# Fast Approximation of the Shape Diameter Function

Maurizio Kovacic
Dept. Mathematics & Computer Science
University of Cagliari
Via Ospedale, 72
09124, Cagliari (Italy)
mau.kovacic@gmail.com

Fabio Guggeri
Dept. Mathematics & Computer Science
University of Cagliari
Via Ospedale, 72
09124, Cagliari (Italy)
guggeri@unica.it

Stefano Marras
Dept. Mathematics & Computer Science
University of Cagliari
Via Ospedale, 72
09124, Cagliari (Italy)
stefano.marras@unica.it

Riccardo Scateni
Dept. Mathematics & Computer Science
University of Cagliari
Via Ospedale, 72
09124, Cagliari (Italy)
riccardo@unica.it

**ABSTRACT**

In this paper we propose an optimization of the Shape Diameter Function (SDF) that we call Accelerated SDF (ASDF). We discuss in detail the advantages and disadvantages of the original SDF definition, proposing theoretical and practical approaches for speedup and approximation. Using Poisson-based interpolation we compute the SDF value for a small subset of randomly distributed faces and propagate the values over the mesh. We show the results obtained with ASDF versus SDF in terms of timings and error.

**Keywords:** Segmentation, Poisson equation.

## 1 INTRODUCTION

The Shape Diameter Function (SDF) [18] has proven very useful for mesh skeletonization and segmentation. Closely related to the Medial Axis Transform (MAT) [5], it defines a scalar function over the points of a mesh representing the diameter of the shape's volume at each point while being computationally lightweight as compared to the MAT. The main contribution of the SDF is its taking into account the interior of the mesh and its volumetric information as opposed to the majority of segmentation algorithms that rely on local surface features as curvature; for each primitive it computes the diameter of the object along its interior, the result is a pose invariant function of the local volume that yields a good mesh partitioning. Moreover, it defines a set of internal points for each primitive, halfway through its interior, that can be used for skeleton extraction. In this paper we focus on optimization for segmentation purposes: a study of the behavior of the SDF function over the mesh suggests that, as main variations occur only in a small subset of the faces, it is possible to lower the number of computations with little to no effect on the final result by means of a constrained Poisson interpolation over the mesh (see figure 1 for an example). We will present a summary of the published works related to our problem in Section 2; Section 3 will contain a detailed description of the original SDF algorithm, while details on the Poisson Equation used to efficiently interpolate the values can be found in Section 4, with results and comparison between the optimized version and the original implementation in Section 5. Section 6 contains our conclusions and suggestions for future works.

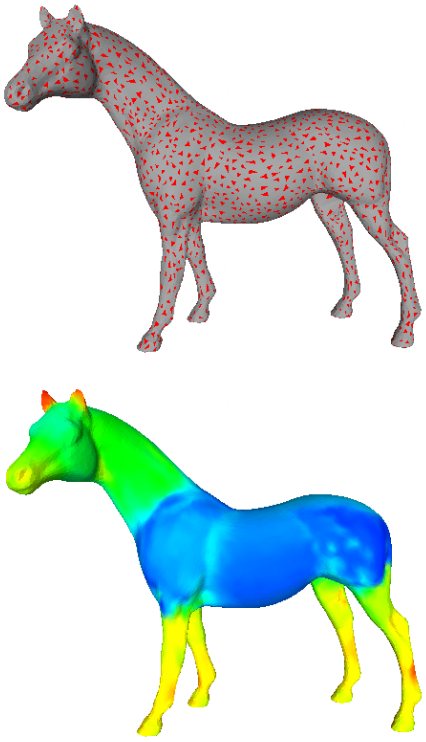

**Figure 1: The SDF function is computed only on the selected faces (red in the top image). The constrained interpolation results in a smooth and accurate descriptor (bottom image).**

## 2 RELATED WORK

### 2.1 Mesh Segmentation

There is a large amount of research works that study segmentation with many different strategies aiming to extract a semantically consistent partitioning. Most of the approaches rely on local surface properties for segmentation, as geodesic distance, angular distance or normal direction. In [12] the mesh is partitioned using the minima rule, that states that human perception divides shapes along the concave discontinuities; a snake contour is relaxated along the surface until convergence using geometric features as curvature and centricity. Minima rule is used also in [16] where the authors define an algorithm called Fast Marching Watersheds that identifies the regions bounded by contours of negative curvature; a similar technique can be found in [14] where an extension of the 2D morphological watershed segmentation over the mesh is used to partition the shape according to curvature. Katz and Tal, in [10], adopt the strategy to decompose the mesh in a hierarchical manner using fuzzy clustering according to a combination of surface metrics and using minimum graph cuts to extract the patch boundaries. Cohen-Steiner and colleagues in [8] propose a framework for shape approximation based on shape partitioning, obtaining a face clustering that minimizes an error metric based on normal deviation. These surface features may provide good results, however their sensitiveness on local surface variations makes them unsuitable for a pose invariant segmentation. The main advantage of the SDF algorithm is that it takes advantage of the connection between surface and volume of the object; such kind of approach can be found in [15], where the mesh is intersected with a sphere over each vertex and the behavior of the intersections is used for partitioning, or in [3] where geometric primitives are iteratively fitted to the mesh.

### 2.2 Applications of the Poisson Equation in Computer Graphics

The Poisson equation (PE) has been used in a large number of application areas. In computer graphics and related fields, one of the main applications of Poisson equation is the image processing. Perez and colleagues in [17] used Poisson equation in order to modify the content of an image; particularly, the gradient of the original image is computed, then modified according to desired result, and finally Poisson equation is solved to obtain a new image. Perez uses this approach in order to enhance the original image, by correcting local illumination or changing local color by defining appropriate constraint fields; the solution of the PE will be the image that best fit these constraint fields. In image processing, the Poisson equation has been used also for seamless image mosaic (as presented in [13], [20] and [19]), since the Poisson-based interpolation guarantees a smooth transition between two images (a property that makes Poisson equation also suitable for photomontage, in [1]). Poisson equation has been used also in geometry processing and mesh editing system. For example, in [22], gradient fields are used in order to model coordinate functions, and mesh editing is performed by locally adapting the gradients and solving the Poisson equation for the new coordinate functions. In this way, it is possible to perform operations such as deformation, merging, smoothing and denoising. Alexa [2] uses discrete Laplace and Poisson models to perform mesh editing and detail transfer from one mesh to another, while Xu et al. [21], in order to realize shape interpolation, use PE to find an intermediate shape that allows a smooth transition between the source shape and the target shape. PE has been also used in order to reconstruct the surface of a mesh starting from a point cloud as shown in [11] by Kazhdan et al.
Also, there is a number of definition and formulation of the Laplace-Beltrami operator, such as in [4], [6] or [7]; in this work, we propose an implementation of the Laplace-Beltrami operator which is slightly different from the previous ones but that showed to perform well and to be easy to implement.

## 3 THE SHAPE DIAMETER FUNCTION

The main idea behind the Shape Diameter Function is to take into account the volumetric information of the shape by defining a scalar function on the mesh representing the diameter of the interior of the object, similarly to the MAT where the scalar field represents the distance between each point to the nearest boundary point. However, while the MAT is computationally expensive and requires a discretization of the space, the SDF results in a faster and more robust descriptor.

Given a mesh M the SDF is a scalar function on the surface ($f_p : M \rightarrow \mathbb{R}$) defined as the neighborhood diameter of the object at each surface point $p \in M$. Such diameter is extracted by casting a cone of rays from the point $p$ to the interior of the mesh according to the inverse normal at $p$ and computing the distance between $p$ and each intersection between the rays and the mesh. In order to improve the robustness of this approach, *false* intersections are removed from the computation: those intersection points whose normal is in the *same direction* as

the point $p$, that is when the angle between the normals is less than 90° are not considered in the final computation. The definition is extremely simple and intuitive, however it is highly sensitive to noise and local variations; further improvement is obtained by considering just those rays whose length fall within a standard deviation from the median of all lengths in order to remove spurious intersections. The final value of the SDF is a weighted average of the remaining lengths: the weights are the inverse of the angle between the ray to the center of the cone, due to the fact that rays with larger angles are much more frequent and therefore must have smaller importance in the final averaging.

The authors show that the best results are obtained by casting 30 rays into a cone of 120° per point; smaller angles don't discriminate between the object parts and are extremely sensitive to local features, whilst larger angles cause some rays to intersect unrelated parts of the mesh and add errors to the computation. In our paper we stick to this parameters to produce comparable results with the ones provided in the original algorithm.

The algorithm as defined doesn't guarantee pose invariance. The authors propose a small number of bilateral filtering steps in order to reduce the variation of the SDF value after a pose change; one may refer to the original paper for the formulation of the filtering as it is unnecessarily verbose for our purposes. As for the intersection search, an Octree is used to spatially index the elements of the mesh for a reduced number of ray-triangle intersection tests.

# 4 ACCELERATED SDF (ASDF) VIA POISSON INTERPOLATION

The main contribution of our optimization method is based on an observation of the behavior of the SDF function for regular meshes. On simple meshes the difference between the SDF value of a primitive (face or vertex) and the SDF value of its neighborhood is approximately zero for primitives in the same *part* of the object and it increases smoothly on the boundaries of the *parts*. This means that little to none additional information is obtained by the SDF computation for a vertex or face whose neighbors have already been evaluated. Furthermore, the bi-lateral filtering step proposed in the original paper lowers the importance of a single computation in the final output. Figure 2 shows the normalized, absolute value Laplacian of the SDF computed for each face according to the formula:

**Figure 2: The absolute differences in SDF value between each face and its neighborhood, coded from blue where close to zero to red where maximum, shows that the function has very small variations on most of the surface**

$$F(p) = |\sum_{v \in N(p)} w_v SDF(p) - \sum_{v \in N(p)} w_v SDF(v)|$$

where $N(p)$ is the neighborhood of $p$ and $w_v$ is the weight of face $v$ defined as 1 over the distance between the barycenter of $v$ and the barycenter of $p$. In the image, the color red means zero or close to zero while blue is the maximum difference. It is worth to notice that the difference in SDF is higher where the shape changes sharply: thus, for segmentation purposes, it is possible to approximate the SDF on the whole mesh by propagating the function value computed on a small subset of faces. The mean of propagation we choose is solving a Poisson equation with Dirichlet boundary conditions, a technique that obtained good results in mesh editing [22] and image processing [17] under similar circumstances. This technique allows to easily compute a constrained interpolation over the mesh guaranteeing computational efficiency and robustness of the results.

## 4.1 Poisson Equation

The formulation of the Poisson equation that we use is defined as follows:

$$\Delta f = \nabla \mathbf{v} \quad \text{with } f|_{\partial\Omega} = f^*|_{\partial\Omega}$$

where $f$ is an unknown scalar function, $\mathbf{v}$ a guidance vector field, $\nabla \mathbf{v}$ is the *divergence* of $\mathbf{v}$, $\Delta$ is

the Laplace operator and $f^*$ defines the values of a known scalar function at the boundary $\partial\Omega$ of a selected region $\Omega$. Solving this equation allows to reconstruct the unknown function by interpolating the boundary values so that the gradient of $f$ is as close as possible to the vector field $\mathbf{v}$, resulting in a smooth and seamless propagation that satisfies some user prescribed conditions; the unknown values in the user-selected region $\Omega$ are set to the known function $f^*$ in the border of $\Omega$ so that no seam is visible between the known and unknown regions, and the values change smoothly according to $\mathbf{v}$.

In our framework, the SDF is both the known and unknown scalar function and $\Omega$ is defined as the set of faces whose SDF hasn't been computed yet. $f$ consists of the known, exactly computed values of SDF where $f^*$ is the interpolated value of the SDF function where no actual ray casting will be performed. The choice of the guidance vector field for the interpolation is nontrivial: we want the Laplacian of the propagated SDF to be the same as the divergence of $\mathbf{v}$, thus we need to understand the behavior of the SDF function according to its neighborhood.

A good approximation of the divergence can then be obtained by the opposite of the curvature on each face $curv(p)$ (computed as the mean of its vertices' curvature): it is plausible to expect the diameter of a shape to slightly increase where the Gaussian curvature is less than zero, that is, where the normals of the faces converge and the neighborhood is concave. This assumption, while not taking into account the shape of the other side of the mesh, is still good enough for small neighborhoods.

The final formula for each unknown face is:

$$\sum_{v \in N(p)} w_v f(p) - \sum_{v \in N(p) \cap \Omega} w_v f(v) = \sum_{v \in N(p) \cap \partial\Omega} w_v f^*(v) - curv(p)$$

where the Laplacian is computed on the dual graph of the mesh. The above formulation causes each unknown SDF to be a function of the known values over the boundary and the local curvature, whereas a face with no known neighbors will obtain a value completely dependent on the curvature variation.

## 4.2 Face Selection

We did not point out how we select the faces over which we compute the SDF. In early stage of our development we were using fancy schemes for identifying the correct subset of set that were thought to maximize the *dispersion* of the selected faces over the mesh.

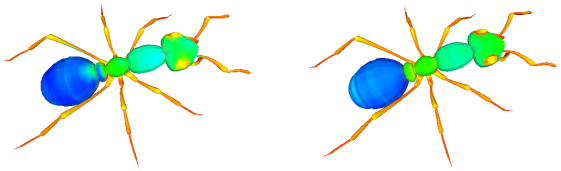

**Figure 3: SDF computation on the ant using 5% (left) and 100% (right) of the primitives, shown from red to blue according to the local thickness. It is noticeable how the presence of small local differences doesn't affect the global result with each segment easily distinguishable**

This choice had one major advantage: the deterministic choice of the face subset, but also a clear disadvantage since the selection stage was time consuming and the improvement over the original SDG was definitely moderate.

We then decided to follow a randomized scheme in selecting the faces. We, thus, compute an initial permutation of the face set and, in a second step, we select any single face in a simple manner, just picking every $n^{th}$ face and setting a flag on it identifying it as a *seed* for the solution of the Poisson equation. The mix of randomized input and Poisson equation revealed to be the best choice for our purposes.

# 5 RESULTS AND DISCUSSION

In this section we discuss the results obtained by our method in terms of time and error between the optimized and unoptimized version. For segmentation purposes it isn't mandatory that the SDF value of each single primitive is correct; in fact the segmentation process tends to split patches where the change rate is high on a significant area, while ignoring local peaks on the gradient. Therefore the correctness of a single face or vertex is discarded in favor of an overall correctness that is achieved by our propagation algorithm for a dense enough sampling. We can see in figure 3 that there is no substantial difference between the downsampled (5%) and original (100%); the differences in the values are restricted to a local point of view, whereas the global segmentation remains consistent.

To further discuss the relevance of these differences we show in figure 4 a map of the errors between a 10% subsampled and a complete SDF with blue being zero and red being the max difference. We can see how the highest differences are located along the boundaries of the mesh parts, due to the

sensitiveness of the original SDF definition on diameter variation.

However table 1 shows that the magnitude of this differences is low and doesn't strongly influence the outcome of the segmentation; moreover, the peaks in the errors occur on a small set of boundary faces, rapidly decreasing in their neighborhood: this may result in a fuzziness of the final segmentation border, with a negligible number of faces that are assigned to a patch that is different from the expected one, but still no substantial errors in the final segmentation.

As for the computational advantages of this optimization, we show in figure 5 a plot of the maximum and average error over the percentage of samples. Figure 6 shows the timings for the same computations. Timings don't reflect the ones presented in [18] and are obtained by an unoptimized single-thread implementation using the VCG library for the ray-triangle. We can anyway obtain an implementation independent speedup with a small error cost.

What is relevant of the ADSF can be understood looking with one eye at table 1 and the other at figure 5 reporting data about the same mesh: this will tell us that introducing less then 3% of error in the computation selecting only one face every tenth (as already mentioned this does not influence the overall segmentation at all) we gain one order of magnitude in the time spent for the computation passing from sixty to slightly more than six seconds.

## 6 CONCLUSIONS AND FUTURE WORK

In this paper we presented our proposal for speeding-up an algorithm for mesh segmentation that uses the SDF function. We showed how the Poisson equation defined on the mesh vertices can be used to propagate, with a limited error, the value of a subsampled SDF by computing the function for a randomly distributed set of faces.

| Faces % | Max SDF error | Avg. SDF error |
|---|---|---|
| 50% | 0.4022 | 0.0099 |
| 20% | 0.5054 | 0.0198 |
| 10% | 0.5292 | 0.0272 |
| 5% | 0.5538 | 0.0374 |
| 2% | 0.5374 | 0.0575 |
| 1% | 0.5813 | 0.0695 |

**Table 1: Max and average error in SDF according to the percentage of faces used for computation. The number are normalized over the SDF values, so they ranges from 0 to 1. The computation was performed on the horse mesh.**

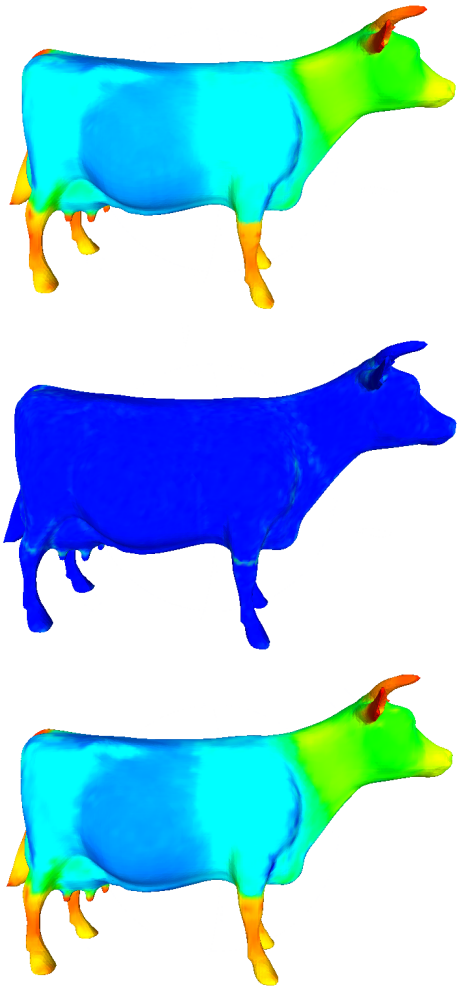

**Figure 4: In this picture you can visually appreciate the difference between the SDF computed on all the faces and a subset (10% of the total number). The image on top represents the result of the computation of the SDF on all the faces, while the one in the bottom represents the results of the computation performed on only 10% of the faces. In the middle image, the differences between the two results are graphically mapped on the mesh, with blue indicating no difference and colors towards red indicating larger and larger differences.**

The percentage of the samples can go down to 5% without sensible differences in the outcome.

A future improvement that we would like to explore is the choice of the samples according to their morphological significance instead of a random choice (see figure 7 for an example); we would like to study the behavior of our strategy if using a Gaussian sphere subsampling [9] where the samples are uniformly distributed over a Gaussian

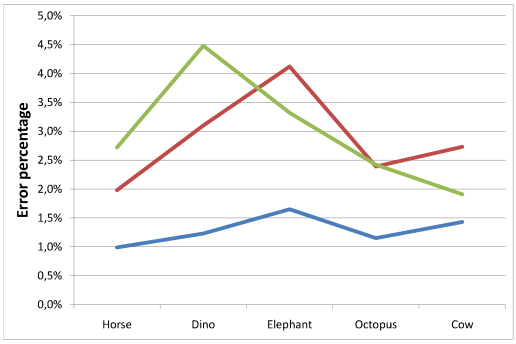


**Figure 5: Average error in SDF in function of the percentage of faces used for five different meshes. Green line are errors when selecting 10% of the faces, red line when selecting 20% and blue line when selecting 50%.**

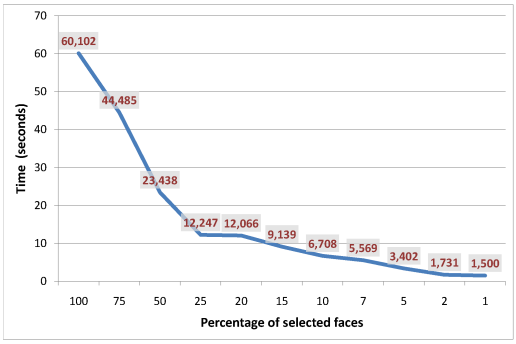


**Figure 6: Computational times in function of the number of selected faces. The times were taken while processing the horse mesh.**

sphere according to their normal resulting, therefore, more representative of the shape.

We also plan to improve the spatial organization for the intersection search: computing the ray-mesh intersection is a well known problem in Computer Graphics even outside the computational geometry area. In fact there is a lot of work on ray-tracing due to its centrality in rendering, and many of the techniques adopted in this field can be applied to the SDF problem in order to optimize the intersection search. While the original paper uses octrees as a mean of spatial indexing, it's reasonable to think that a KD-Tree can outperform it even when the data grow large or huge. It would be interesting to see how a specialized structure can further lower the computational times. One more open issue is how the parallelism implicit in ray-tracers can be exploited to work out a GPU implementation of the whole accelerated SDF. Furthermore, works on ray-tracing showed the usefulness of the SIMD paradigm with ray packing.

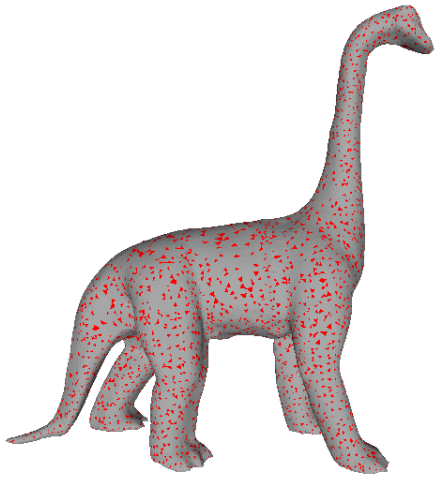

**Figure 7: Random face sampling (7%) - The quasi-uniform sampling gives no weight to the features of the mesh.**

## ACKNOWLEDGEMENTS

We would like to thank Lior Shapira for his suggestions and helpfulness.

## REFERENCES

[1] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 294–302, New York, NY, USA, 2004. ACM.

[2] Marc Alexa. Mesh editing based on discrete Laplace and Poisson models. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 51–59, New York, NY, USA, 2006. ACM.

[3] Marco Attene, Bianca Falcidieno, and Michela Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181–193, 2006.

[4] Mikhail Belkin, Jian Sun, and Yusu Wang. Discrete Laplace operator on meshed surfaces. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 278–287, New York, NY, USA, 2008. ACM.

[5] H. Blum. A transformation for extracting new descriptions of shape. In *Models for the Perception of Speech and Visual Form*, pages 362–380, 1967.

[6] Alexander I. Bobenko. Delaunay triangulations of polyhedral surfaces, a discrete

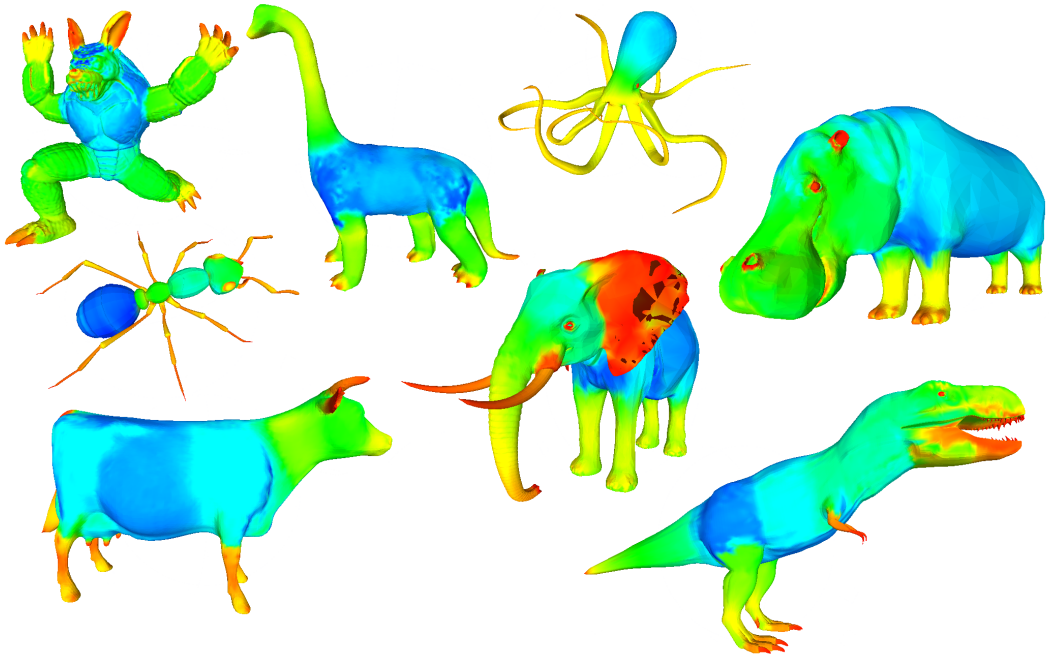

**Figure 8: Examples of application of ASDF to several different meshes. We preferred, for sake of understanding, here and in the rest of the paper, to show just the result of ASDF instead of the results of the segmentation phase.**


Laplace-Beltrami operator and applications. In *SCG '08: Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 38–38, New York, NY, USA, 2008. ACM.

[7] Ming Chuang, Linjie Luo, Benedict J. Brown, Szymon Rusinkiewicz, and Michael Kazhdan. Estimating the Laplace-Beltrami operator by restricting 3D functions. *Computer Graphics Forum*, 28(5):1475–1484, July 2009.

[8] David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. Variational shape approximation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 905–914, New York, NY, USA, 2004. ACM.

[9] Pablo Diaz-Gutierrez, Jonas Bösch, Renato Pajarola, and M. Gopi. Streaming surface sampling using gaussian $\varepsilon$-nets. *The Visual Computer*, 25:411–421, 2009.

[10] Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 954–961, New York, NY, USA, 2003. ACM.

[11] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[12] Yunjin Lee, Seungyong Lee, Ariel Shamir, Daniel Cohen-Or, and Hans-Peter Seidel. Intelligent mesh scissoring using 3D snakes. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference*, pages 279–287, Washington, DC, USA, 2004. IEEE Computer Society.

[13] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In *In Eighth European Conference on Computer Vision (ECCV 2004)*, pages 377–389. Springer-Verlag, 2003.

[14] Alan P. Mangan and Ross T. Whitaker. Partitioning 3D surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, 1999.

[15] Michela Mortara, Giuseppe Patanè, Michela Spagnuolo, Bianca Falcidieno, and Jarek Rossignac. Blowing bubbles for multi-scale analysis and decomposition of triangle meshes. *Algorithmica*, 38(1):227–248, 2003.

[16] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *2003 Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, pages 27–32, June 2003.

[17] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 313–318, New York, NY, USA, 2003. ACM.

[18] Lior Shapira, Ariel Shamir, and Daniel Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24:249–259, 2008.

[19] Richard Szeliski. Image alignment and stitching: a tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):1–104, 2006.

[20] Matthew Uyttendaele, Ashley Eden, and Richard Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. In *2001 Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 509–516, December 2001.

[21] Dong Xu, Hongxin Zhang, Qing Wang, and Hujun Bao. Poisson shape interpolation. *Graph. Models*, 68(3):268–281, 2006.

[22] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 644–651, New York, NY, USA, 2004. ACM.