Design, Implementation, and Evaluation of the Surface_mesh Data Structure

Daniel Sieger and Mario Botsch

Computer Graphics & Geometry Processing Group,
Bielefeld University, Germany
{dsieger,botsch}@techfak.uni-bielefeld.de

Summary. We present the design, implementation, and evaluation of an efficient and easy to use data structure for polygon surface meshes. The design choices that arise during development are systematically investigated and detailed reasons for choosing one alternative over another are given. We describe our implementation and compare it to other contemporary mesh data structures in terms of usability, computational performance, and memory consumption. Our evaluation demonstrates that our new Surface_mesh data structure is easier to use, offers higher performance, and consumes less memory than several state-of-the-art mesh data structures.

1 Introduction

Polygon meshes, or the more specialized triangle or quad meshes, are the standard discretization for two-manifold surfaces in 3D or solid structures in 2D. The design and implementation of mesh data structures therefore is of fundamental importance for research and development in as diverse fields as mesh generation and optimization, finite element analysis, computational geometry, computer graphics, and geometry processing.

Although the requirements on the mesh data structure vary from application to application, a generally useful and hence widely applicable data structure should be able to (i) represent vertices, edges, and triangular/quadrangular/polygonal faces, (ii) provide access to all incidence relations of these simplices, (iii) allow for modification of geometry (vertex positions) and topology (mesh connectivity), and (iv) allow to store any custom data with vertices, edges, and faces. In addition, the data structure should be easy to use, be computationally efficient, and have a low memory footprint.

Since it is hard to implement a mesh data structure that meets all these goals, many researchers and developers in both academia and industry rely on publicly available C++ libraries like CGAL [8] (computational geometry), Mesquite [6] (mesh optimization), and OpenMesh [5] (computer graphics).

However, even these highly successful data structures have their individual deficits and limitations, as we experienced during several years of research and teaching in geometry processing. In this paper we systematically derive the design choices for our new `Surface_mesh` data structure and provide an analysis and comparison to the widely used mesh data structures of CGAL, Mesquite, and OpenMesh. These comparisons demonstrate that `Surface_mesh` is easier to use than these implementations, while at the same time being superior in terms of computational performance and memory consumption.

2 Related Work

Due to their fundamental nature, a wide variety of data structures to represent polygon meshes have been proposed. Some are highly specialized to only represent a certain type of polygons, such as triangles or quadrilateral elements. Others are designed for specific applications, e.g. parallel processing of huge data sets. In general, mesh data structures can be classified as being either *face-based* or *edge-based*. We refer the reader to [18, 4] for a more comprehensive overview of mesh data structures for geometry processing.

In its most basic form a face-based data structure consists of a list of vertices and faces, where each face stores references to its defining vertices. However, such a simple representation does not provide efficient access to adjacency information of vertices or faces. Hence, many face-based approaches additionally store the neighboring faces of each face and/or the incident faces for each vertex. Examples for face-based mesh data structures include CGAL's 2D triangulation data structure [8], Shewchuck's Triangle [23], Mesquite [6], and VCGLib [29].

In contrast to face-based approaches, edge-based data structures store the main connectivity information in edges or halfedges [2, 13, 7, 20]. In general, edges store references to incident vertices/faces as well as neighboring edges. Kettner [18] gives a comparison of edge-based data structures and describes the design of CGAL's halfedge data structure. Botsch et al. [5] introduce OpenMesh, a halfedge-based data structure widely used in computer graphics. Alumbaugh and Jiao [1] describe a compact data structure for representing surface and volume meshes by halfedges and half-faces.

Furthermore, a fairly large number of publications describe more specialized mesh representations. For instance, Blandford et al. [3] introduce a compact and efficient representation of simplicial meshes containing triangles or tetrahedra. Other works focus on data structures for non-manifold meshes [10, 11], highly compact representations of static triangle meshes [14, 15], or mesh representations and databases for numerical simulation [12, 27, 22, 9].