M4G: Manifolds for GPUs Library

André Maximo* Luiz Velho*

Abstract

The *Manifolds for GPUs* library – M4G – is a compendium of tools to build an atlas structure from a dense-polygon mesh to represent a surface in multiresolution in the GPU. The library is divided in two main parts: the first defines a halfedge-based mesh with stellar operators and $\sqrt{2}$-subdivision adaptivity; the second provides a set of applications to build a regular atlas-based representation of a given mesh. Both functionalities complement other publicly available libraries, placing the M4G library on an ideal position to construct a manifold-based representation of a mesh better suitable for GPUs.

1 Introduction

Recent advances in modeling, graphics and image processing allow us to envision a system where these three areas are joined together. The many mathematical theories underpinning the discrete geometry of meshes point to a convergence between geometry and image processing, while the fast pace of graphics-hardware processing power increasingly support this convergence. The main barrier for the convergence to happen is that the leading publicly available geometric softwares, such as *MeshLab* [3] and *ParaView* [6], and libraries, such as *OpenMesh* [5] and *CGAL* [7], do not yet combine all these different concepts.

A fundamental problem in geometry processing is the one of converting a surface representation form into another. This problem occurs whenever the surface data is generated or acquired in a representation that cannot be handled by the intended application, or the representation is not the more appropriate one. Here we present a library and a set of tools to convert from the traditional surface representation, namely a triangulated two-manifold mesh, to an atlas-based representation suitable for applications in graphics processing units (GPUs). The key features of our representation are two-fold. First, it allows adaptivity control of large and complex surface models. Second, it offers a "minimum" interface that should be general enough for supporting conversion from most representation forms. Both features were made possible by the combination of two well-established notions in geometry processing: an atlas and a mesh subdivision scheme.

An *atlas*, $\mathcal{A} = \{(U_i, \varphi_i)\}_{i \in I}$, on a surface $S \subset \mathbb{R}^3$ is a collection of *charts*, (U_i, φ_i), where U_i is a subset of S, called the *chart domain*, and $\varphi_i : U_i \to \varphi_i(U_i) \subseteq \mathbb{R}^2$ is a bijective map, called the *chart map*, that maps U_i onto a subset, $\varphi_i(U_i)$, of $\mathbb{R}^2$. The chart domains "cover" S, i.e., $S = \bigcup_{i \in I} U_i$, and two or more of them may overlap at the same point in S.

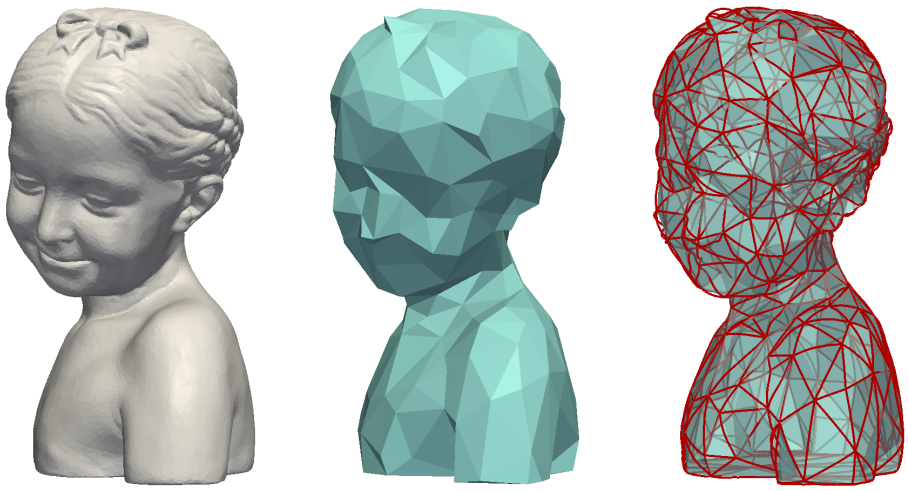
From a given atlas $\mathcal{A}$ on S, we build a polygonal mesh representation of S. Our construction is based on an adaptive subdivision scheme [22], which can be controlled by the CPU, and a dynamic tessellation, which can be done by the GPU, aiming at representing any mesh property with multiresolution. The resulting representation is a dynamic adaptive surface representation appropriate for GPU processing.

In this work we present a library dedicated to represent a discrete mesh in the computational perspective of GPUs. Our *Manifolds for GPUs* library – M4G – provides an adaptive representation of a mesh in the CPU, using charts in an atlas structure, that can be used to further tessellate regions of the same mesh in the GPU, on each chart interior. The M4G library also provides a number of useful applications to promote a GPU-based geometric system: different conversion algorithms from triangular to quadrangular (*tri2quad*) mesh based on recent articles [9, 21]; mesh simplification using CGAL [7] to create base meshes from dense-polygon meshes; fast geodesics curves computation [1] for each edge of a base mesh; mesh parameterization also using CGAL [7] to build chart domains for each face of a base mesh; and a samplify procedure to create regularly sampled charts based on triangle rasterization using ordinary scan conversion from OpenGL [4]. In summary our library is a compendium of source codes and applications to aid the development of a fully integrated geometric system.

*Institute of Pure and Applied Mathematics, {andmax,lvelho}@impa.br

2 Related Work

The concept of atlas generalizes the one of parameterization, and it has been used in the context of texture mapping [15], remeshing [17], and geometry encoding [18]. In all cases, the surface S was assumed to be represented by a polygonal mesh. Here, we employ the notion of atlas without assuming any particular representation form for S. Instead, we assume that the atlas $\mathcal{A}$ is described by a network of curves, e.g. geodesics, on S which defines the chart domain boundaries. Figure 1 details this concept in our library, a dense-polygon mesh is simplified to a base mesh and a set of geodesic curves are computed for each edge of the base mesh.



(a) (b) (c)

Figure 1: Example of the Bimba dataset, showing the original dense mesh (a), the base mesh (b) and the geodesics curves associated with the edges of the base mesh (c). The rendering was done using ParaView [6].

The adaptive multiresolution feature of the representation provided by our M4G library is particularly suited for progressive visualization [12]. The ability to separate the mesh into parts that must be sampled densely and parts that can be represented by coarse geometry is also useful for reducing the bandwidth required for transmitting a surface between different computational domains (e.g. hard-disk, main memory and GPU memory).

Perhaps the most powerful concept behind the manifold representation is that it enable us to work locally on a surface similarly to the two-dimensional Euclidean space. This is achieved through a parameterization, that establishes a mapping of the surface embedded in 3D to a region of the plane. The parameterization also defines a coordinate system on the surface. But since this mapping is essentially flattening a curved surface, inevitably it will cause geometric distortions. Parameterization methods try to reduce these distortions in order to preserve certain properties, such as angles, distances and areas [11]. Also, depending on how this mapping is computed, it can be designed to conform to a canonical region, such as a regular polygon (therefore constraining the boundary) or to leave the boundary free as an additional degree of freedom for distortion minimization [10]. One of the tools of our M4G library is a parameterization tool allowing the exploration of methods and border contraints to construct charts for a given mesh.

Surface parameterization also depends on the topology of the surface. Since it is, in general, different from the plane it is not possible to map the entire surface without cutting it open. In that respect, the parameterization methods can be classified into local and global, that respectively compute the mapping for small surface patches or the entire surface [17].

The atlas structure we consider relies on a network of curves on the surface. Based on the assumption that we can compute curves across chart domains, as well as subdivide their boundary curves, we can define an intrinsic, curve-based parametrization of each chart. This kind of parametrization is particularly useful because it naturally yields a chart map, and allows for a good control of both the map and the boundary of the region. An example of this concept is the *Multiresolution Adaptive Parameterization of Surfaces* (MAPS) [13]. In addition, it has been successfully used before for computing geodesic distances in the context of mesh parametrization [14] and remeshing [19]. Finally, the collection of individual charts, i.e. the domains and their associated maps, is a piecewise parametrization for the whole surface.

Independently of the surface representation, very often for compatibility reasons, it becomes necessary to convert to a polygonal approximation in order to perform certain tasks, such as visualization. In these situations, it is desirable to have a mechanism to produce an adaptive mesh. Examples of such strategies are the progressive meshes [12] and the stellar 4-k meshes [24]. Here we adopt the 4-8 adaptation scheme [23], which is a variant of the stellar 4-k mesh, in our base data structure inside our library.

3 The M4G Library

Our library is divided in eight repositories, publicly available in our institution gitorious server (`http://gitorious.impa.br/about`) under the GNU GPL 3 License [2]. The first repository contains our main contribution and it is called *adaptive 4-8 mesh* (*a48*) based on [25] and extending [23]. The a48 repository (at $ git clone `git://gitorious.impa.br/a48/a48.git`) hosts an efficient implementation of a polygon mesh data structure, using half-edge [16] (cf. class MeshT in figure 2); a powerful set of operators based on Stellar Theory, named stellar operators, for mesh manipulation (cf. class StellarMeshT); and an adaptive mesh representation with simplification and refinement strategies (cf. class AdaptiveMeshT). The next two repositories, called *mesh* (`/a48/mesh.git`[1]) and *stellar* (`/a48/stellar.git`[1]), are subsets of the first repository, hosting all codes related to MeshT and StellarMeshT, made available for teaching purposes. The top block of figure 2 summarizes our first repository structure (for a more in-depth documentation refer to `www.impa.br/~andmax/a48`).

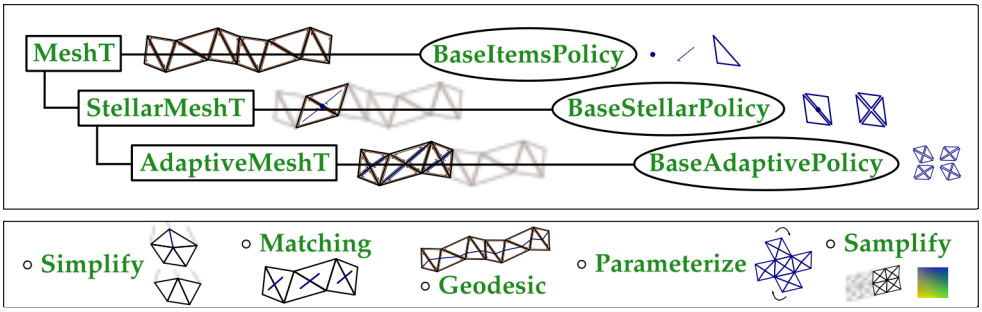


Figure 2: M4G library overview: (top) the classes and policies to manipulate and represent a meshed surface; (bottom) the applications toward a GPU-based geometric system.

The last five repositories host five different applications constructed to support a manifold-based representation of a mesh on the GPU. The first is called *simplify* (at $ git clone `git://gitorious.impa.br/m4g/simplify.git`) and contains a command-line software to expose CGAL's simplification process [7] to build base meshes from dense-polygon meshes (cf. application Simplify in figure 2). The second is called *matching* (`/m4g/matching.git`[1]) and uses Lemon [8] to implement a triangular-to-quadrangular application based on a 2-matching graph-based algorithm [9] on the triangular mesh dual (cf. Matching). Two other algorithms for triangular-to-quadrangular conversion [21, 25] can be used from the *tri2quad* application in the first a48 repository. These three tri2quad conversion algorithms were carefully implemented and return better and correct results where MeshLab [3] fails.

The next repository is called *geodesic* (`/m4g/geodesic.git`[1]) and contains another command-line software to expose geodesic curves computation [1] (cf. application Geodesic) using the implementation made available with the *Fast exact and approximate geodesics on meshes* algorithm [20]. Each geodesic is computed in parallel (using as many CPU cores as available) over a dense mesh connecting vertices of each corresponding base-mesh edge. The fourth repository is called *parameterize* (`/m4g/parameterize.git`[1]) and uses CGAL [7] to unveil several local parameterization algorithms with different border types (cf. application Parameterize). The last application repository is called *samplify* (`/m4g/samplify.git`[1]) and uses OpenGL [4] to rasterize a surface patch generating a regularly sampled chart from a parameter domain (cf. application Samplify). The bottom block of figure 2 summarizes our application repositories.

The repositories of the M4G library are separate on purpose to facilitate its different uses. Each repository is a module with its own dependancies, making it easier for a subset utilization (e.g. an input triangular mesh enhanced with stellar operations and tri2quad functions can be used to develop a new simplification process not possible using only CGAL [7]). All eight repositories have user instructions (in a read-me file) with simple usage examples and sample models.

[1]For the full repository include: `git://gitorious.impa.br` before each address.

Acknowledgment

We acknowledge the grant of the first author provided by Brazilian agency CNPq (National Counsel of Technological and Scientific Development).

References

[1] Geodesic. http://code.google.com/p/geodesic.

[2] GNU General Public License version 3. http://www.gnu.org/licenses/gpl.html.

[3] MeshLab. http://meshlab.sourceforge.net.

[4] OpenGL. http://www.opengl.org.

[5] OpenMesh. http://openmesh.org.

[6] ParaView. http://www.paraview.org.

[7] CGAL, Computational Geometry Algorithms Library. http://www.cgal.org.

[8] LEMON, Graph Library. http://lemon.cs.elte.hu.

[9] J. Daniels II, M. Lizier, M. Siqueira, C. T. Silva, and L. G. Nonato. Template-based Quadrilateral Meshing. *Computers & Graphics*, 35(3):471 – 482, 2011. http://dx.doi.org/10.1016/j.cag.2011.03.024.

[10] M. Desbrun. Processing Irregular Meshes. In *Proceedings of the Shape Modeling International 2002 (SMI'02)*, pages 157–, Washington, DC, USA, 2002. IEEE Computer Society. http://portal.acm.org/citation.cfm?id=882487.884134.

[11] M. Floater, K. Hormann, and M. Reimers. Parameterization of Manifold Triangulations. In *Approximation Theory X: Abstract and Classical Analysis*, pages 197–209. Citeseer, 2002. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.899.

[12] H. Hoppe. Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '96, pages 99–108, New York, NY, USA, 1996. ACM. http://doi.acm.org/10.1145/237170.237216.

[13] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, and D. Dobkin. MAPS: Multiresolution Adaptive Parameterization of Surfaces. In *Proc. ACM/SIGGRAPH Conf.*, pages 95–104, 1998. http://10.1145/280814.280828.

[14] H. Lee, Y. Tong, and M. Desbrun. Geodesics-based One-to-One Parameterization of 3D Triangle Meshes. *IEEE MultiMedia*, 12:27–33, January 2005. http://portal.acm.org/citation.cfm?id=1042196.1042327.

[15] J. Maillot, H. Yahia, and A. Verroust. Interactive Texture Mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 27–34, New York, NY, USA, 1993. ACM. http://doi.acm.org/10.1145/166117.166120.

[16] M. Mäntylä. *An introduction to solid modeling*. Computer Science Press, 1988. http://books.google.com.br/books?id=CJVRAAAAMAAJ.

[17] N. Ray, W. C. Li, B. Lévy, A. Sheffer, and P. Alliez. Periodic Global Parameterization. *ACM Trans. Graph.*, 25:1460–1485, October 2006. http://doi.acm.org/10.1145/1183287.1183297.

[18] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-Chart Geometry Images. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 146–155, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. http://portal.acm.org/citation.cfm?id=882370.882390.

[19] O. Sifri, A. Sheffer, and C. Gotsman. Geodesic-based Surface Remeshing. In *In Proc. 12th Intnl. Meshing Roundtable*, pages 189–199, 2003. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.76.1900.

[20] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. Gortler, and H. Hoppe. Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics (TOG)*, 24(3):553–560, 2005. http://dx.doi.org/10.1145/1186822.1073228.

[21] M. Tarini, N. Pietroni, P. Cignoni, D. Panozzo, and E. Puppo. Practical Quad Mesh Simplification. *Computer Graphics Forum (Special Issue of Eurographics 2010)*, 29(2):407–418, 2010. http://vcg.isti.cnr.it/Publications/2010/TPCPP10.

[22] L. Velho. Stellar Subdivision Grammars. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '03, pages 188–199, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. http://portal.acm.org/citation.cfm?id=882370.882396.

[23] L. Velho. A Dynamic Adaptive Mesh Library based on Stellar Operators. *Journal of Graphics Tools*, 9(2):1–29, 2004.

[24] L. Velho and J. Gomes. Variable Resolution 4-K Meshes. In *Computer Graphics and Image Processing, 2000. Proceedings XIII Brazilian Symposium on*, pages 123–130, 2000. http://10.1109/SIBGRA.2000.883904.

[25] L. Velho and D. Zorin. 4-8 Subdivision. *Computer Aided Geometric Design*, 18(5):397–427, 2001. http://dx.doi.org/10.1016/S0167-8396(01)00039-5.