Eurographics Italian Chapter Conference (2010)
E. Puppo, A. Brogni, and L. De Floriani (Editors)

Tools and Applications for Teaching and Research in Computer Graphics

Fabio Guggeri, Marco Livesu, Riccardo Scateni

University of Cagliari, Dept. of Mathematics and Computer Science - Italy

Abstract

In this paper we present the work in progress along with some preliminary research results in the field of Computational Geometry and Mesh Processing obtained by the Computer Graphics Group of the University of Cagliari, Italy. We focus on the work in mesh analysis by introducing the developement of a lightweight visualization and processing tool that helped expanding the aims of the group by letting the students from the University move their first steps in Computer Graphics. We show some results obtained by the group with the focus on the usefulness of a common framework of reference.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

As the interest in Computer Graphics (CG) increased at the University of Cagliari, the need for tools that allowed to experiment and study the topic became stronger and stronger. While such tools, as MeshLab [CCR08] or OpenFlipper [ope], are free, versatile and offer a great number of possibilities for the development of mesh processing algorithms, their complexity makes them quite unfriendly to the newcomer, not allowing to easily lay hands on every aspect the term *Computer Graphics* covers; moreover, teaching can benefit from tools that allow a direct experimental approach, where the student has to deal with simple codes with everything in sight, with no interfaces or library calls that can obscure the whole functioning, thus allowing to endure a *trial and error* approach when modifying the program code to see if the results agree with what expected. Finally, the recent research activities carried on by the group focused on aspects that went outside the scope of the mere *Mesh Processing*; these reasons brought to the development of a simple and versatile family of applications where each student or researcher can focus directly on the goal of his/her work, from rendering to high-level data visualization, without having to deal with the complex infrastructures that advanced tools as the ones cited before are made of. We introduce this software family in section 2 along with some examples of its usage in teaching CG, showing experimental results in the field of mesh processing in Section 3. We conclude listing in Section 4 the upgrades currently under development and the desired additional features that are available for students that want to undertake a path in CG during their undergraduate carreers, along with a summary of the research projects that are carried on at the moment of writing.

2. A family of lightweight 3D viewers

The need to coordinate the works of many people and help students to get acquainted with Computer Graphics problems has brought to the developement of light, versatile and easily usable tools for visualization and processing. Gathered under the name of *CGView*, these projects form a continuously work-in-progress framework that helps the programmer to easily focus on each aspect of the problem, from accessing GPU buffers to the visualization of data. The CGView family is focused on two main goals: giving the student the possibility to view and explore the 3D environment with direct understanding of the code involved, and allowing a programmer to develop a mesh processing algorithm without having to deal with complex interfaces. The first goal brought to the father project, which we will be introducing in the next subsection; we will later explain in Subsection 2.2 the satellite projects that resulted from the need to satisfy our second goal.

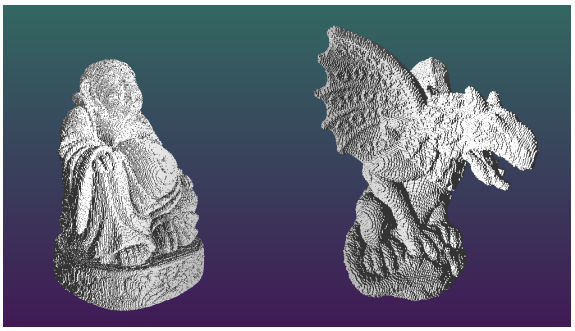

Figure 1: *High-resolution voxelizations of the Buddha and Gargoyle models*

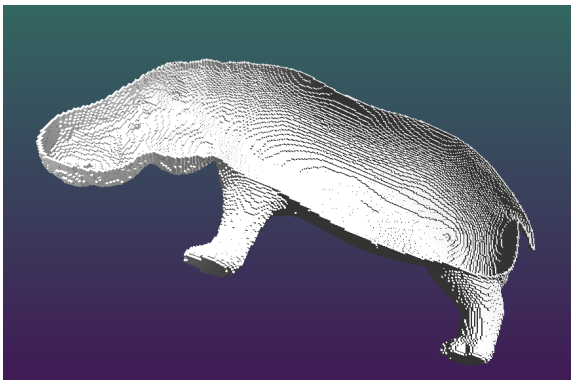

Figure 2: *Section of the Hippo model showing the interior of the voxelized surface*

2.1. The 3D object visualizer

Based on the VCG library for mesh processing [vcg] and the Qt Toolkit for its interface [qt], CGView is capable of opening and managing other kinds of data such as voxel grids or topological skeletons.

Mesh visualization

The application can visualize the mesh in every usual presentation as wireframe, points, flat or smooth shaded and can add materials into rendering. Moreover it can visualize all the main informations about the mesh, such as bounding box, axes (see figure 6), normals etcetera. However, as said before, the main goal of the application is to give the student the possibility to experiment and easily add its own work into the program, so the features are in constant upgrade.

As for mesh processing, the application takes advantage of the methods provided by the VCG library along with some added algorithms developed by researchers and students using one of the satellite applications we will describe in Subsection 2.2; we will also show in section 3 some research results obtained using our framework.

Figure 3: *Slice visualization for the Bunny model helps a visual understanding of the surface*

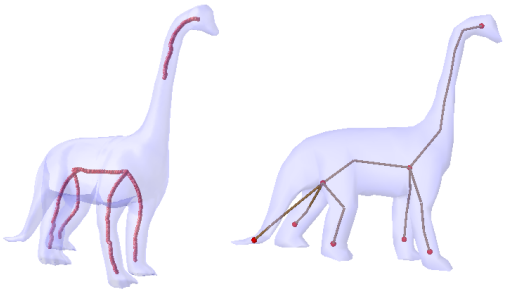

Figure 5: *Skeleton visualization using CGView. We show on the left the result of [CSYB05] and on the right the output from [ATC*08]*

Voxel management

Aside from mesh processing, CGView can load, manage and visualize voxel grids and can create mesh voxelizations of different resolutions, discretizing only the surface of the object or the interior according to some user-defined parameters. Figures 1 and 2 show these features along with the possibility to visualize only slices of the surface for mesh understanding purposes (Figure 3). The voxel grid data structure gives the programmer the methods for grid navigation and topology control, so the programmer has a simple and efficient tool for voxel processing and results visualization. Moreover the possibility of saving voxel grids in different formats can help testing other algorithms such as [CSYB05] or [GS99].

Topological skeletons

To facilitate works on skeleton extraction and analysis, CGView can manage skeletons obtained by algorithms like [ATC*08] and [CSYB05], helping the understanding and comparisons. In figure 5 we show an example of skeletons loaded from different algorithms, offering a way to visually compare the outputs. We will show in Section 3 some examples of the usefulness of CGView for the developement of skeleton extraction algorithms.

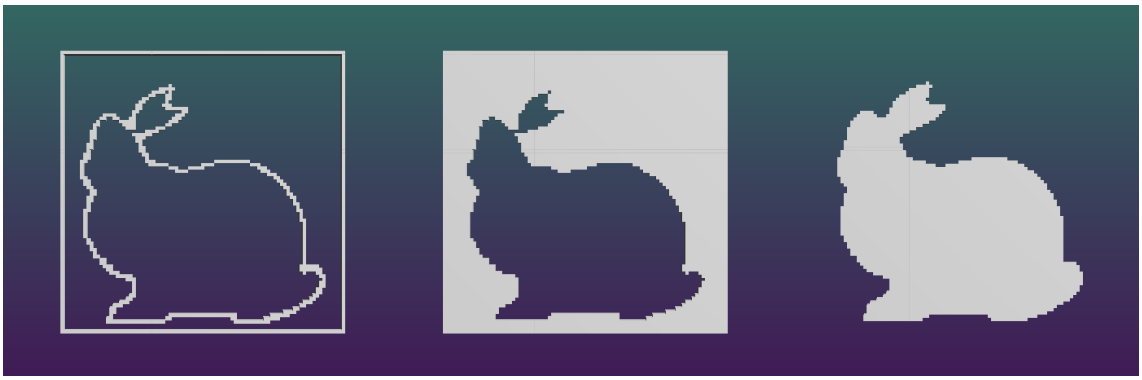

Figure 4: *Example of flood-filling for the discretization of the mesh interior. After the rasterization of the surface (left) the space is flood-filled from the exterior (center). The complement of this flood-filling joint with the surface forms the discretized interior (right)*

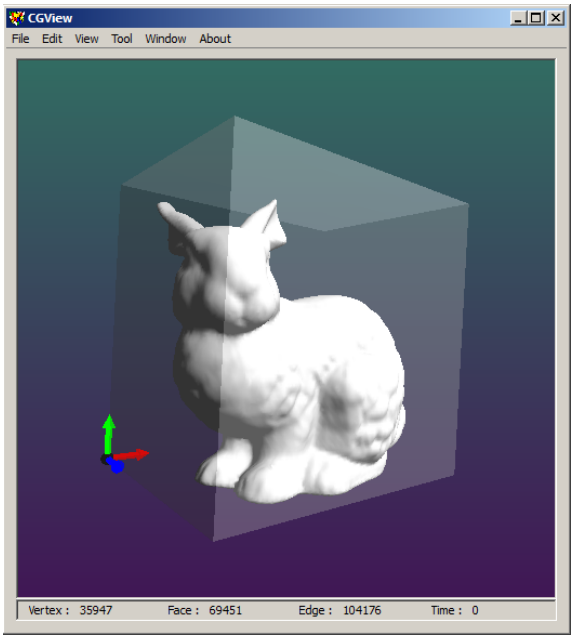

Figure 6: *The main window of CGView showing the mesh, its bounding box and the axes*

2.2. A template for mesh processing algorithms

The main CGView application is made up of several pieces developed by different people in the group. While adding features results in increasing the complexity of the code, thus contraddicting with the purpose of supplying the programmer with a simple framework, a basic version of CGView has been kept as a template for future developments. Each person interested in experimenting or programming a new feature for the main project is given a simple application that can only open, visualize and access the 3D mesh: by building a standalone application for each algorithm the prupose of simplifying the work of the programmer is satisfied. Moreover, its simplicity has helped the students to move their first steps in Computer Graphics: as the number of classes and lines of code is low, the student can take advantage of the direct correspondance between code and result, allowing to experiment with ease. As a consequence, the CGView family has grown to include all the side projects while the best, more stable and useful features of such projects end up being incorporated in the main viewer.

3. Mesh processing and analysis

Research on mesh processing has been carried on using the tools described in the previous section. We describe here some algorithms whose development was simplified thanks to the usage of the CGView template.

3.1. Skeleton extraction using GPU buffers

The idea, presented as a poster at SGP 2010 [GS], is focused on inferring a 3D curve-skeleton from the informations obtained by the 2D medial axis of each projection on the image plane (see Figure 7). We now give an overview of our method; due to the possible ambiguity of the term *skeleton*, from now on we'll refer to *medial axis* for the 2D case. We assume there is a correspondence between the skeleton and the medial axes of each projection such that each medial axis is an accurate representation of the projected skeleton; given this assumption we compute the medial axes of the projected mesh from many points of view. Each medial axis is then reprojected in 3D in order to gather informations in a voxel grid where each voxel stores a probability of being part of the skeleton. These values are interpreted as intensity values in a 3D grayscale image in order to perform an intensity-driven thinning algorithm to yield the final skeleton.

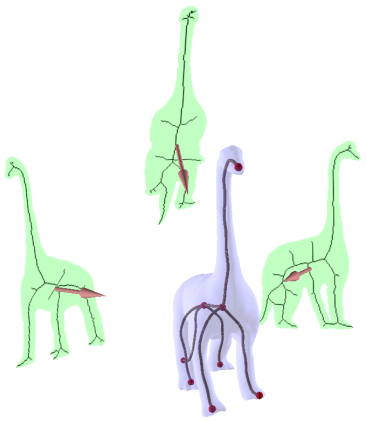

Figure 7: *An overview of the idea behind the skeleton extraction method described in Subsection 3.1. The 3D skeleton is obtained thanks to the informations of the medial axes of the shape projections*

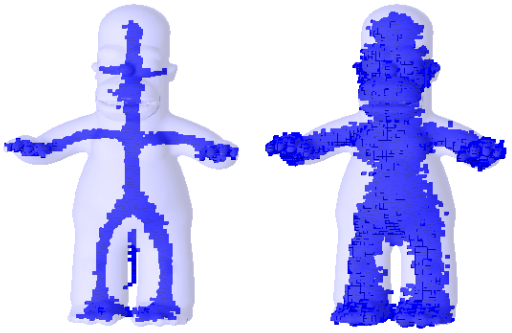

Figure 8: *CGView allows to visualize discretized scalar fields in voxel grids. In the image we show two different threshold for a voxel grid computed using the algorithm in Section 3.1*

Medial axis computation

The mesh is looked at from a set of evenly distributed points on a sphere so that the ambiguity of possible superimpositions is reduced. The mesh is rendered and its silhouette is extracted from the GPU buffers as a binary image. The medial axis is then computed using [SdB94] and pruned. This is mainly where having the possibility to easily access the GPU buffers comes in handy; in CGView is easy to lay hands on the rendering process so that one can extract the silhouette of the mesh with little overhead using OpenGL commands.

Voxel grid reprojection

Given the direction of view of each projection, each pixel in the medial axis casts a ray in the given direction. Such rays are then rasterized in a voxel grid in a voting fashion: each voxel value is incremented, stating that the voxel has now a higher probability of being part of the skeleton. The GPU depth buffer is used to rescrict the rasterizations to the interior of the mesh, resulting in a discrete scalar field computed on a voxelization of the mesh. Yet again, GPU access is simplified by the structure of CGView, while voxel visualization (see Figure 8) and accessing has been *borrowed* from the main project with no need for further coding.

Extraction

The grid is then considered as a grayscale image and a thinning algorithm is applied on it. The thinning is intensity-driven, in a way similar to the ORG algorithm described in [YCS00]: we first construct a tree representation of the connectivity of the voxels such that every path intensity is maximized, then the tree is pruned in order to remove low valued paths while preserving the topology and the thinness of our structure. Topological constraints are applied to the pruning process in order to consider loops that cannot be correctly described by an acyclic structure as a tree.

Post processing

The skeleton is refined during the final step of the algorithm: an internal pruning process collapses short internal arcs in order to merge branching nodes that are too near each other. An endpoint preserving laplacian smoothing is applied to the skeletal structure reducing the boundness of each node to the voxel grid coordinates; the arcs thus obtained are more visually appealing and more representative of the shape of the mesh.

3.2. Accelerated Shape Diameter Function

Another work developed with the help of CGView has been published under the name of *Accelerated Shape Diameter Function* [KGMS10]; it is basically a suggestion for a fast and efficient approximation of the *Shape Diameter Function* (SDF) [SSCO08] using the Poisson Equation for the interpolation of the values. 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. Given a mesh M the SDF is a scalar function on the surface ($f_p : M \to \mathbb{R}$) 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. The definition is extremely simple and intuitive; the value of the SDF is a weighted average of the ray lengths: the weights are the inverse of the angle between the ray to the center of the cone. The authors finally propose a bilateral filtering step in order to guarantee pose invariance.

Accelerated SDF (ASDF) via Poisson Interpolation

The main contribution of our optimization method is based on an observation of the behaviour 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. The difference in SDF between neighboring primitives 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 [YZX*04] and image processing [PGB03] under similar circumstances. This technique allows to easily compute a constrained interpolation over the mesh guaranteeing computational efficiency and robustness of the results.

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 $\mathbf{v}$, Δ is the Laplace operator and f^* defines the values of a known scalar function at the boundary $\partial\Omega$ of a selected region Ω. 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; our proposal, that resulted in good approximation of the divergence, was to use the opposite of the curvature on each face $curv(p)$ (computed as the mean of its vertices' curvature).

The final formula for each unknown face is:

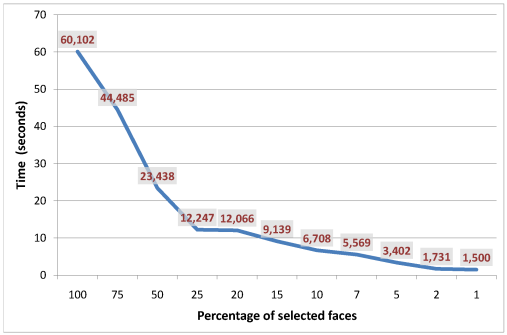

Figure 9: *Computational times in function of the number of selected faces. The times were taken while processing the horse mesh.*

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

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.

This technique resulted in a good and fast approximation, allowing to reduce the computational times with a small error. See Figure 9 for an example of the speedup obtained by our approach, while in Figure 10 we show the differences between an accurate and approximated SDF computation.

4. Future improvements and goals

As previously said, the CGView project is continuously under development due to the fact that research and teaching take advantage of the basic template for experimenting, while desired features for the main viewer are added once they are implemented by someone in the team. We list in this section a suggestion of possible improvements that could be implemented by students willing to dedicate their Bachelor or Master thesis to Computer Graphics, followed by some research goals the group hopes to reach in the immediate future.

4.1. Possible add-ons for the viewer

CGView is mainly a teaching and research tool, so possible future expansions should be focused on the knowledge and

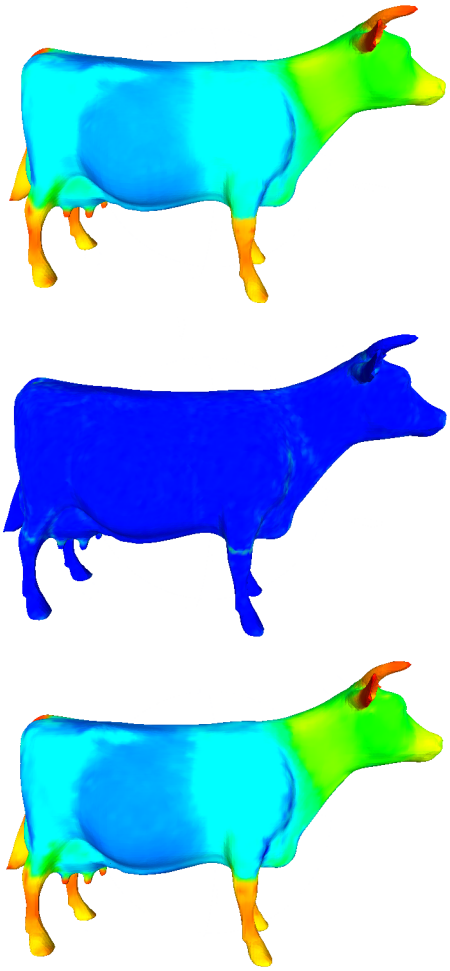

Figure 10: *In this picture you can visually appreciate the difference between the SDF computed on all the faces and a subset (10% of the total number). The image on top represents the result of the computation of the SDF on all the faces, while the one in the bottom represents the results of the computation 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.*

experience they bring to the programmer other than usefulness or fanciness, as more powerful and better programs are already available and by now CGView isn't going to compete with such tools. Anyway, the viewer lacks support for textures; as this is a central topic in Computer Graphics, the viewer would benefit from an implementation of texture mapping. On the interaction side, it is desirable to have an added support for a 3D navigation system like 3DConnexion ([3dc]), while mesh processing methods would benefit from

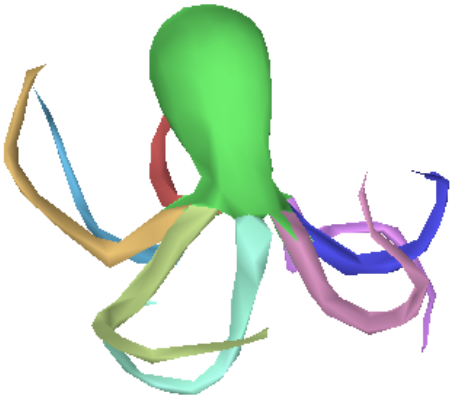

Figure 11: *Skeleton-driven segmentation on the Octopus model.*

different face/vertex selection methods that are missing, or the possibility to open and access more than one mesh at a time. Finally, on the rendering side, an idea would be the implementation of different shaders, e.g the Toon Shader.

4.2. Research aims

Currently the work of the CG group is focusing on improving the skeleton extraction method described in Subsection 3.1 and its uses in mesh partitioning; in Figure 11 some preliminary results in this field are shown. In parallel, the group is working on a project that started as a toy problem but ended up drawing more interest, that is, the application of *chaotic maps* to meshes (see Figure 12) for purposes of cryptography and steganography, as such methods proved useful for the same goals in the field of image processing ([Jia10], [WRJ*07], [ZXSH10]).

5. Conclusions

We presented the tools developed by the Computer Graphics group in Cagliari University and the usefulness they had in research and teaching. Since the CGView project was born many students could take advantage of a common framework for their tasks, improving interaction and knowledge sharing and thus helping to increment the productivity of the group. The CGView project is nowadays the base for every work in mesh processing that is being carried on by the research team at the University, and is currently being used as an experimentation tool in teaching Computer Graphics to the students.

References

[3dc] http://www.3dconnexion.com.

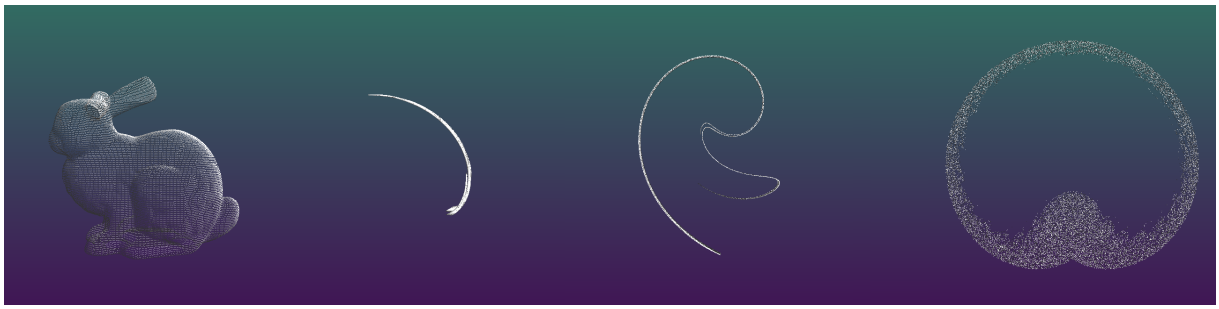

Figure 12: *From left to right, application of the Ikeda map to the points of the Bunny with increasing number of steps.*

[ATC*08] AU O. K.-C., TAI C.-L., CHU H.-K., COHEN-OR D., LEE T.-Y.: Skeleton extraction by mesh contraction. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–10.

[CCR08] CIGNONI P., CORSINI M., RANZUGLIA G.: Meshlab: an open-source 3d mesh processing system, April 2008.

[CSYB05] CORNEA N., SILVER D., YUAN X., BALASUBRAMANIAN R.: Computing hierarchical curve-skeletons of 3d objects. 945–955.

[GS] GUGGERI F., SCATENI R.: How to compute a mesh skeleton using projections in five simple steps. SGP 2010 Poster session.

[GS99] GAGVANI N., SILVER D.: Parameter-controlled volume thinning. *Graphical Models and Image Processing 61*, 3 (1999), 149 – 164.

[Jia10] JIA X.: Image encryption using the ikeda map. *Intelligent Computing and Cognitive Informatics, International Conference on 0* (2010), 455–458.

[KGMS10] KOVACIC M., GUGGERI F., MARRAS S., SCATENI R.: Fast approximation of the shape diameter function. In *GraVisMa 2010 Proceedings* (2010).

[ope] http://www.openflipper.org.

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Transactions on Graphics (SIGGRAPH'03) 22*, 3 (2003), 313–318.

[qt] http://qt.nokia.com.

[SdB94] SANNITI DI BAJA G.: Well-shaped, stable, and reversible skeletons from the (3,4)-distance transform. 107–115.

[SSCO08] SHAPIRA L., SHAMIR A., COHEN-OR D.: Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput. 24*, 4 (2008), 249–259.

[vcg] http://vcg.sourceforge.net.

[WRJ*07] WANG Y., REN G., JIANG J., ZHANG J., SUN L.: Image encryption method based on chaotic map. pp. 2558 –2560.

[YCS00] YIM P., CHOYKE P., SUMMERS R.: Gray-scale skeletonization of small vessels in magnetic resonance angiography. 568–576.

[YZX*04] YU Y., ZHOU K., XU D., SHI X., BAO H., GUO B., SHUM H.-Y.: Mesh editing with poisson-based gradient field manipulation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 644–651.

[ZXSH10] ZHANG Y., XIE J., SUN P., HUANG L.: A new image encryption algorithm based on arnold and coupled chaos maps. vol. 1, pp. 308 –311.