Tripod: a minimalist data structure for embedded triangulations

Jack Snoeyink[*]   Bettina Speckmann[†]
UBC Computer Science
`[speckmann,snoeyink]@cs.ubc.ca`

Abstract

We show that a vertex-based data structure that keeps only 6 pointers per vertex can store triangulations, navigate them, and maintain them under swap operations. By comparison, edge-based structures such as the winged-edge take 18–24 pointers per vertex, and triangle-based structures take 12 pointers per vertex.

Introduction   Representations of planar subdivisions are central to applications of geometric computing. Triangulations are particularly important: In solid modelling, boundary representations are often composed of triangulations because graphics hardware is optimized for triangles and triangle strips; In finite element analysis, triangle meshes are fundamental; In geographic information systems, triangulated irregular meshes (TINs) model digital terrain data. Even planar subdivisions that are not triangulations may be represented as such: the Voronoi diagram, for example, is commonly represented through its dual Delaunay triangulation.

Edge-based structures, such as the winged-edge [1], are commonly used to support operations for embedded graphs. For each edge, these structures store pointers to the two incident vertices, the next edges clockwise (cw) and counter-clockwise (ccw) around these two vertices, and possibly pointers to faces, if there is data that must be associated with faces. *Face-based* (or triangle-based) structures store, for each triangle, pointers to the adjacent triangles and incident vertices. Since, by Euler's relation, the number of edges in a planar triangulation with n vertices is $3n - 6$, the edge-based structures take 18–24 pointers per vertex and face-based structures take 12 pointers per vertex.

We investigate a *vertex-based structure* suggested by Martin Heller of the University of Zurich. We show that this structure with only 6 pointers per vertex is well-defined, and that it can support maintenance operations (including insert and swap), although not in constant time. All winged-edge navigation operations are supported in constant time.

The Tripod structure   We define the tripod data structure for an embedded planar graph in which every face, including one distinguished *outer face*, is a triangle. We assign directions

to the edges such that the outer face is a cycle with the rest of the graph to the left, and every vertex not on the outer face has exactly three outgoing edges.

Schnyder [2] has shown that every planar triangulation has such a representation, and that the outgoing edges from a vertex v can be labeled 0, 1, and 2 in ccw order around v such that all incoming edges between i and $i+1$ are labelled $i+2$ for integers $0 \leq i \leq 2$ and addition modulo 3. See Figure 1. This labelling induces a partition of the edges into three sets that define three spanning trees, each including one vertex of the outer face and all vertices inside. For an edge e that is outgoing from v, we let e.inc and e.dec be the outgoing edges from v that correspond to incrementing and decrementing the label of e.
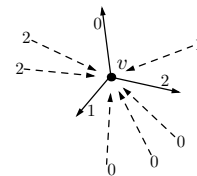

Figure 1: Labels

Each edge e that is outgoing from vertex v carries cw and ccw pointers, named e.ccw and e.cw. Let f be the next edge ccw (cw) from e around vertex v. The ccw pointer e.ccw (cw pointer e.cw) indicates the vertex that is the non-v endpoint of f. In the four examples in Figure 2, the circled vertex is e.ccw.

Let us make two remarks: First, note that an edge does not point to its own destination; that pointer will be recovered from edges incident on the same triangle. Second, there are two essentially different cases for the edge f: when f is incoming at v, then e.ccw points to the vertex that holds f, as one would expect, but when f is outgoing, then e.ccw and f.cw point to triangle vertices, only one of which will contribute the third edge to the triangle.
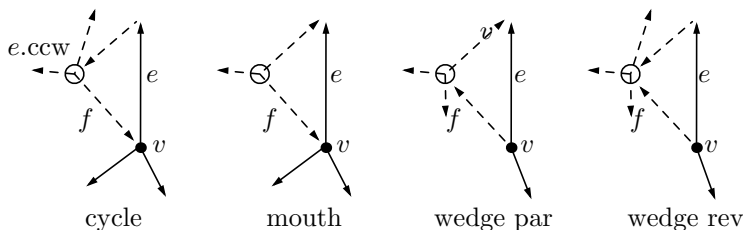

Figure 2: Cases for triangles left of edge e

Figure 2 illustrates four types of triangles that can occur to the left of an outgoing edge e. We observe that the type of triangle can be tested in constant time.

Theorem 1 *The four types of triangles of Figure 2 can be identified by the following tests:*

cycle *Triangle edges form a ccw cycle. Test by* e.ccw$^3 == e$.

mouth *Non-e triangle edges outgoing from same vertex. Test by* e.ccw.inc.cw $== e$.

wedgepar *Two edges out of v; third points in direction of e. Test by* e.ccw.cw.dec $== e$.

wedgerev *Two edges out of v; third in opposite of direction of e. Test by* e.inc.cw.ccw $== e$.

Proof: Verifying that each test accepts its type is easy. Showing that each fails for the other types is tedious, and is omitted in this abstract. ∎

Tests for the triangle right of e can be obtained by swapping .ccw with .cw and .inc with .dec.

2

The cases have corresponding Schnyder labels as in Figure 3, which allow us to identify the relevant edges out of the vertex e.`ccw`. An alternative is to have e.`ccw` point to a particular edge of a vertex; this can be done so that the labels are relative rather than absolute, which can make some manipulations more efficient.
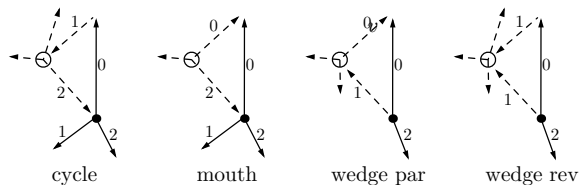

Figure 3: Labels for the cases

Operations   With this structure, an oriented edge can be identified by specifying its vertex, Schnyder label, and direction. By determining the type of an adjacent triangle, all standard navigation operations on triangulations, such as determining the next edge around a face or vertex, can be supported in $O(1)$ time.

Many maintenance operations can also be supported in $O(1)$ time: When a vertex v is inserted into a triangle, v is created with three new outgoing edges, and pointers are set after determining the type of the triangle. When an edge e, outgoing from v, is swapped and a neighboring triangle has an edge f incoming to v, then the roles of e and f can be changed while preserving three edges per vertex and Schnyder labels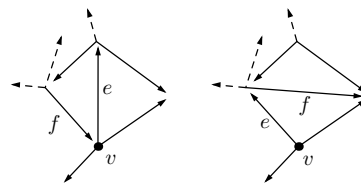. This is illustrated in Figure 4. There is one exception: when the triangles right and left of edge e are in the wedge cases, then a swap operation on e cannot proceed until some new edge is found that can be outgoing for v. We prove

Figure 4: Diagonal swap

Lemma 2 *There is a directed cycle of edges through v that does not include edge e.*
We can reverse the edges on this cycle to obtain the case where we can swap.

Evaluation   The Tripod data structure has one obvious advantage in its low memory requirements. A less obvious advantage has been suggested by Martin Heller: since it is based on vertices, which are the least ephemeral elements in most applications that use triangulation data structures. Thus, on can use static memory allocation, and, in large data sets, one can partition the structure based on spatial position for better locality of reference and organization of secondary storage.

The obvious disadvantages are increase in access time and some added programming complexity due to its parsimonious use of pointers.

References

[1] B. G. Baumgart. A polyhedron representation for computer vision. In *Proc. AFIPS Natl. Comput. Conf.*, volume 44, pages 589–596. AFIPS Press, Alrington, Va., 1975.

[2] W. Schnyder. Embedding planar graphs on the grid. In *Proc. 1st ACM-SIAM Sympos. Discrete Algorithms*, pages 138–148, 1990.