Eurographics Symposium on Geometry Processing 2008
Pierre Alliez and Szymon Rusinkiewicz
(Guest Editors)

Volume 27 (2008), Number 5

Motorcycle Graphs: Canonical Quad Mesh Partitioning

David Eppstein[†1], Michael T. Goodrich[†1], Ethan Kim[2], and Rasmus Tamstorf[3]

[1]Computer Science Department, University of California, Irvine
[2]School of Computer Science, McGill University
[3]Walt Disney Animation Studios

Abstract

We describe algorithms for canonically partitioning semi-regular quadrilateral meshes into structured submeshes, using an adaptation of the geometric motorcycle graph *of Eppstein and Erickson to quad meshes. Our partitions may be used to efficiently find isomorphisms between quad meshes. In addition, they may be used as a highly compressed representation of the original mesh. These partitions can be constructed in sublinear time from a list of the extraordinary vertices in a mesh. We also study the problem of further reducing the number of submeshes in our partitions—we prove that optimizing this number is NP-hard, but it can be efficiently approximated.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Boundary representations

1. Introduction

Quadrilateral meshes have many applications in computer graphics, surface modeling, and finite element simulation. The simplest quadrilateral meshes are *structured meshes*, in which connections between quadrilaterals form a regular grid, but in complicated domains it may be necessary to use *semi-regular meshes* in which this structure is interrupted by a small number of *extraordinary vertices* that do not have degree four. We study how to partition a semi-regular mesh into a small number of structured submeshes. This problem is motivated by the following computer graphics applications:

Mesh compression. A partition into structured submeshes forms a compressed representation of a mesh that may provide substantial space savings. There is a large literature on geometry compression, but it has largely studied unstructured triangle meshes [Dee95, TR98, TG98, GS98, COLR99]. King et al. [KRS99] compress highly irregular quadrilateral meshes by splitting each quadrilateral into two triangles and then compressing the resulting triangle mesh; they do not take advantage of any regular structure that may be present. We show that semi-regular meshes may be compressed to a size proportional to the small number of extraordinary vertices.

Our work directly concerns only the mesh topology, but may also be of use in compressing mesh geometry. The *geometry images* technique [GGH02], which represents the x, y, and z coordinates of vertices as the R, G, and B channels of a color image and applies standard image compression methods to this image, requires a partition of the model into structured submeshes. Gu et al. handle this step by remeshing, while we find such a partition without changing the mesh. Alternatively in [TR98], positions of already-encoded nearby vertices are combined to predict the position of each new vertex, and only a small correction term need be encoded using an entropy-based encoder. We expect the regular vertex neighborhoods provided by our partition into structured submeshes to help simplify the predictors for this process.

Mesh isomorphism. In feature film production many characters may all be created from the same base topology; Figure 1 shows just a few different characters out of about 70 that all use the same mesh topology. This shared structure allows for significant savings on modeling, rigging, and texture painting if the work done on one character (including blendshape targets, corrective shapes for pose space deformation, weighting assignments for rigs, and UV coordinates for texture mapping) may be transferred to all the others using the one-to-one correspondence between the models. Given that all the characters use the

[†] This work was done while Profs. Eppstein and Goodrich were consultants to Walt Disney Animation Studios.

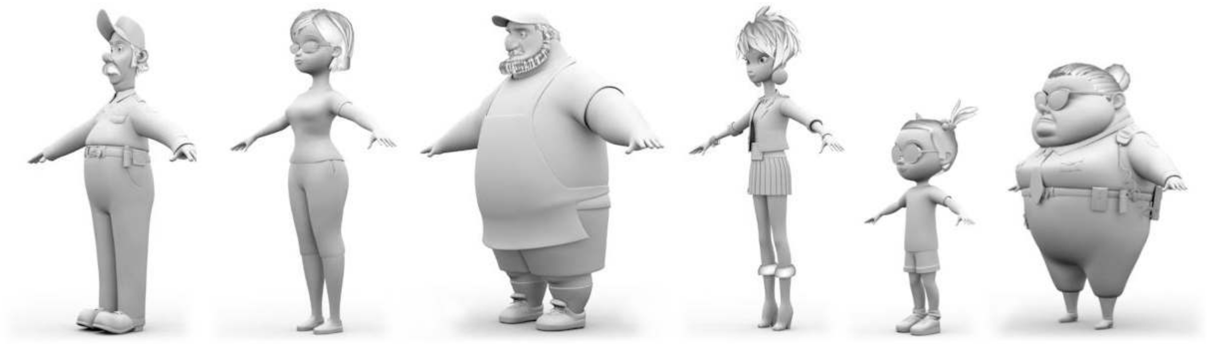
Figure 1: *Several animated characters from a set of about 70 in which the characters' bodies have isomorphic meshes.*

same topology such a correspondence clearly exists, but unfortunately it is not always readily available.

For efficiency reasons, applications typically assign contiguous ranges of ids to vertices, faces, and edges and store the related information in contiguous blocks of memory, renumbering objects when necessary to keep them contiguous. Even a simple operation of breaking a model into multiple pieces and then reassembling the parts can cause all vertices, edges, and faces to be renumbered unpredictably. To recover from this problem we must reestablish the isomorphism between two meshes. For models of bounded genus, isomorphisms may be found in polynomial time [FM80, Gro00, HW74, Mil80], but these algorithms are complex and (except for planar models) superlinear in their runtime. By applying graph isomorphism algorithms to a compressed partition of the model rather than to the uncompressed input mesh, we may speed them up without sacrificing accuracy or robustness. To ensure that isomorphisms will always be found when they exist, our partition must be *canonical*: the same mesh must always generate the same partition. The partitioning algorithm we describe has this property.

Texture mapping. A partition into submeshes provides a convenient framework for describing the correspondence between a two-dimensional texture and the three-dimensional model onto which it is mapped: one can store a rectangular texture image for each submesh, and map each structured submesh regularly onto its bitmap. A partition with few submeshes minimizes the overhead associated with the bitmap objects and reduces visual artifacts at the seams between submeshes.

Along with the isomorphism problem treated here, it is often important to find partial matches between meshes that are not completely isomorphic. For example, if one scene is formed from another by removing a model's head and reattaching it in a combinatorially different twisted position, the models will not be isomorphic as a whole, but we would still like to find separate isomorphisms between the heads and the bodies. The method we describe here is not directly suitable for this task, because small changes to a model may propagate and cause large changes to our partition, but in [EGKT08] we describe applications of similar mesh decomposition techniques to the problem of finding large shared submeshes between two similar but non-isomorphic models.

Additionally, these techniques may apply to areas beyond graphics such as scientific computation. Code for the finite element method can be greatly streamlined when applied to structured quadrilateral meshes. By partitioning unstructured meshes into structured submeshes, it should be possible to achieve similar speedups for semi-regular meshes.

With these motivations, we provide the following results:

- We show how to generate a partition that is "canonical", in the sense that it depends only on the connectivity of the initial mesh. Such a canonical partition is of particular interest in mesh isomorphism, as it allows us to find isomorphisms in time depending on the compressed size of the mesh rather than on its overall number of elements.
- We present data showing that on meshes from animation applications the canonical partition substantially reduces storage size compared to the initial unpartitioned mesh.
- We show that, if we are given an initial list of extraordinary vertices in a mesh, we can construct our canonical partition in time proportional to the number of edges in the partition, which may be substantially smaller than the number of edges in the entire input mesh.
- We consider problems of minimizing the size of the compressed mesh representations. Many problems of this type are NP-complete but, as we show, our canonical partition approximates them to within a constant factor.
- We find heuristics for reducing the size of compressed representations beyond the size of our canonical partition.

2. Definitions

We define an *abstract quadrilateral mesh* to be a structure (V, E, Q) where V is a set of *vertices*, E is a set of *edges*, and Q is a set of *quadrilaterals*, with the following properties,

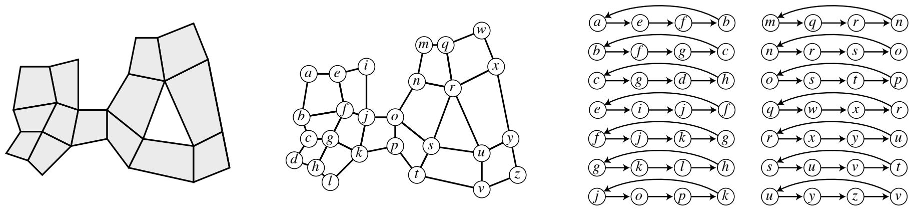

Figure 2: *An example of an abstract quadrilateral mesh. On left is a graphical view of the overall mesh, at center the graph (V,E), and at right the cycles in Q.*

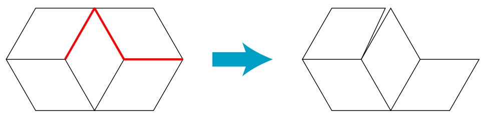

Figure 3: *Cutting a mesh along a set of edges produces a submesh.*

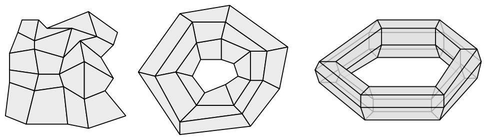

Figure 4: *The three possible topologies of a structured mesh: disk, annulus, or torus.*

which encapsulate the requirement that the mesh must form an orientable manifold:

- (V,E) forms a simple connected undirected graph.
- Each quadrilateral in Q consists of an oriented cycle of four edges in E.
- Each edge in E belongs to one or two quadrilaterals in Q.
- Any two quadrilaterals in Q share a single edge, share a single vertex, or do not share any features.
- If two quadrilaterals share an edge, it has opposite orientations in the two quadrilaterals.
- If two quadrilaterals q and q' share a single vertex v, there is a sequence of quadrilaterals $q = q_0, q_1, \ldots q_k = q'$ such that each two consecutive quadrilaterals q_i, q_{i+1} share an edge that has v as its endpoint.

The *boundary* of a quadrilateral mesh consists of all edges that belong to exactly one quadrilateral, and all vertices incident to an edge of this type. A *submesh* $M' = (V', E', Q')$ of mesh $M = (V,E,Q)$ is formed by a one-to-one mapping from Q' to Q such that any two quadrilaterals of Q' that share an edge correspond to quadrilaterals that share an edge in Q, and any two quadrilaterals of Q' that share a vertex correspond to quadrilaterals that share an edge or a vertex in Q. A submesh can be formed by cutting apart Q along boundary edges of Q' and keeping a connected component of the result (Figure 3). An *ordinary vertex* of a mesh is a non-boundary vertex incident with four edges or a boundary vertex incident with at most three edges; any other vertex is *extraordinary*. A mesh is *structured* if it has no extraordinary vertices, and *unstructured* otherwise. The main subjects of this paper are *structured partitions* in which the quadrilaterals of a mesh are partitioned into a small number of structured submeshes.

3. Classification of structured meshes

As we now show, meshes without extraordinary vertices must have a very specific structure.

Define an *(a,b)-grid* to be the structured mesh of unit squares in the rectangle $\{(x,y) \mid 0 \le x \le a \text{ and } 0 \le y \le b\}$. Two meshes are *isomorphic* if their vertices, edges, and quadrilaterals may be placed in a one-to-one incidence-preserving correspondence; the mesh in the far left of Figure 4 is isomorphic to a $(4,5)$-grid. The *Euler characteristic* of mesh $M = (V,E,Q)$ is $\chi(M) = |V| - |E| + |Q|$. The *deficiency* $D(v)$ of vertex v that is incident to d edges is $4-d$ if v is a non-boundary vertex, or $3-d$ if v is a boundary vertex.

Lemma 1 $\chi(M) = \frac{1}{4}\sum_{v \in V} D(v)$.

Proof There are four quadrilateral-edge incidences per quadrilateral, two per non-boundary edge, and one per boundary edge, so $2|E| - B = 4|Q|$ where B is the number of boundary edges. Similarly, counting vertex-edge incidences per edge and per vertex, $2|E| = \sum_{v \in V} d(v) = 4|V| - B - \sum_{v \in V} D(v)$. Adding these two equalities and dividing by four produces the result. ☐

Lemma 2 Any structured mesh M is homeomorphic to a disk, an annulus, or a torus.

Proof An ordinary vertex has non-negative deficiency, so a structured mesh has $\chi(M) \ge 0$. Any 2-manifold may be formed from a sphere by adding k holes and h handles for some (k,h); its Euler characteristic is $2-2k-h$. The only combinations that result in a nonnegative Euler characteristic are the sphere ($k = h = 0$), disk ($k = 1$ and $h = 0$), annulus ($k = 2$ and $h = 0$), and torus ($k = 0$ and $h = 1$). However, the sphere has no structured mesh: by Lemma 1, it has at least one vertex with positive deficiency, but a structured mesh with no boundary has zero deficiency. ☐

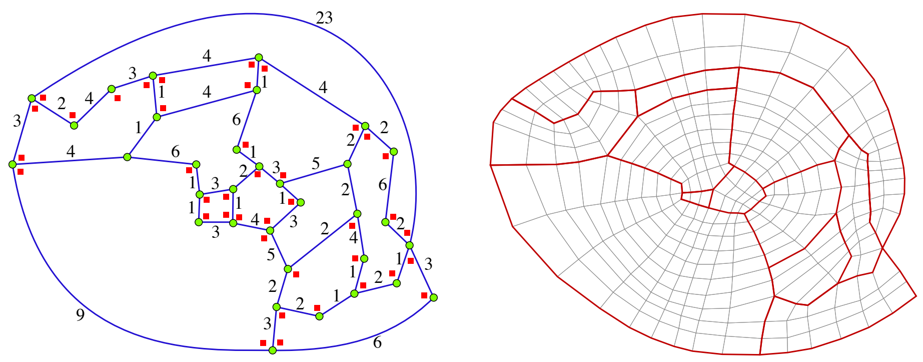

Figure 5: *Left: A schematic partition. The red squares indicate vertex-face incidences that are marked as corners. Right: A quadrilateral mesh and structured partition corresponding to this schematic partition.*

Lemma 3 Any structured mesh homeomorphic to a disk is isomorphic to an (a,b)-grid for some a and b.

Proof We use induction on the number of quadrilaterals in the mesh M. By Lemma 1 M has exactly four degree-two vertices. Let ℓ denote the shortest distance along the boundary between any two degree-two vertices, and let S be the submesh of M consisting of the quadrilaterals along the boundary between the two closest vertices (breaking ties arbitrarily). From the regularity of the vertices, S consists of a strip of quadrilaterals in the form of a $(1,\ell)$-grid. If S is the entire mesh, the result holds directly. Otherwise, removing S from M leaves a smaller submesh, which must be isomorphic to a (k,ℓ)-grid for some k by the induction hypothesis; adding S back produces a $(k+1,\ell)$-grid. Thus, we may set $a=k+1$ and $b=\ell$. □

4. Representing a structured partition

We define in this section the *schematic partition*, a compressed combinatorial description of a partition into structured submeshes that allows us to reconstruct without loss the original mesh. Intuitively, this partition is a graph that has as its vertices the corners of the structured submeshes and as its edges the paths along the submesh boundaries, labeled with the number of edges on each path. The corner of one submesh may lie along the side of another submesh, so we also need information within each face of the graph (which we represent as marks on certain vertex-face incidences) describing where to place the submesh's corners. More formally, we define a schematic partition to be a multigraph embedded on a 2-manifold without boundary, such that:

- Each edge is marked with a *length*.
- Some vertex-face incidences are marked as *corners*.
- Each vertex has at least two incident edges. A vertex with exactly two incident edges has one of its vertex-face incidences marked as a corner.
- Each face of the embedding is topologically a disk, and has either zero or four corners.
- At least one face incident to each edge has four corners.
- If a face has four corners, then these corners partition the boundary of the face into four paths, such that the two paths in each opposite pair of paths have the same length.

It is convenient to represent schematic partitions using the *winged edge* data structure [Bau72], augmented with lengths and marks; our actual implementation uses the Boost graph library [SLL02] with an additional layer representing the embedding. If mesh M is partitioned into grids, we may represent it by a schematic partition: Let X be the vertices with nonzero deficiency in M or in at least one submesh, and let Y be the edges on the boundary of some submesh. Then Y forms a family of paths with vertices in X as endpoints; we form a schematic partition with one edge per path, labeled by the length of the path. Conversely, from a schematic partition G formed in this way from a mesh M, we can form a mesh isomorphic to M by replacing each marked face of G by an (a,b)-grid, where a and b are the edge lengths of the face. Thus a schematic partition forms a compressed representation requiring space proportional to the number of edges in G rather than to the number of vertices, edges, and quadrilaterals in M. For instance, the schematic partition in Figure 5 can be represented by a winged edge structure with 44 edge objects, compared to the full quadrilateral mesh which has 323 vertices, 622 edges, and 300 quadrilaterals.

5. The motorcycle graph

We now describe a technique for finding a partition of an abstract quadrilateral mesh. Our approach is based on the *motorcycle graph*, a construction inspired by a video game in the 1982 Disney movie *Tron* (Figure 6) and previously used in algorithms for constructing straight skeletons [EE99, CV07]. In the Disney movie, players ride "light cycles" which move horizontally and vertically within a playing field, leaving paths behind them that are visible as glowing walls that form obstacles for later play. The motorcycle graph for a system of particles in the plane is formed

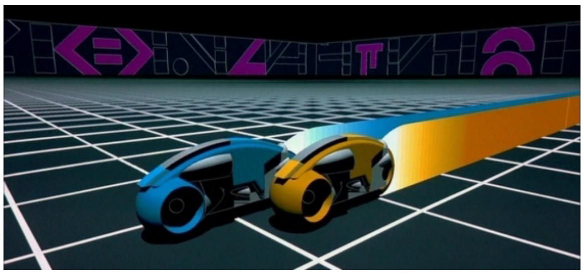
Figure 6: *"Light cycles" from the 1982 movie* Tron.

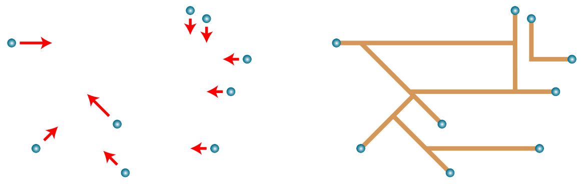
Figure 8: *An initial system of particles and velocities in the plane (left) and the resulting motorcycle graph (right).*

by moving each particle in a straight line at its initial velocity, stopping particles that meet other particles' tracks. The paths traced out by the particles in this system form a *pseudoforest* (a graph in which each connected component has at most one cycle) called the motorcycle graph (Figure 8).

We may similarly define a motorcycle graph for a quadrilateral mesh, as follows. Begin by placing a particle on each edge that is incident to an extraordinary vertex, moving outwards from that vertex. Then, in a sequence of time steps, move each particle along the edge on which it is placed. When a particle reaches an ordinary interior vertex it moves in the next time step to the opposite edge at that vertex. However, when two oppositely-traveling particles meet, when a particle meets a vertex that has previously been traversed by itself or another particle, or when a particle meets a boundary vertex of the mesh, it stops. When three or four particles meet simultaneously at a vertex, they all stop. However, when exactly two particles meet perpendicularly at a vertex, we use the "right hand rule": the particle clockwise from the right angle formed by the two particles' tracks stops, while the other particle keeps going. The *motorcycle graph* of the mesh is the set of edges traversed by particles as part of this process, together with all boundary edges of the mesh. Figure 7 shows an example of motorcycle graph construction.

The motorcycle graph partitions the mesh into regions with no extraordinary vertices, so it forms a structured partition of M. If M is not itself already structured, each mesh in this partition is topologically a disk: no torus can be part of a partition of a larger unstructured mesh, and no annulus may be split off from a larger mesh by the track of one of the particles unless the annulus has an extraordinary vertex v on its boundary, in which case the annulus would have been further partitioned by another particle from v.

Theorem 1 For bounded genus meshes, the numbers of vertices, edges, and faces of the schematic partition for the motorcycle graph are within a constant factor of the minimum possible for any schematic partition of M.

Proof Let M have n extraordinary vertices. Each vertex of the schematic partition of the motorcycle graph is an extraordinary vertex or a point where a particle stops, and the number of particles is four times the number of extraordinary vertices plus the sum of the deficiencies, so by Lemma 1 the partition has at most $5n - 4\chi(M)$ vertices. However, any schematic partition for M must include every extraordinary vertex, so the number of vertices in the schematic partition of the motorcycle graph is at most five times optimal plus a constant. Similarly, the number of edges in the schematic partition of the motorcycle graph is at most $8n - 8\chi(M)$ (each particle creates one edge when it starts and another when it stops) while any schematic partition must have at least n, so the number of edges is at most eight times optimal plus a constant. Finally, the number of faces of the schematic partition of the motorcycle graph is at most $3n - 3\chi(M)$ (each particle has one corner of a face clockwise of it where it starts, and creates two corners of faces when it stops, and each face has four corners) while, in an optimal partition, there must be at least $(n - \chi(M))/8$ faces (there must be $(n - \chi)/2$ vertices with negative deficiency to balance out the other vertices with positive deficiency, and each must be one of the four corners for some face), so the number of faces is at most 24 times optimal plus a constant. ☐

If the mesh M is given together with a list of its extraordinary vertices, in such a way that we can quickly look up the adjacency structure of M at any of its vertices, then we may construct the motorcycle graph in time proportional to the total number of edges traced out by the particles in the motorcycle graph; this time may be substantially smaller than the size of the input mesh. To do so, create a hash table H of vertices that have already been reached by a particle in the motorcycle graph, and simulate the motion of the particles that form the motorcycle graph one step at a time. Within each step, use H to determine whether any particle should stop because of reaching a vertex that another particle has already passed through, or whether any two or more particles meet at a common vertex.

6. Mesh isomorphism

One of our main motivating applications was finding isomorphisms of meshes. Because the process described above treats all vertices equivalently, and contains no nondeterministic steps, the motorcycle graph is determined uniquely by the connectivity of the mesh, and isomorphic meshes will get isomorphic motorcycle graphs. Therefore, we may test isomorphism between quadrilateral meshes on bounded-genus surfaces by representing their motorcycle graphs as

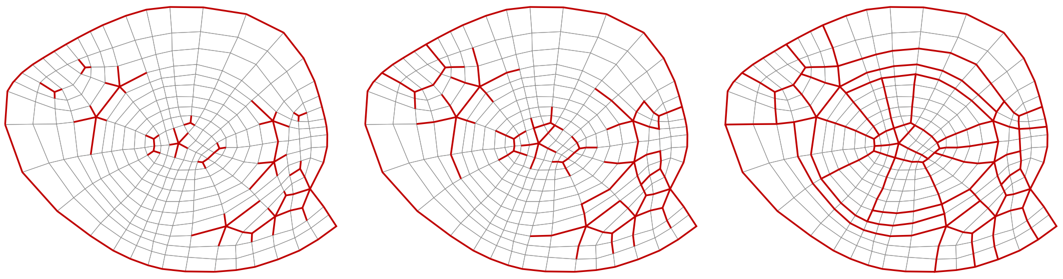

Figure 7: *Left: The construction of the motorcycle graph, after one time step. Particles emanating from each extraordinary vertex have moved one edge away from their initial positions. Center: The construction after two time steps; some particles have collided with each other. Right: The completed motorcycle graph.*

schematic partitions and applying a graph isomorphism algorithm that respects the surface embedding and edge and corner labels of the schematic partition. Graph isomorphism is most difficult when vertices have local neighborhoods that are similar to each other, causing the graph isomorphism algorithm to have to search farther to find the distinct features that identify the correct correspondence between the vertices of isomorphic meshes. In a semi-regular mesh, the ordinary vertices all have similar neighborhoods to each other, and even the extraordinary vertices may be surrounded by similar patches of ordinary vertices, making it more difficult to find the isomorphism between meshes. In our schematic partition, only the irregular features of the partition remain. By reducing a mesh to its salient features, this representation simplifies the task of the isomorphism algorithm.

Our task is simplified, relative to much work on bounded-genus graph isomorphism (e.g. [FM80, Gro00, Mil80]), by the fact that we already have a surface embedding for our meshes and for the schematic partitions of their motorcycle graphs. A simple and effective algorithm for isomorphism of embedded graphs is to choose arbitrarily a *flag* within one of the schematic partitions (an incident triple of a vertex, edge, and face), and test all equally-labeled flags of the other schematic partition for an isomorphism that maps the two flags into each other. Each test may be performed in linear time by performing parallel depth first searches starting from the two flags and verifying that the two searches remain synchronized and encounter the same sequences of adjacencies and labels [Mil80]. In the worst case, this algorithm takes time quadratic in the size of the schematic partitions; however, in practice it is likely faster because for most starting pairs of flags a mismatch will be detected quickly. [HW74] showed that, for planar graphs, a faster but more complicated linear-time isomorphism algorithm is possible, based on performing a sequence of reductions on the graphs that simplify them while preserving isomorphisms.

Thus, for planar quadrilateral meshes, we can compute isomorphism in time linear in the size of the schematic partition, while for bounded-genus meshes we can compute isomorphism in time quadratic in the size of the schematic partition. Although our result does not always provide an improvement in asymptotic complexity over direct application of graph isomorphism algorithms to the original meshes, it is a practical improvement, in several ways: First, the only part of the algorithm in which the uncompressed representation of the input mesh is handled, the construction of the motorcycle graph, is much simpler than the Hopcroft-Wong algorithm, and so likely has much smaller constant factors in its linear runtime. Second, for nonplanar surfaces, any compression in the schematic partition translates directly to an improvement in the size from which the quadratic runtime is determined. Third, if the extraordinary vertices are known, we may perform the whole algorithm in time sublinear in the input size. And fourth, the motorcycle graph construction need be performed only once for each mesh, allowing subsequent isomorphism tests to be performed efficiently.

7. Smaller partitions

In some applications of structured partitions, we may be prepared to spend a greater amount of preprocessing time, and sacrifice canonicalness of the resulting partition, in exchange for greater compression. We may make several observations related to minimizing the number of vertices in any schematic partition for a mesh M. First, each extraordinary vertex of the input mesh must be a vertex of the schematic partition. Second, at each extraordinary vertex, edges of the schematic partition must follow paths that use at least every other outgoing edge; for, if two consecutive edges at the vertex remained unused, the result would be an extraordinary boundary vertex of degree four or greater in some submesh of the partition. Third, the motorcycle graph construction may be modified to produce a valid structured partition by allowing particles to travel at different velocities or with different starting times; the simultaneous movement of the particles is important in making the motorcycle graph canonical but not in generating a correct partition. And fourth, the schematic partition formed from a motorcycle graph includes as a vertex an ordinary vertex v of M

only if some particle reaches v after some other particle has already reached it, or if two particles reach v simultaneously.

Thus, we may often find a smaller partition than the motorcycle graph itself by a process in which we build up the partition by adding a single path at a time, at each step starting from an extraordinary vertex and extending a path from it until it hits either another extraordinary vertex or an ordinary vertex that has previously been included in one of the paths. In this process, we should give priority first to paths that extend from one extraordinary vertex to another, because these paths cannot cause us to add any additional vertices to our partition. Secondly, we should prefer paths the initial edge of which is an even number of positions from some other edge around the same extraordinary vertex, in order to use as few paths emanating from that vertex as possible. Once no two consecutive edges at an extraordinary vertex remain unused, the partition process may terminate with a valid partition. The partition in Figure 5, for instance, may be constructed by a process of this type, and is significantly simpler than the motorcycle graph partition of the same mesh in Figure 7.

An alternative approach to reducing the complexity of partitions into structured submeshes would be, instead, to form the motorcycle graph, and then to repeatedly merge pairs of structured submeshes the union of which is still structured. For instance, the motorcycle graph in Figure 7 contains many mergeable pairs of submeshes. This approach would not be able to find certain partitions, for instance those in which some instances of the right hand rule have been replaced by a symmetric left hand rule, so it may be less effective at finding small partitions, but it would have the advantage of working within the compressed mesh after an initial motorcycle graph construction phase, and therefore could likely be implemented to run more efficiently than the careful selection of paths described above.

Some special cases of finding optimal partitions may be solvable in polynomial time. Ohtsuki [Oht82] shows that a polygon in the plane with horizontal and vertical sides (possibly with holes) may be partitioned into a minimum number of rectangles, in polynomial time; the same algorithm may be adapted to partition of a mesh of axis-aligned rectangles into a minimum number of structured submeshes. However this method does not minimize the number of vertices and edges of a schematic partition, and applies only to a very restricted subset of quadrilateral meshes.

8. Computational complexity

To show that the problem of finding optimal structured partitions is hard, we model it as a decision problem: for a given semi-regular mesh, and a given numerical parameter k, we may ask whether there is a partition into k structured submeshes, a partition corresponding to a schematic partition with k vertices, or a partition corresponding to a schematic partition with k edges. We show that all three variants of

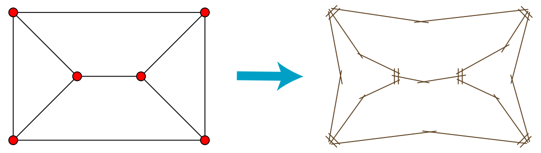

Figure 9: *Reduction from cubic planar independent set to crossing-free segments.*

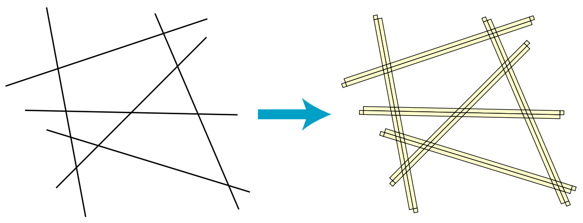

Figure 10: *Reduction from line segment arrangement to quadrilateral mesh.*

the problem are NP-complete by a reduction from maximum independent sets in cubic planar graphs [GJ79, GJ77] via an intermediate problem, finding k non-crossing line segments from a given set of n line segments with the restriction that, if any two input line segments intersect, they cross rather than overlapping. This intermediate problem is related to maximum independent sets in line segment intersection graphs [IA83] but due to the restriction on how segments intersect we need a new NP-completeness proof for it as well.

Lemma 4 It is NP-complete, given a number k and a set of n line segments in the plane, intersecting only in proper crossings, to determine whether there exists a non-crossing subset of k line segments.

Proof The problem is clearly in NP. To prove NP-hardness, we reduce it from maximum independent sets in cubic planar graphs. Given a cubic planar graph G, with n vertices, find a straight-line embedding of G on a grid with small integer coordinates [FPP90, Sch90], and form a line segment arrangement A by placing two short parallel line segments side by side at the location of each vertex, and replacing each edge by a pair of crossing line segments, one passing through the parallel segments at each endpoint of the edge, as shown in Figure 9. Then G has an independent set of size k if and only if A has a crossing-free subset of size $2n+k$. This reduction completes the NP-completeness proof. $\square$

Figure 10 depicts our transformation from an arrangement A to a quadrilateral mesh. Choose a sufficiently small $\varepsilon > 0$, and replace each line segment s of A with seven line segments: three parallel to s at distance ε from each other and two more near each end of s, perpendicular to s and again at distance ε from each other. The perpendicular segment closer to the center of s connects all three of the line segments parallel to s, while the perpendicular line segment far-

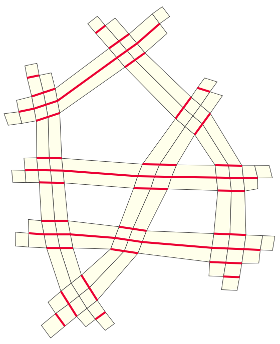

Figure 11: *Schematic view of the highway mesh from Figure 10, and an optimal partition of the mesh into structured submeshes.*

ther from the center of s connects only two of them. As the figure shows, these new line segments form a submesh of quadrilaterals two quadrilaterals wide and $3x+4$ quadrilaterals long where x is the number of other segments that cross s. There are two extraordinary vertices, one near each endpoint of s. One can imagine these submeshes as depicting highways, with two lanes of traffic (one for each quadrilateral of width); the four quadrilaterals where two highways meet form junctions of the highway system. With this analogy in mind, we call a mesh formed from this transformation a *highway mesh*, and we call the set of four quadrilaterals surrounding the intersection of two segments of the arrangement a *junction*. Figure 11 depicts the same abstract quadrilateral mesh as the highway mesh of Figure 10, with the positions of the vertices moved to make the quadrilateral mesh structure more apparent. In red is shown an optimal partition of this mesh into structured submeshes: optimal in number of submeshes, number of schematic partition vertices, and number of schematic partition edges.

In each junction of Figure 11, two paths connect the four extraordinary junction vertices in pairs; this pattern is not a coincidence. A case analysis (shown in Figure 12) proves that any highway mesh M has an optimal partition (in terms of its number of vertices, edges, or submeshes) such that within every junction J of M two pairs of extraordinary vertices of J are connected by paths, regardless of whether the schematic partition of M includes a non-boundary path entering J on zero, one, two, three, or all four of its sides.

We may construct an optimal partition for any highway mesh by first deciding which two pairs of extraordinary vertices at each junction to connect via paths and then optimally subdividing the remaining submeshes (each of which, having at most two extraordinary vertices, is easy to subdivide). The choice of which paths to use at each junction corresponds to a *casing* of the original line segment arrangement: a choice, for each crossing point, of which line segment is thought of as passing over the crossing point and

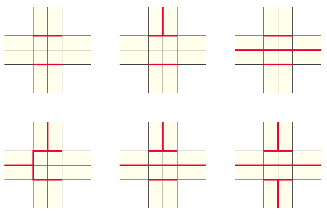

Figure 12: *Optimal partitions for each possible set of non-boundary paths entering a junction.*

which is thought of as passing under. The line segment that passes over the crossing is the one parallel to the two paths chosen at the corresponding junction. Define a segment of the arrangement to be *visible* for some casing if it passes above each of its crossings. The visible line segments for any casing cannot cross each other, and any non-crossing set of segments may be made visible in an appropriate casing. Thus, finding a casing that maximizes the number of visible segments is equivalent to the NP-complete problem of finding the largest non-crossing subset of the arrangement; for an investigation of other optimal casing problems see [EvKMS07]. If a highway mesh has n line segments with j junctions, and is cased with k visible line segments, then the optimal schematic partition consistent with this casing connects the two extraordinary vertices at the endpoints of each visible line segment and has $4j+10n-2k$ vertices, $6j+12n-3k$ edges, and $j+3n-k$ submeshes.

Theorem 2 It is NP-complete, given a quadrilateral mesh and a parameter k, to find a structured partition of the mesh with k submeshes, k vertices in the schematic partition, or k edges in the schematic partition.

Proof Each of these problems is clearly in NP. We reduce the problem of finding a crossing-free subset of line segments to each of these problems using the highway mesh construction described above. The given arrangement A has k crossing-free segments, if and only if there exists a schematic partition with at most $4j+10n-2k$ vertices, at most $6j+12n-3k$ edges, and at most $j+3n-k$ structured submeshes. Thus, since each of these problems in NP and has a reduction from a known NP-complete problem, it is itself NP-complete. $\square$

9. Empirical results

We implemented the motorcycle graph partition, and applied it to six meshes from a feature film animation application. As can be seen in Table 1, the number of edges in the schematic partition (the controlling factor for the size of its winged-edge representation) ranged from 6.33% to 11.71% of the number of edges in the original mesh, with some trend towards better compression on larger meshes. The reduction in the number of vertices was similar although not quite so great. Thus, even without the heuristic improvements discussed earlier, the space savings of the motorcycle graph is

	Original mesh		Schematic partition		Relative size	
model	vertices	edges	vertices	edges	vertices	edges
A	820	1603	98	123	11.95%	7.67%
B	1070	2110	164	247	15.33%	11.71%
C	3099	6034	286	408	9.23%	6.76%
D	6982	13933	711	1251	10.18%	8.98%
E	9958	19889	749	1299	7.52%	6.53%
F	10281	20530	761	1300	7.40%	6.33%

Table 1: *Results of applying the motorcycle graph construction to six quadrilateral meshes.*

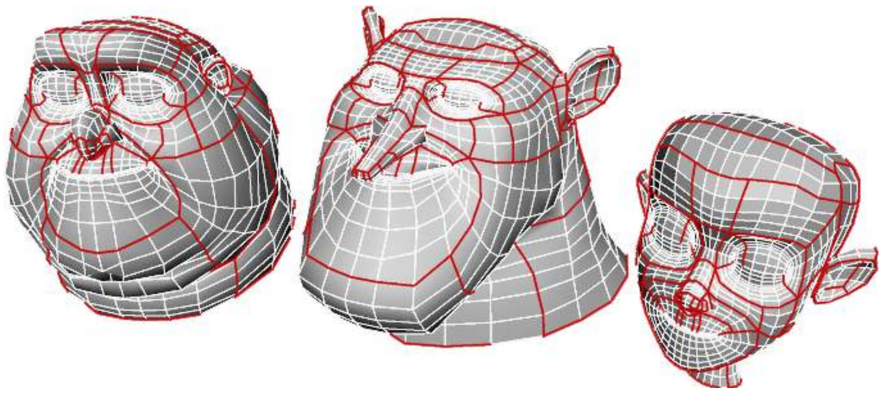

Figure 13: *Three characters from the 2007 movie* Meet the Robinsons*, with isomorphic but geometrically different meshes, and the motorcycle graphs for their meshes. The motorcycle graph compresses the meshes, with 2163 vertices and 4289 edges, into a schematic partition with 121 vertices and 222 edges.*

substantial. We would expect to see a similar reduction in the time for performing isomorphism tests on compressed meshes with respect to the times for testing their uncompressed counterparts; in this case, the speedup might be even larger due to the presence of labels in the schematic partition which can only help in isomorphism testing.

Figures 13 and 14 show examples of partitions computed by our implementation.

10. Conclusions

We have investigated and solved a number of problems involving canonical and optimal partitioning of unstructured quadrilateral meshes into structured meshes. However, many questions remain for future research:

- The motorcycle graph approximates the optimal structured partition to within a constant factor in the number of vertices, edges, and submeshes, but the constant factor that we have been able to prove is large. On the other hand, our experiments showed that it yields good compression in practice. Can we prove tighter bounds on its quality, or on the quality of improvements to it that more carefully choose which extraordinary vertices to connect and which order to connect them?
- How hard is it to approximate the optimal structured partition? Do there exist polynomial time approximations with any desired degree of approximation, or is there a limit to how closely we may approximate it?
- Our proof that optimal partitioning problems are hard relies on meshes with high genus. Are these problems hard for the low-genus meshes more commonly found in graphics applications?
- If we measure a structured partition's quality not by its compressed size but by the total number of edges of the original mesh that must be cut to produce the partition (an estimate of the visual artifacts at seams between submeshes), the motorcycle graph may be very far from optimal. Is it possible to compute or approximate the structured partition that uses as few cut edges as possible?
- Some highly symmetrical meshes may have more than one isomorphism. An example that came up in our application involved shirt buttons represented as separate objects that could be rotated in many ways while preserving combinatorial isomorphism. Generally, in such cases, there is a preferred isomorphism that we should find (for instance, one that preserves the geometric orientation of the objects) but the techniques we describe here do not address this issue, and it warrants further work.

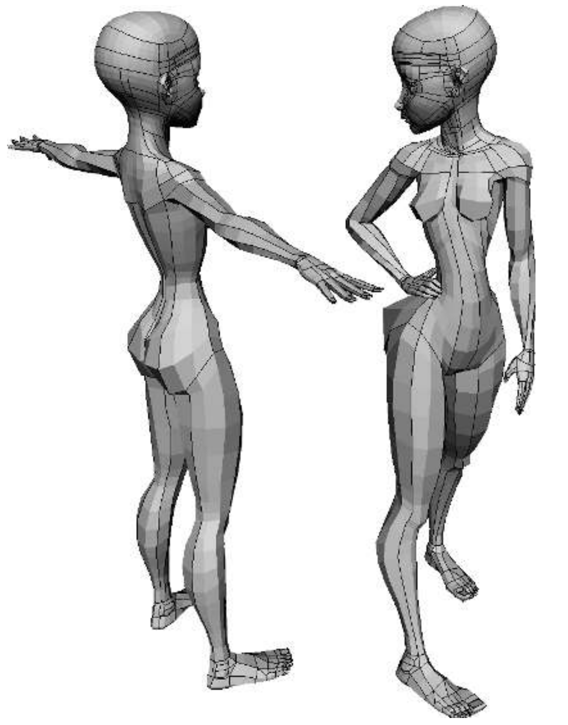

Figure 14: *The motorcycle graph computed by our implementation on an example mesh. The original mesh is shown as a set of grayscale quadrilaterals, and the motorcycle graph edges as black lines. The two figures shown have meshes that are nearly but not precisely isomorphic.*

In addition, it would be of interest to gather more data on the application of structured partitions to the problems described in the introduction. We have implemented an exact mesh isomorphism strategy based on the motorcycle graph, and the results are promising.

References

[Bau72] BAUMGART B. G.: *Winged edge polyhedron representation*. Tech. Rep. CS-TR-72-320, Stanford University, 1972.

[COLR99] COHEN-OR D., LEVIN D., REMEZ O.: Progressive compression of arbitrary triangular meshes. In *Proc. 10th IEEE Visualization Conf.* (1999), p. 11.

[CV07] CHENG S.-W., VIGNERON A.: Motorcycle graphs and straight skeletons. *Algorithmica 47*, 2 (2007), 159–182.

[Dee95] DEERING M.: Geometry compression. In *Proc. 22nd SIGGRAPH* (1995), pp. 13–20.

[EE99] EPPSTEIN D., ERICKSON J.: Raising roofs, crashing cycles, and playing pool: applications of a data structure for finding pairwise interactions. *Discrete and Computational Geometry 22*, 4 (1999), 569–592.

[EGKT08] EPPSTEIN D., GOODRICH M. T., KIM E., TAMSTORF R.: Approximate topological matching of quadrilateral meshes. In *Proc. Shape Modeling International* (2008). To appear.

[EvKMS07] EPPSTEIN D., VAN KREVELD M., MUMFORD E., SPECKMANN B.: Edges and switches, tunnels and bridges. In *Proc. 10th. Int. Worksh. Algorithms and Data Structures* (2007), Springer-Verlag, Lecture Notes in Computer Science 4619, pp. 77–88.

[FM80] FILOTTI I. S., MAYER J. N.: A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In *Proc. 12th ACM Symp. Theory of Computing* (1980), pp. 236–243.

[FPP90] FRAYSSEIX H. D., PACH J., POLLACK R.: How to draw a planar graph on a grid. *Combinatorica 10*, 1 (1990), 41–51.

[GGH02] GU X., GORTLER S. J., HOPPE H.: Geometry images. In *Proc. 29th SIGGRAPH* (2002), pp. 355–361.

[GJ77] GAREY M. R., JOHNSON D. S.: The rectilinear steiner tree problem is NP-complete. *SIAM J. Applied Mathematics 32*, 4 (1977), 826–834.

[GJ79] GAREY M. R., JOHNSON D. S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[Gro00] GROHE M.: Isomorphism testing for embeddable graphs through definability. In *Proc. 32nd ACM Symp. Theory of Computing* (2000), pp. 63–72.

[GS98] GURNHOLD S., STRASSER W.: Real time compression of triangle mesh connectivity. In *Proc. 25th SIGGRAPH* (1998), pp. 133–140.

[HW74] HOPCROFT J. E., WONG J. K.: Linear time algorithm for isomorphism of planar graphs. In *Proc. 6th ACM Symp. Theory of Computing* (1974), pp. 172–184.

[IA83] IMAI H., ASANO T.: Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *J. Algorithms 4* (1983), 310–323.

[KRS99] KING D., ROSSIGNAC J., SZYMCZAK A.: *Connectivity compression for irregular quadrilateral meshes*. Tech. Rep. GIT-GVU-99-36, Georgia Institute of Technology, 1999.

[Mil80] MILLER G.: Isomorphism testing for graphs of bounded genus. In *Proc. 12th ACM Symp. Theory of Computing* (1980), pp. 225–235.

[Oht82] OHTSUKI T.: Minimum dissection of rectilinear regions. In *Proc. Int. Symp. Circuits And Systems (ISCAS)* (1982), pp. 1210–1213.

[Sch90] SCHNYDER W.: Embedding planar graphs on the grid. In *Proc. 1st Annual ACM-SIAM Symp. Discrete Algorithms* (1990), pp. 138–148.

[SLL02] SIEK J., LEE L.-Q., LUMSDAINE A.: *The Boost Graph Library: User Guide and Reference Manual*. Addison-Wesley, 2002.

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proc. Graphics Interface* (1998), pp. 26–34.

[TR98] TAUBIN G., ROSSIGNAC J.: Geometric compression through topological surgery. *ACM Trans. Graphics 17*, 2 (1998), 84–115.