Left: Trui, Right: Peppers. Resolution: 256×256

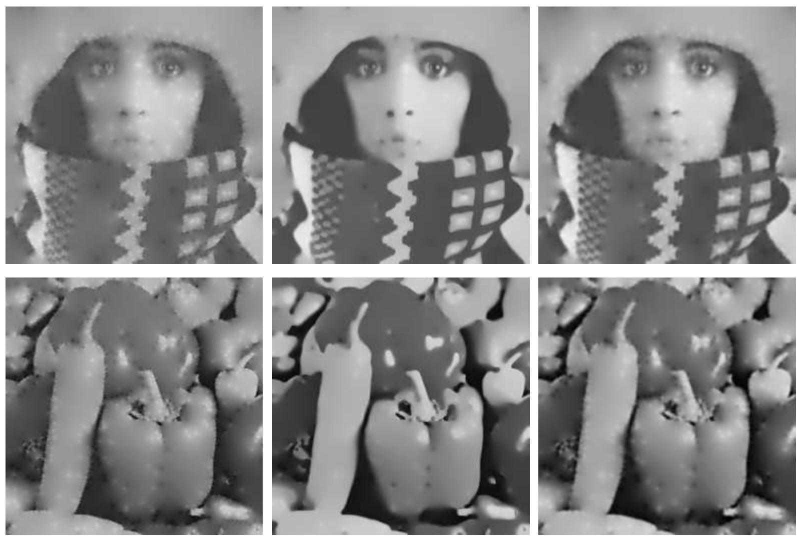
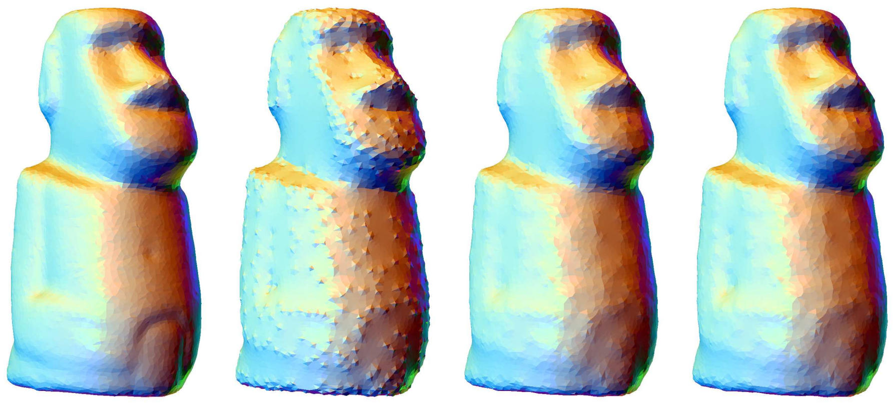
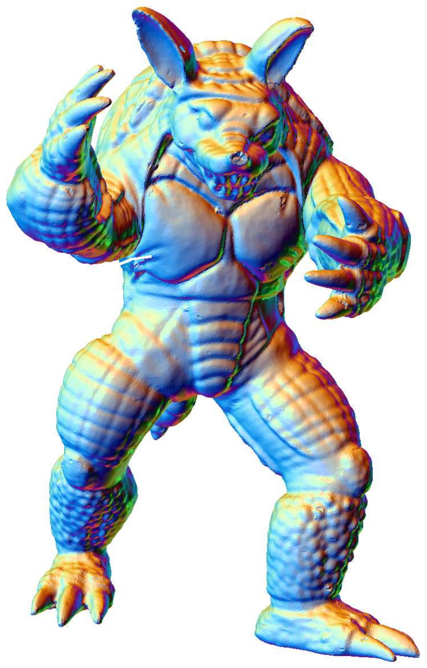


Fig. 3. Reconstructions from 2.5 % of pixels. Top: Trui. Bottom: Peppers. From left to right: linear diffusion, anisotropic diffusion, modification II.

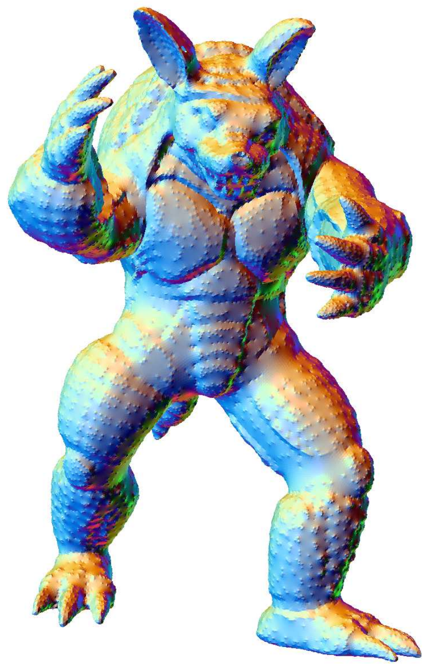


(a) Ground truth (b) Geometric diffusion (c) Modification I (d) Modification II

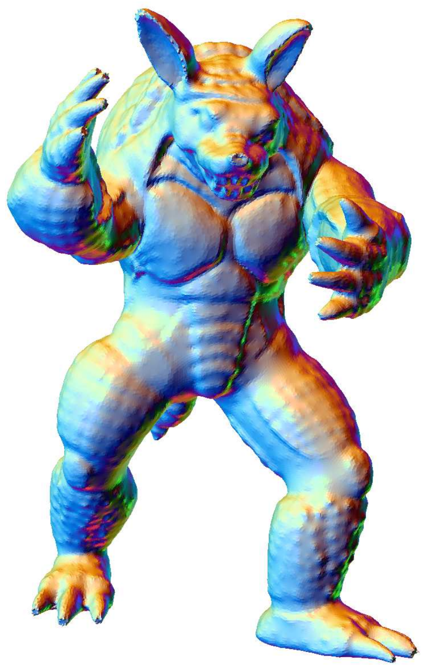
Fig. 4. Reconstruction from 10% of vertices, Moai experiment. Total number of vertices: 10002, total number of triangles: 20000. (a) Ground Truth, (b) interpolation by geometric diffusion, (c) modification I, (d) modification II



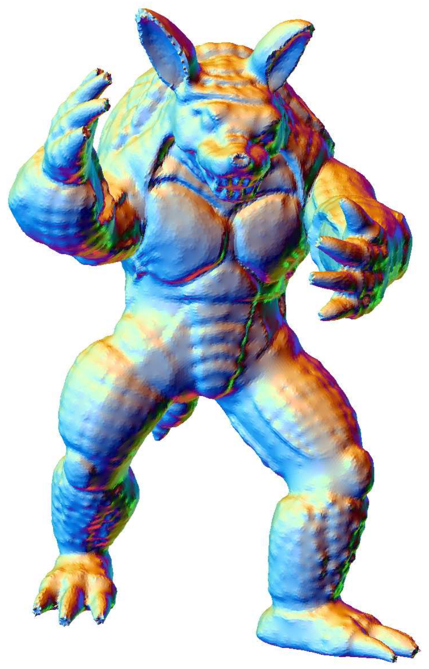
(a) Ground truth



(b) Geometric diffusion

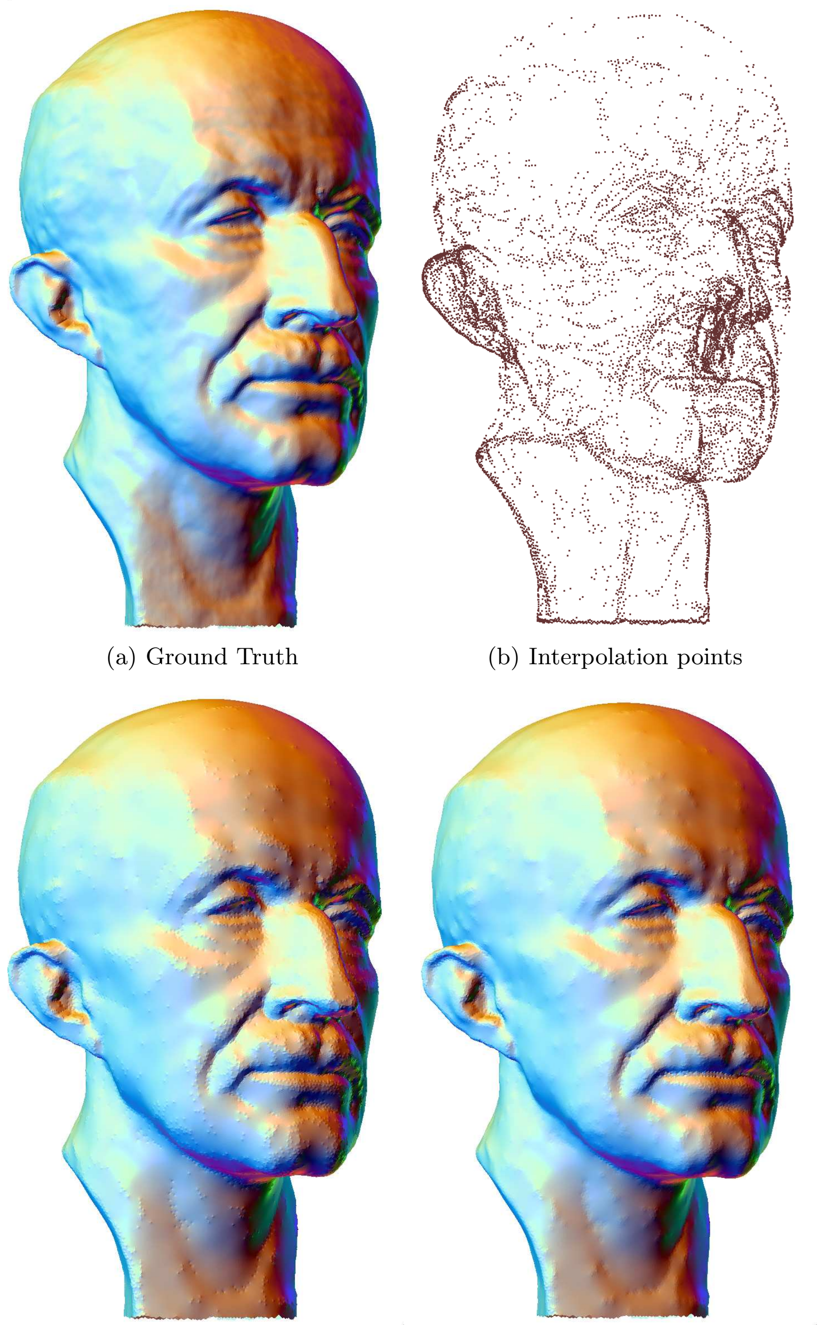


(c) Modification I



(d) Modification II

Fig. 5. Reconstruction from 10% of vertices, Armadillo man experiment. Total number of vertices: 165954, total number of triangles: 331904. (a) Ground Truth, (b) interpolation by geometric diffusion, (c) modification I, (d) modification II



(a) Ground Truth (b) Interpolation points

(c) Geometric diffusion (d) Modification II

Fig. 6. Reconstruction from 5% of vertices, Max Planck experiment. Total number of vertices: 199169, total number of triangles: 398043. (a) Ground Truth, (b) selected interpolation points, (c) interpolation geometric diffusion, (d) modification II