Visualization of Three-Dimensional Maps

LUIS A. P. LOZADA[1] C. X. DE MENDONÇA[2] JORGE STOLFI[1]

[1]Inst. of Computing, University of Campinas
Caixa Postal 6176, 13083-970, Campinas, SP, Brazil
{lozada,stolfi}@dcc.unicamp.br

[2]Informatics Dept., University of Maringá
CEP 87020-900 - Bloco 19, Maringá, PR, Brazil
xavier@din.uem.br

Abstract. A three-dimensional map is a partition of a 3D manifold into topological polyhedra. We consider here the problem of visualizing the topology of a three-dimensional map given only its combinatorial description. Our solution starts by automatically constructing a "nice" geometric realization of the map in $\mathbb{R}^m$, for some $m \geq 4$. The geometric realization is chosen by optimizing certain aesthetic criteria, measured by *energy functions*. We then project this model to $\mathbb{R}^3$, and display the resulting multi-celled solid object with a variety of specialized rendering techniques.

1 Introduction

Our visual experience is confined to a three-dimensional world, and therefore we find it difficult to understand the structure of objects that do not fit in three-space. Among such objects, the 3-dimensional *maps*—which include the shells of 4-dimensional polytopes—are particularly important, given their theoretical interest [15] and their applications in mechanics [21], robotics [18], and other fields.

In those applications, one often feels the need to visualize the structure of a 3D map for which one has only a combinatorial description. That means creating a suitable *geometric model* of the map. Here we describe an automatic technique for building such models.

Three-dimensional maps are precisely defined in section 2. Section 3 explains how we decompose the given map into tetrahedra, as a first stage of its modeling. Section 4 tells how the resulting tetrahedral mesh is immersed in $\mathbb{R}^m$, $m \geq 4$. The mesh is then adjusted so as to optimize certain visual effectiveness scores, which we describe in Section 5. The minimization methods we use are discussed in Section 6. Section 7 describes the computer graphics techniques that can we use to visualize the resulting object.

Prior work. There is an extensive literature on automatic drawing of graphs [1], a task which can be viewed as a "1.5-dimensional" version of our problem. Indeed, many of the tools we use—such as vertex-vertex springs to avoid fold-over—are directly borrowed from existing graph-drawing algorithms.

Rosi et al. [16, 19] studied the automatic visualization of two-dimensional maps (subdivisions of surfaces). Figure 1 shows the topology of a map on the torus surface $\mathbb{T}_2$, and a geometric model of it which was built with their tools. Our modeling and optimization techniques are natural extensions of that work.

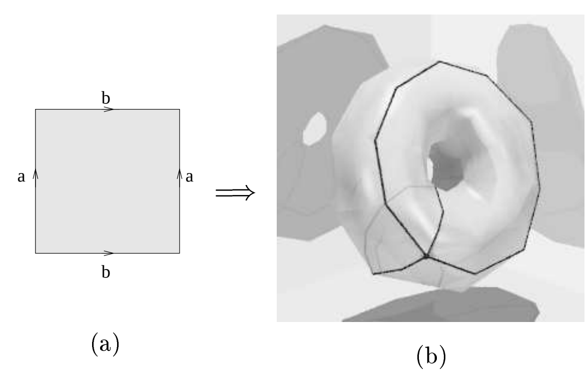


Figure 1: Automatic visualization of a two-dimensional map: (a) gluing schema, (b) optimized geometric model.

Well before computers were available, people were already trying to visualize 4D objects by producing physical models of their "perspective" projections to 3D space. See for instance Coxeter's book [4]. As early as 1967, computer graphics pioneer M. A. Noll produced movies of 4D polyhedral wireframe models by drawing each frame with a plotter and transferring the result to film [17]. Since then, many authors explored projection-based viewing of 4D polytopes, for instance

Carey et al. [3], Hanson and Pheng [9], S. Hollasch [11], Steiner and Burton [20], and several others. More recently, in the movie *Not Knot* [8], Gunn and Maxwell provided a fine example of three-dimensional map visualization, an "inside" view of a tessellation of 3D hyperbolic space by regular dodecahedra.

2 Manifolds and maps

We assume known the basic concepts of set topology, such as topological space, homeomorphism, closure, interior, etc., as defined in standard textbooks [13]. If X is a subspace of a topological space Y, we denote by $\kappa_Y X$ the closure of X in Y. (We will omit the subscript Y when it is obvious from the context.)

A *d-ball* is a topological space homeomorphic to the open unit-radius ball $\mathbb{B}_n$ of $\mathbb{R}^n$. A *d-dimensional manifold* (or *d-manifold* for short) is a compact topological space where every point has a neighborhood that is a d-ball. A *d-dimensional map* (or *d-map*) is a finite partition $\mathcal{C}$ of a d-manifold X, where, for each element c of $\mathcal{C}$, there is a continuous map ψ_c from the closed k-ball $\kappa\mathbb{B}_k$ onto κc, such that (1) $\psi_c(\mathbb{B}_k) = c$, and (2) ψ_c can be partitioned into a finite number of homeomorphisms whose ranges are single elements of $\mathcal{C}$. Any such mapping is called a *gluing of c into $\mathcal{C}$*.

By condition (1), each element c of a map is a k-ball, for some $k \in \{0..d\}$; we then say that c is a *k-element*. For $k = 0, 1, 2, 3$, the k-elements of a map $\mathcal{C}$ are its *vertices*, *edges*, *faces*, and *cells*, respectively, denoted by $V\mathcal{C}$, $E\mathcal{C}$, $F\mathcal{C}$ and $C\mathcal{C}$. The *k-skeleton* of $\mathcal{C}$ is the set of all its elements with dimension $\leq k$.

Topology of a d-map. Two maps $\mathcal{C}'$, $\mathcal{C}''$ defined on manifolds X', X'' are said to be *homeomorphic*, or *topologically equivalent*, if there is a homeomorphism ϕ of X' to X'' such that the image of each element of $\mathcal{C}'$ by ϕ is an element of $\mathcal{C}''$. The *topology* of a map $\mathcal{C}$ is the class of all maps that are homeomorphic to it.

The topology of a map can be described combinatorially in a number of ways. A popular method is to give a *polyhedral gluing schema*, consisting of a set of disjoint polyhedra with labeled faces and labeled arrows on the edges. Each polyhedron represents a cell of the map, and the labels define how the faces are to be glued in pairs to make the map. For example, figure 2 shows a polyhedral gluing schema for a 3-map whose underlying manifold is the *3-torus* $\mathbb{T}_3 = \mathbb{S}_1 \times \mathbb{S}_1 \times \mathbb{S}_1$.

The *automatic map visualization problem* is: given only the *topology* of a 3-map, represented by any suitable data structure [2, 5, 7, 14], produce a *geometric* model that will make its topology visible.

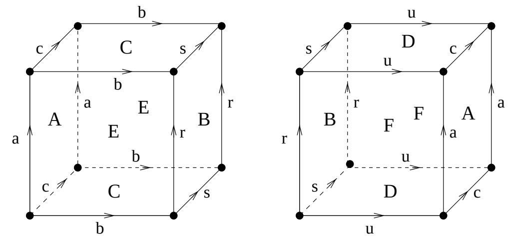
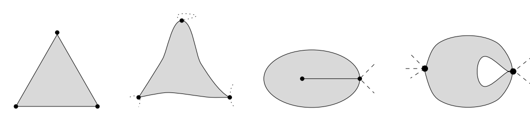


Figure 2: Polyhedron gluing schema for a 3-map on the three-dimensional torus $\mathbb{T}_3$.

3 Triangulating the map

Since the faces and cells of a 3-map may be arbitrarily complicated, instead of modeling $\mathcal{C}$ we model some map $\mathcal{C}'$ which is a suitable *refinement* (in the set partition sense) of $\mathcal{C}$, with elements of bounded topological complexity. Needless to say, the extra faces, edges and vertices introduced by this subdivision process must be omitted in the final images.

Topological triangulation. A *geometric k-simplex* is the open convex hull of $k + 1$ affinely independent points of $\mathbb{R}^m$ (for any $m \geq k$). Note that every k-simplex is an (open) k-ball. A k-element c of a d-map $\mathcal{C}$ is said to be a *topological k-simplex* if it has a gluing ψ_c into $\mathcal{C}$ whose domain is the closure κt of a geometric simplex t, and such that the image under ψ_c of each j-dimensional face of t, for every j, is an element of $\mathcal{C}$. A topological k-simplex c of $\mathcal{C}$ is said to be *proper* if ψ_c can be chosen so that it is itself a homeomorphism (i.e. so that the images of all faces of t are pairwise disjoint), as in figure 3(b); otherwise c is said to be *improper*, as illustrated in figures 3(c) and 3(d).

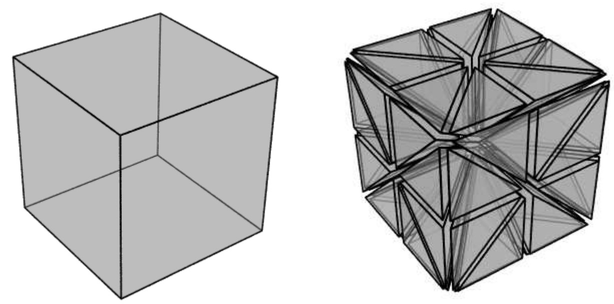


(a) (b) (c) (d)

Figure 3: A geometric 2-simplex (a), and topological 2-simplices: (b) proper, (c), (d) improper.

A *d-dimensional triangulation* is a d-dimensional map, all of whose elements are topological simplices. A triangulation is *proper* if all its simplices are proper.

Barycentric subdivision. The (*topological*) *barycentric subdivision* of a d-map $\mathcal{C}$ is a standard refinement $\mathcal{C}^\Delta$ of $\mathcal{C}$ into proper topological simplices. Informally, $\mathcal{C}^\Delta$ is obtained by recursively computing the barycentric subdivision of the $(d-1)$-skeleton of $\mathcal{C}$, and then extending that subdivision radially inwards, into each d-element of $\mathcal{C}$. See figure 4. It can be shown that the barycentric subdivision of a d-map $\mathcal{C}$ is unique up to isomorphism.



(a) (b)

Figure 4: A map with a single cubical cell (a), and its barycentric subdivision (b).

Adaptive refinement. A single step of barycentric subdivision may not produce a fine enough triangulation; further refinements may be necessary in order to produce a meaningful geometric model. The barycentric subdivision is too expensive for this purpose because it multiplies the number of tetrahedra by 24 and creates vertices of very high degree. It is usually better to apply others subdivision schemes, only to the few simplices that need it (and their neighbors, as required). Figure 5 shows some such schemas.

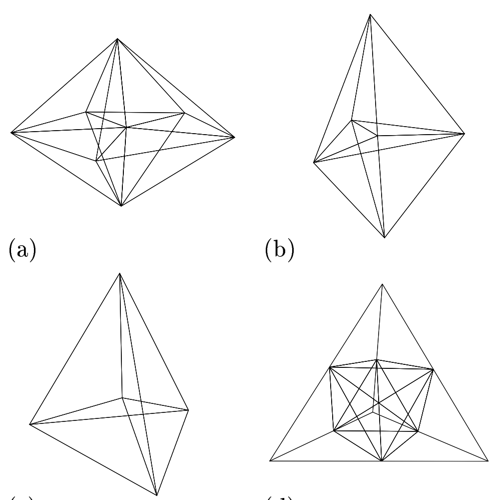


Figure 5: Some local refinement schemas for splitting (a) an edge, (b) a face, and (c,d) a tetrahedron. (Note that schema (d) requires dividing each of the adjacent tetrahedra, too).

4 Geometric representation

Once we have a proper triangulation $\mathcal{T}$ of the original d-map $\mathcal{C}$, as defined in section 3, we build a *simplicial representation* of $\mathcal{C}$ by assigning to each (topological) simplex of $\mathcal{T}$ a (geometric) simplex in some space $\mathbb{R}^m$, $m \geq d$, preserving element incidences.

Configurations. A simplicial representation of a triangulation $\mathcal{T}$ is uniquely determined by the coordinates of its vertices $V\mathcal{T}$. Therefore, we define a *configuration* of $\mathcal{T}$ as a function $\phi : V\mathcal{T} \to \mathbb{R}^m$. By extension, for each k-element c of $\mathcal{T}$ with vertices $\{v_0, \ldots, v_k\}$, we denote by $\phi(c)$ the open convex hull of the set $\{\phi(v_0), \ldots, \phi(v_k)\}$. We also denote by $\phi(\mathcal{T})$ the colletion $\{\phi(c) : c \in \mathcal{T}\}$.

If $\mathcal{T}$ is a proper d-dimensional triangulation, and the vertex positions $\phi(v)$ are in general position, then, for every $c \in \mathcal{T}$, the set $\phi(c)$ will be a geometric simplex homeomorphic to c. Moreover, if elements c_1, c_2 of $\mathcal{T}$ are incident to each other, the same will hold for the simplices $\phi(c_1), \phi(c_2)$. Therefore, except for possible intersection between elements, the collection $\phi(\mathcal{T})$ will be a simplicial representation of $\mathcal{C}$, as defined above.

Coincident elements. A simplicial representation $\phi(\mathcal{T})$ is visually effective only if distinct elements of $\mathcal{T}$ are mapped to distinct geometric simplices. However, if $\mathcal{T}$ contains two distinct j-elements c_1, c_2 that share the same $j+1$ vertices, the simplices $\phi(c_1)$ and $\phi(c_2)$ will *always* coincide, for *any* configuration ϕ. In that case we say c_1 and c_2 are *topologically coincident*.

It turns out that the barycentric subdivision $\mathcal{C}^{\Delta}$ of a map $\mathcal{C}$, while free from improper simplices, may still contain coincident pairs. We can remove such defects by subdividing some of the offending simplices, as discussed in Section 3. In particular, the triangulation $\mathcal{C}^{\Delta\Delta}$ (the barycentric subdivision of $\mathcal{C}^{\Delta}$) is provably free of coincident pairs.

5 Energy functions

To quantify the intuitive notion of "nice model," we borrowed from the graph-drawing literature the concept of *energy function*—a numerical measure $\mathcal{E}(\phi)$ of a specific kind of visual defect of a given realization ϕ.

An energy function is therefore a function from $(\mathbb{R}^m)^n$ to $\mathbb{R}$, where $n = |V\mathcal{T}|$, which ideally takes its minimum value for configurations ϕ that minimize those particular defects. We have found two energy functions particularly effective for our problem:

Curvature energy. Ideally, the three-dimensional manifold $\bigcup \phi(\mathcal{T})$ should be as smooth as possible. To understand this requirement, consider the visualization of 2-maps by means of triangle-mesh models, as in the work of Rosi et al. [16,19]. A sharp fold or crease in the triangle mesh would distract the eye from the edges of the given 2-map, and make it hard to see how its faces are joined. In order to avoid such artifacts, the dihedral angle between any two adjacent triangles should be as flat as possible.

This insight can be carried over to 3-maps as well. Each tetrahedron of $\phi(\mathcal{T})$ is contained in some 3-dimensional affine subspace of $\mathbb{R}^m$. In general, if t_1, t_2 are two adjacent tetrahedra, their images $\phi(t_1)$ and $\phi(t_2)$ will lie in two distinct 3D spaces V_1, V_2 of $\mathbb{R}^m$, whose intersection is the plane containing the shared face f. In order to flatten out the model $\phi(\mathcal{T})$ at that spot, we need to minimize the angle θ_f between V_1 and V_2. This requirement is captured by the *space curvature energy*, defined as

$$\mathcal{E}_{cv3} = \sum_{f \in F\mathcal{T}} \left(\frac{1}{1+\cos\theta_f} - \frac{1}{2} \right) \qquad (1)$$

Each summation term is approximately $\frac{1}{8}\theta_f^2$ when θ_f is small, and goes to infinity as θ_f approaches $180°$. Therefore, minimizing $\mathcal{E}_{cv3}$ tends to make the angles θ_f as small and equal as possible. The angle θ_f can be computed from the formula $\cos\theta_f = -r_1 \cdot r_2/(|r_1|\,|r_2|)$, where r_i is a vector in V_i perpendicular to the face f, and pointing into the tetrahedron t_i, for $i = 1, 2$.

Similarly, we need to avoid sharp creases in the faces of the original map $\mathcal{C}$. For that purpose we use a *surface curvature energy* function $\mathcal{E}_{cv2}$ analogous to (1), considering only pairs of adjacent triangles of $\mathcal{T}$ that belong to the same face of $\mathcal{C}$. Finally, to avoid sharp bends in the original edges, we use a *line curvature energy* function $\mathcal{E}_{cv1}$, that considers pairs of adjacent edges of $\mathcal{T}$ that belong to the same edge of $\mathcal{C}$.

Long-range spring energy. Two other important aesthetic criteria are the uniformity of size and shape of the modeled elements, and the avoidance of self-intersections. Both criteria are addressed by the *spring energy* $\mathcal{E}_{spr}$, a concept originally proposed by Kamada and Kawai for graph drawing on the plane [12]. The idea is to insert a virtual spring between every pair of vertices u, v, adjacent or not, which tries to keep them at the "correct" distance. The function $\mathcal{E}_{spr}$ can be interpreted as the total elastic energy stored in the springs:

$$\mathcal{E}_{spr} = \sum_{\substack{u,v \in V\mathcal{T} \\ u \neq v}} K_{uv} \left[\left(\frac{l_{uv}}{d_{uv}} \right)^2 + \left(\frac{d_{uv}}{l_{uv}} \right)^2 - 2 \right] \qquad (2)$$

Here $l_{uv} = |\phi(u) - \phi(v)|$ is the current Euclidean distance between the vertices u and v in $\mathbb{R}^m$; d_{uv} is the graph-theoretical distance between u and v in the 1-skeleton of $\mathcal{T}$; and K_{uv} is the stiffness of the spring (usually set to $1/d_{u,v}^2$).

Each term of formula (2) has minimum value (zero) when $l_{uv} = d_{uv}$, increases approximately like $(l_{uv}/d_{uv})^2$ for $l_{uv} \gg d_{uv}$, and approximately like $(d_{uv}/l_{uv})^2$ for $l_{uv} \ll d_{uv}$. Therefore, minimizing $\mathcal{E}_{spr}$ tends to bring all edge lengths close to the natural lengths d_{uv}. This in turn implies other desirable characteristics, such as uniform triangle areas and uniform tetrahedral volumes. On the down side, $\mathcal{E}_{spr}$ is fairly expensive to compute, since the number of terms is $\Theta(n^2)$.

Note that we apply formula (2) to the triangulation $\mathcal{T}$, rather than to the original map $\mathcal{C}$. The experience of Rosi et al. [16, 19] in the visualization of 2-maps suggests that the elements of $\mathcal{C}$ which need to be subdivided more finely are those which have the most complicated connections to the rest of the map. Therefore, if we make the elements of $\mathcal{T}$ uniform in size, we will give to each element of $\mathcal{C}$ a size appropriate to its topological complexity.

Mixed energy functions. Since each energy function usually measures only one aesthetic criterion, its minimum usually fails to meet other relevant criteria. Therefore one should normally use some combination of all those energies, such as $\mathcal{E} = \alpha_1\mathcal{E}_{spr} + \alpha_2\mathcal{E}_{cv3} + \cdots$ for some positive weights α_i.

6 Optimization

Having chosen an energy function $\mathcal{E}$, the next step is to find a configuration that minimizes it. This is the hardest part of the problem, since $\mathcal{E}$ depends (nonlinearly) on mn variables, and usually has a large number of local minima.

We use a *gradient descent* (GD) optimization technique to move from a given starting configuration to a nearby *local* minimum of $\mathcal{E}$. The method tries to follow the trajectory $\phi(t)$ (in the space of configurations) defined by the differential equation $d\phi/dt = -\nabla\mathcal{E}(\phi(t))$, starting with the initial configuration $\phi(0)$, until we reach a point where $\nabla\mathcal{E} = 0$. We solve this differential equation numerically, using a simple Euler integrator with adaptive stepsize [22].

We also tested a *single coordinate* (SC) optimization method, which consists of optimizing one variable at a time, using Brent's univariate minimization algorithm [22], while all other variables are held fixed. A "diagonal" step is performed every $n+1$ such "axial" steps, along the line connecting the outcomes of the first and the last of these steps.

Although the SC method is easier to implement than the GD method, we found that its convergence is exceedingly slow. However, as observed by Rosi et al. [16], we could speed up the SC method (by several orders of magnitude) if we evaluated only those terms of the energy function that depended on the coordinate being optimized. We haven't had yet the chance to try this optimization for 3D map modeling.

Figure 6 illustrates the progress of the GD optimizer, with the $\mathcal{E}_{spr}$ energy, for a triangulation with 80 vertices and 384 tetrahedra.

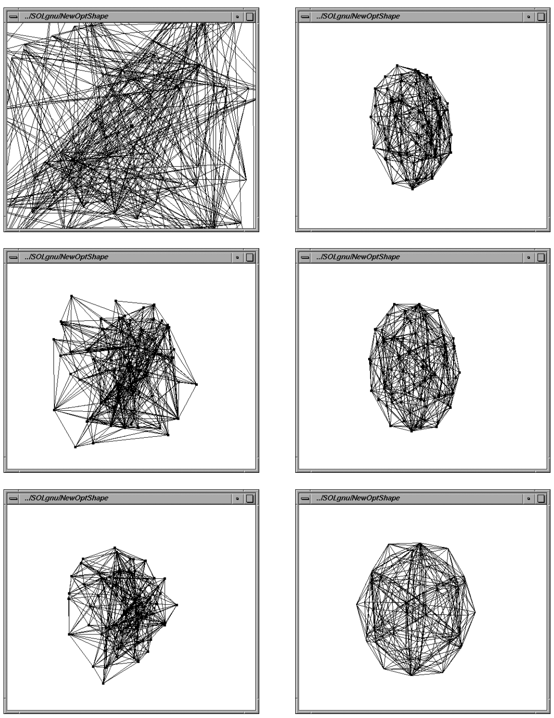


Figure 6: Optimization of an 80-vertex triangulation. In the sequence: the initial random configuration, and intermediate configurations after 5, 50, 100, 200 and 600 energy evaluations.

Searching for a global minimum. Since the energy function usually has several local minima, we repeat this search for various random initial configurations, and select the local minimum with lowest energy as the answer. Unfortunately, there is no automatic way to tell when the energy is "low enough" to provide a good visualization. So, in practice, we fix a computation budget, and return the best configuration which we can find within that limit.

We have considered using generic optimization heuristics, such as simulated annealing, to extend the search beyond the nearest local minimum. However, in our experience with 2-map visualization [16,19], such heuristics are all too slow to be useful.

Multi-scale optimization. The optimization grows very fast as the number of vertices increases, not only because of the increased cost of evaluating the energy functions, but also because a larger model has more undesirable local energy minima, where the optimization may get trapped. Therefore, for models with thousands of vertices, we generally use a multi-scale approach. Namely, we first optimize a coarse model, just fine enough to remove improper simplices and coincident pairs. Then we subdivide that model, retaining its geometry (meaning that we assign initial coordinates to the new vertices by affine interpolation); and use that configuration as the starting step for a second round of optimization. In this way we have been able to produce optimized models with more than 15000 tetrahedra, in only a few hours of computer time (on a 200 Mhz workstation).

Normalization. For 3-maps that are known to be partitions of the hypersphere $\mathbb{S}^3$, it may be desirable to enforce the constraint $|\phi(v)| = 1$ for every vertex $v \in V\mathcal{T}$. Ideally, this goal can be achieved by constrained optimization techniques. A cheaper alternative is to use unconstrained optimization, and then normalize the final solution by doing

$$\phi(v) \leftarrow \phi(v) - b; \quad \phi(v) \leftarrow \phi(v)/|\phi(v)|$$

where b is the barycenter of all the vertices. This method relies on the assumption that the optimal configuration will be the shell of a convex 4D polytope—which is often the case when the energy function is dominated by the curvature term $\mathcal{E}_{cv3}$.

7 Visualization of 3D maps

Once we have an optimized realization ϕ of the triangulation $\mathcal{T}$ in $\mathbb{R}^m$, the next problem is to help the user "see" that m-dimensional object.

Our approach is to first project the object $\phi(\mathcal{T})$ from the modeling space $\mathbb{R}^m$ to ordinary space $\mathbb{R}^3$ by a straightforward extension of the standard perspective transformation, as described by Hollasch [11]. We find that any such projection is equivalent to some parallel projection from $\mathbb{R}^m$ to $\mathbb{R}^4$, followed by perspective projection of $\mathbb{R}^4$ to $\mathbb{R}^3$. Therefore, we will assume $m = 4$ from now on.

The result of the projection from $\mathbb{R}^4$ to $\mathbb{R}^3$ is a collection of tetrahedra, glued by their faces, usually with many overlapping pairs. We then use standard rendering techniques, such as ray tracing, to display this object on a computer screen, stereo viewer, or other graphics device.

Element looks. Since our goal is to show the structure of the given map $\mathcal{C}$, rather than that of $\mathcal{T}$, we render each k-simplex of $\phi(\mathcal{T})$ in a different style, depending on its dimension and the dimension of the enclosing element of $\mathcal{C}$. In particular, we render those vertices of $\mathcal{T}$ which are also vertices of $\mathcal{C}$ as small opaque spheres. Edges and vertices of $\mathcal{T}$ that come from edges of $\mathcal{C}$ are rendered as ball-and-stick "wires" connecting those spheres. Triangles of $\mathcal{T}$ that come from faces of $\mathcal{C}$ are rendered as perforated and/or semi-transparent surfaces. Except for silhouette faces (see below), all other elements of $\mathcal{T}$ are omitted from the image.

Hidden cell elimination. The projection of a 3-map onto $\mathbb{R}^3$ generally consists of two or more interpenetrating solids, connected to each other only through their common surfaces, and each of them independently subdivided into tetrahedra. This situation is analogous to the projection of a closed mesh of triangles from $\mathbb{R}^3$ to $\mathbb{R}^2$: the result of the latter will generally be two or more superposed planar polygons, connected only through their common boundaries, each independently divided into triangles. See figure 7(a).

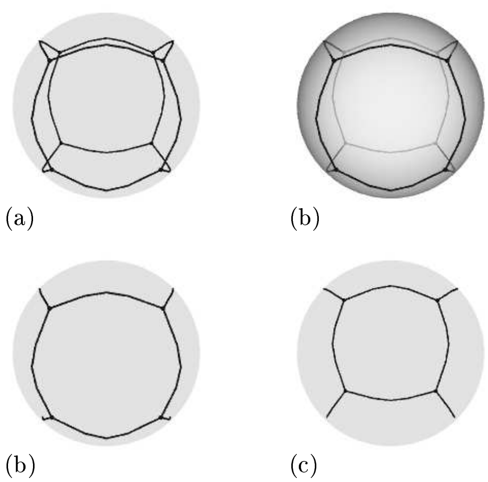


Figure 7: A 2-map modeled as a partition of $\mathbb{S}^2$. Counterclockwise: (a) full projection, (b) the front (visible) part, (c) the back (invisible) part, and (d) full projection with visibility cues.

As figure 7 shows, even the simplest maps are hard to understand when all the layers of the projection are rendered together. Thus, when projecting the model $\phi(\mathcal{T})$ from $\mathbb{R}^4$ to $\mathbb{R}^3$, we need to *remove the hidden cells*, just as we remove the hidden faces when projecting triangle meshes from $\mathbb{R}^3$ to $\mathbb{R}^2$. The idea is to consider an element of $\phi(\mathcal{T})$ *hidden* whenever the line of projection *in* $\mathbb{R}^4$ hits some other element before reaching the "camera" of the $\mathbb{R}^4 \to \mathbb{R}^3$ projection.

Visualizing the hypercube. Figure 8 illustrates this idea. The given 3-map is the boundary of the *4-dimensional cube* or *hypercube*, whose gluing schema is shown in figure 8(a). The model shown here, with 11072 vertices and 55296 tetrahedra, resulted from barycentric subdivision of the original map, followed by two stages of 12-fold refinement. The geometry of this triangulation was found automatically, by minimizing the energy $\mathcal{E} = \mathcal{E}_{spr} + 100\mathcal{E}_{cv3}$ (600 function evaluations). Figures 8(b)–(c) show the visible and hidden halves of the projection. The complete 3-map, figure 8(d), consists of these two subdivided balls with their surfaces identified.

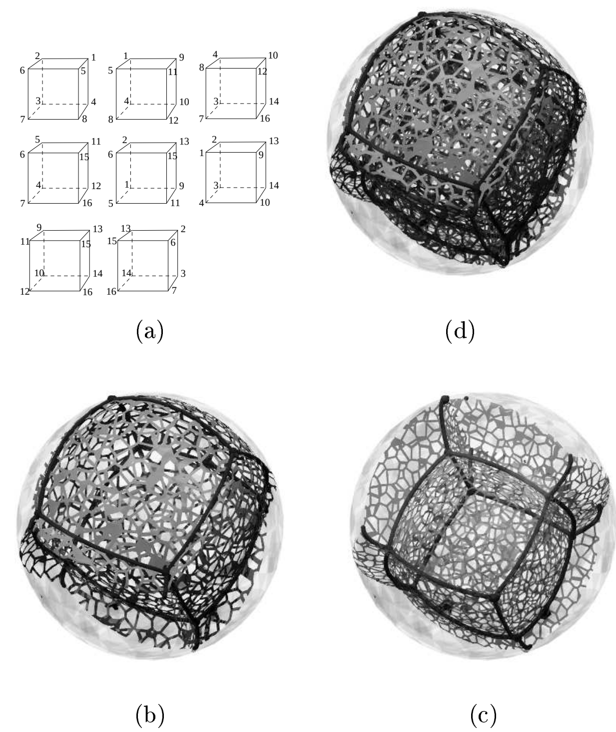


Figure 8: The hypercube map. Counterclockwise: (a) the gluing schema; (b) the "front" ("visible") part of the 3D projection; (c) the "back" ("hidden") part, and (d) the full projection.

Silhouette faces. Figures 8(b) and 8(c) should be compared with figures 7(b) and (c). Note that some of the cells, faces and edges may lie partly in the "front" half, partly in the "back" half of the projected model. In particular, to display the true extent of the cells of $\mathcal{C}$, we must render the *silhouette triangles* of $\mathcal{T}$—which lie at the interface between "front" and "back" tetrahedra—even if they don't belong to faces of $\mathcal{C}$.

Visibility hints. Instead of removing the "hidden" elements, we may distinguish them by *4D visibility hints*, e. g. by reducing their brightness and/or saturation and/or opacity, as was done in figure 8(b) and following. A cheaper alternative, which does not require a 4D visibility test, is to use *4-dimensional depth cueing*, where the looks of each element are made to depend on its distance from the 4D observer *in* $\mathbb{R}^4$.

The 16-cell. Figure 9(a) shows another example, the *16-cell*, which is the 3-skeleton of the *cross-polytope* or *hyperoctahedron* (dual of the hypercube). The original map, with 8 vertices and 16 tetrahedra, was first optimized with the $\mathcal{E}_{spr}$ (500 evaluations). Three stages of the 12-fold refinement produced a triangulation with 5568 vertices and 27648 tetrahedra, which was optimized for $\mathcal{E} = 0.98\mathcal{E}_{cv3} + 0.01\mathcal{E}_{cv2} + 0.01\mathcal{E}_{cv1}$ (1000 evaluations). Figures 9(b)–(d) show the resulting model.

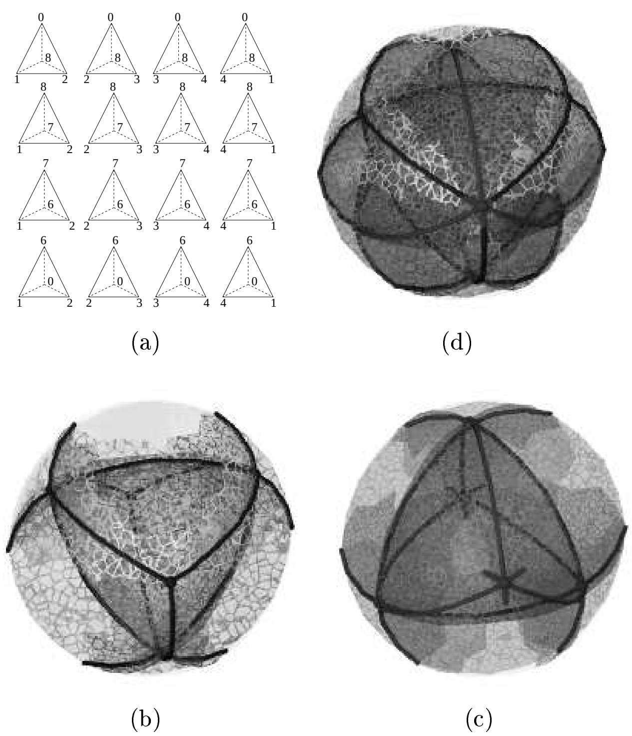


Figure 9: The 16-cell. Counterclockwise: (a) gluing schema; (b) "front" ("visible") part; (c) "back" ("hidden") part; (d) the full projection.

Visibility testing. In general, the visibility of an element can be tested by ray casting in $\mathbb{R}^4$ [11] or with a three-dimensional Z-buffer [20]. As a special case, if the model happens to be the 3-skeleton of a 4D convex polytope, it suffices to test the orientation of each projected tetrahedron in $\mathbb{R}^3$. This method is analogous to the *backface culling* for convex polyhedra [6].

Camera motion in $\mathbb{R}^4$. No static rendering technique, however, is as effective as the use of motion (of the observer or the model) in showing the 4D structure of the maps. As a minimum, we can use motion in $\mathbb{R}^3$ to help the user gain a better understanding of the 3D structure of the projected model. Far more effective, however, is to rotate the model in 4-space, so that the projected 3D object itself changes. See figure 10.

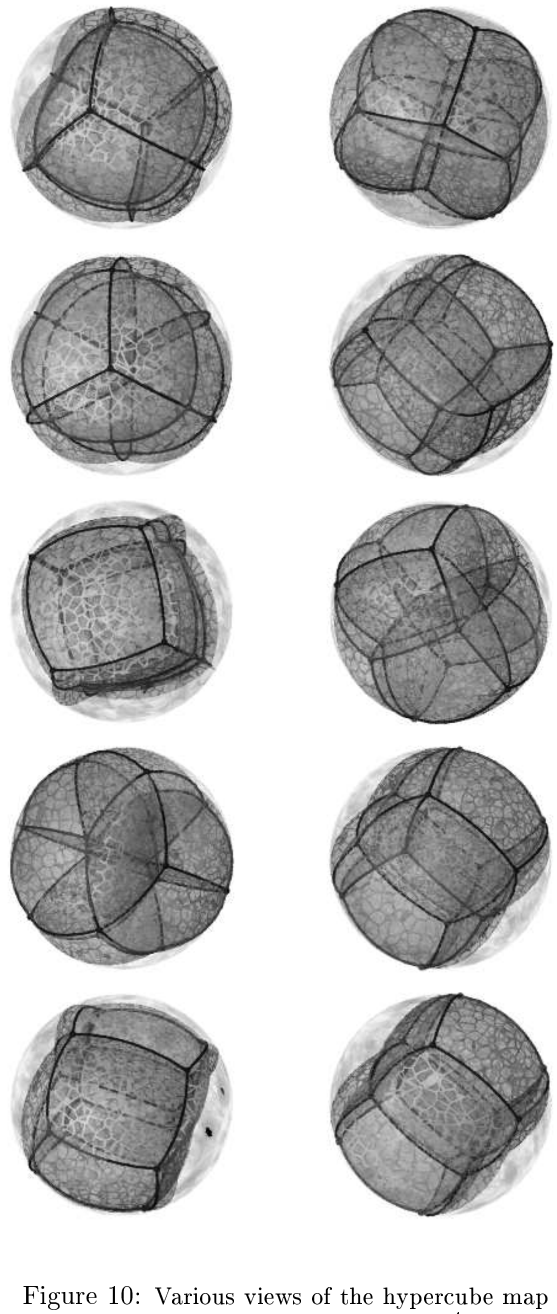


Figure 10: Various views of the hypercube map obtained by rotating the 4D model in $\mathbb{R}^4$.

Other approaches. A completely different approach to this sub-problem is to provide an *inside view* [8] of the model, where the camera is placed inside the 3-manifold, and a 2D image of the map is formed, on some projection plane, by casting rays *within the manifold itself*, along geodesic paths. However, this method requires a smooth model of the map. Moreover, map elements may be missing or duplicated in the image, due to convergence of the geodesics. Thus, in spite of its limitations, projection still seems to be more effective than inside view for showing the global structure of the map.

Implementation details Our modeling and optimization tools were implemented in *Modula-3* [10], and are presently specialized for immersions in $\mathbb{R}^4$. We used the *facet-edge* data structure of Dobkin and Laszlo [5] to represent the topology of the given maps and their triangulations. We used a modified version of Hollasch's *Wire4* tool for real time model inspection, and *POV-Ray* for high-quality rendering.

8 Conclusions and Future Work

Our tests (of which only a tiny sample can be shown here) indicate that the automatic visualization of three-dimensional maps is perfectly feasible, according to the methodology described here. These results are encouraging, but there is still a lot of work to be done.

Acknowledgments

This work has been supported by grant 96/09873-0 from the Foundation for Research Support of the State of São Paulo (FAPESP), and grant 301016/92-5(NV) from the Brazilian Council for Scientific and Technological Development (CNPq). We are grateful to Richard Hollasch and the authors of all free software we used, especially *Emacs*, LaTeX, *Modula-3*, *Ghostview*, *PBMplus*, *POV-Ray*, and *Wire4*.

References

[1] G. Di Battista, P. D. Eades, and R. Tamassia. Algorithms for drawing graphs: An annotated bibilography. Technical report, Departament of Computer Science, Univ. of Newcastle, 1993.

[2] E. Brisson. Representing geometric structures in d dimensions: Topology and order. *Proc. 5th Annual ACM Symp. on Computational Geometry*, pages 218–227, June 1989.

[3] S. A. Carey, R. P. Burton, and D. M. Campbell. Shades of a higher dimension. *Computer Graphics World*, pages 93–94, Oct. 1987.

[4] H. S. M. Coxeter. *Regular Polytopes*. Dover, 1973.

[5] D. P. Dobkin and M. J. Laszlo. Primitives for the manipulation of three-dimensional subdivisions. *Algorithmica*, 4:3–32, 1989.

[6] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1990.

[7] A. L. P. Guedes. Representação de variedades combinatórias de dimensão n. Master's thesis, Universidade Federal do Rio de Janeiro, 1995.

[8] C. Gunn and D. Maxwell. Not knot. Video movie, distributed by Jones and Bartlett, 1991.

[9] A. J. Hanson and A. Pheng. Illuminating the fourth dimension. *IEEE Computer Graphics and Applications*, 12(4):54–62, 1992.

[10] S. P. Harbison. *Modula-3*. Prentice-Hall, 1992.

[11] S. R. Hollasch. *Four-Space Visualization of 4D Objects*. PhD thesis, Arizona State Univ., August 1991.

[12] T. Kamada and S. Kawai. An algorithm for drawing general undirected graphs. *Inf. Proc. Letters*, 31(1):7–15, 1989.

[13] C. Kosniowski. *A First Course in Algebraic Topology*. Cambridge Univ. Press, 1980.

[14] P. Lienhardt. Subdivisions of n-dimensional spaces and n-dimensional generalized maps. In *Proc. of the 5th ACM Symp. on Comput. Geometry*, pages 228–236, 1989.

[15] S. Lins and A. Mandel. Graph-encoded 3-manifolds. *Discrete Mathematics*, 57:261–284, 1985.

[16] L. P. Lozada, C. F. X. de Mendonça, R. M. Rosi, and J. Stolfi. Automatic visualization of two-dimensional celullar complexes. In *Proceedings of Graph Drawing'96 - Lecture Notes in Computer Science*, pages 303–317, 1996.

[17] M. A. Noll. A computer technique for displaying n-dimensional hyperobjects. *Communications of the ACM*, 10(8):469–473, 1967.

[18] A. Paoluzzi. Motion planning + solid modeling = motion modeling. Technical Report 17-89, Dip. di Informatica e Sistemistica, Univ. di Roma "La Sapienza," Rome, Italy, 1989.

[19] R. M. Rosi and J. Stolfi. Automatic visualization of two-dimensional cellular complexes. Technical Report IC-96-02, Institute of Computing, Univ. of Campinas, Brazil, 1996.

[20] K. V. Steiner and R. P. Burton. Hidden volumes: The 4th dimension. *Computer Graphics World*, pages 71–74, Feb. 1987.

[21] W. P. Thurston and J. R. Weeks. The mathematics of three-dimensional manifolds. *Scientific American*, 251(1):94–106, 1984.

[22] S. A. Teukolsky W. H. Press, B. P. Flannery and W. T. Vetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge Univ. Press, 1986.