Polyhedron Realization for Shape Transformation

Avner Shapiro
Department of Computer Science
The Hebrew University
Jerusalem, Israel, 91904

Ayellet Tal
Department of Electrical Engineering
Technion – Israel Institute of Technology
Haifa, Israel, 32000

Abstract

Polyhedron realization is the transformation of a polyhedron into a convex polyhedron with an isomorphic vertex neighborhood graph. We present in this paper a novel algorithm for polyhedron realization, which is general, practical, efficient, and works for any zero–genus polyhedron. We show how the algorithm can be used for finding a correspondence for shape transformation. After the two given polyhedra are being realized, it is easy to merge their *vertex–neighborhood graphs* into a common graph. This graph is then induced back onto the original polyhedra. The common vertex–neighborhood graph allows the interpolation of the corresponding vertices.

Key words: Shape transformation, metamorphosis, correspondence, polyhedron realization.

1 Introduction

Metamorphosis is the gradual evolution of a source object, through intermediate objects, into a target object. It has numerous applications, including scientific visualization and animation sequences in the film and advertising industries. Much of the work done in the area focused on two–dimensional metamorphosis (e.g., [2], [4], [6], [10], [12], [24], [25], [28]). Three–dimensional morphing sequences are harder to compute, compared to their two–dimensional counterparts. They have, however, many advantages over the two–dimensional sequences. For instance, changes in the viewing position do not require a repeated computation of the entire sequence.

Several approaches to three–dimensional metamorphosis have been explored. Hong et al. [14] base their method on matching the faces whose centroids are closest. Bethel and Uselton [3] add degenerate vertices and faces until a common vertex–neighborhood graph is achieved. Chen and Parent [5] extend a transformation for piecewise linear two–dimensional contours to three dimensions. In their method, morphing is done between polyhedra which are represented by sets of three–dimensional planar contours. Kaul and Rossignac [17] use the Minkowski sum of the models to obtain a smooth transition. Parent [21] establishes correspondence by recursively subdividing the surfaces until an identical topology is achieved. Kent et al. [18] [19] create a correspondence between zero–genus polyhedra, by mapping them onto a sphere and merging their vertex–neighborhood graphs. Kanai et al. [16] utilize a similar algorithm by using the harmonic maps of the polyhedra. Several voxel–based methods were also explored ([7], [13], [15], [20], [23]).

Morphing a three–dimensional polyhedron into another is usually done in two steps: finding a correspondence between the polyhedra and interpolating the corresponding vertices. In this paper we focus on the problem of finding a correspondence. We follow the approach introduced in [19], where two zero–genus polyhedra are first "inflated" like balloons until they become spherical. The one-to-one correspondence is then found between points on the surfaces of the two spheres and induced back onto the original polyhedra. The algorithm in [19] works very well for several classes of polyhedra. It remained open in that paper whether there exists an algorithm that is applicable to any zero–genus polyhedra, or even to wider classes of polyhedra. The major contribution of the current paper is devising a novel polyhedron realization algorithm that works for any zero–genus polyhedron.

"Inflating" a polyhedron into a convex polyhedron rather than into a surface of a sphere has a few advantages. First, it is general and works for any zero–genus polyhedron rather than for specific classes. Second, "inflating" a polyhedron into a sphere might result with a non-convex polyhedron. Convexity, however, is a desired property. Third, since spherical arcs need not be intersected, integer arithmetic can be used, and the algorithm can become become more robust. Finally, if a sphere is required, it can be easily obtained from a convex polyhedra.

We say that a three–dimensional polyhedron, P, is *realized* by another three–dimensional polyhedron, P', if P' is convex, and the two polyhedra have isomorphic vertex–neighborhood graphs. In other words, there exists a one-to-one correspondence between the vertex sets and the edge sets of the two polyhedra.

Steinitz [11] proved that a graph is realizable if and only if it is planar and three-connected. In other words, our algorithm can be considered as a different proof of Steinitz's theorem for the special case where the faces are triangular. This is the most common case for real data. Our proof is constructive, and gives rise to an algorithm which is efficient, avoids circular arcs, and is easy to implement. We implemented the algorithm and achieved very good results.

The realization algorithm we propose consists of two phases: simplification and re-attachment. During the simplification phase, vertices are detached from the vertex–neighborhood graph of the polyhedron one by one, and the graph is re-triangulated. This step is repeated until a 4-clique results. The second phase starts by creating a tetrahedron, the convex polyhedron which realizes a 4-clique. The vertices are re-attached to the polyhedron, in the reverse order to their detachment, while maintaining the polyhedron's convexity. The vertex–neighborhood graph of each convex polyhedron, created in the process, is isomorphic to a graph encountered during the simplification stage. Thus, the last polyhedron's vertex–neighborhood graph is isomorphic to the input vertex–neighborhood graph.

In three–dimensional metamorphosis, after the input polyhedra are being realized, the resulting convex polyhedra are merged on the boundary of one of them. The merged edge set is later used for refining the original polyhedra. The result of the mutual refinement is the desired correspondence, which is utilized during the interpolation phase of the shape transformation process.

The remainder of the paper is organized as follows. The next section presents a high–level description of the shape transformation algorithm that uses polyhedron realization. Section 3 describes the realization algorithm, and proves its correctness. Section 4 describes how the two realized polyhedra can be merged into one structure. Section 5 demonstrates our results. We conclude in section 6.

2 The Shape Transformation Algorithm

In this section we describe the general scheme of using a realization algorithm for establishing a correspondence between the vertices of two given polyhedra. Our scheme follows the scheme described in [19]. However, while in [19] the polyhedra are "inflated" into spheres, we realize them into convex polyhedra. We begin with a few definitions.

Definition 2.1 Polyhedron vertex–neighborhood graph, *or the* **1–skeleton of the polyhedron** *is a graph $G = (V, E)$ where V is the set of vertices of the polyhedron, and there exists an edge $e \in E$ that connects two vertices v_i and $v_j \in V$ iff there exists an edge between the two vertices in the polyhedron.*

Definition 2.2 Graph refinement: *$\tilde{G} = (\tilde{V}, \tilde{E})$ is a refinement of the graph $G = (V, E)$ if $V \subseteq \tilde{V}$ and an edge $(v_i, v_j) \in E$ is replaced by a path $(v_i, u_1), (u_1, u_2), \ldots, (u_{n-1}, v_j)$ in $\tilde{E}$ such that $\forall u_i, u_i \in \tilde{V} - V$, and the vertex paths replacing the edges are mutually disjoint.*

Definition 2.3 Polyhedron refinement: *A polyhedron $\tilde{P}$ is a refinement of a polyhedron P, if $\tilde{P}$ and P are congruent, and the vertex–neighborhood graph of $\tilde{P}$ is a graph refinement of the vertex–neighborhood graph of P.*

Definition 2.4 Polyhedron realization: *A polyhedron $P_{|P|}$ is said to realize a polyhedron P, if $P_{|P|}$ is convex and the vertex–neighborhood graphs of $P_{|P|}$ and P are isomorphic.*

We are given a pair of zero–genus polyhedra, P and Q. The algorithm for finding a correspondence between P and Q consists of five phases, and proceeds as follows.

1. Create a convex polyhedron, $P_{|P|}$, whose vertex–neighborhood graph is isomorphic to P's, using the realization algorithm described in the next section.

2. Create a convex polyhedron, $Q_{|Q|}$, which realizes Q, in a similar way.

3. Refine the vertex–neighborhood graphs of $P_{|P|}$ and $Q_{|Q|}$ in order to produce a joint vertex–neighborhood graph. A few merge algorithms were proposed in the past. In [1], two triangulations of corresponding planar polygons are merged. This is also done in [16] when two harmonic maps are being merged. In [19], the faces of polyhedra whose vertices reside on the sphere are merged. Our algorithm, which is described in Section 4, merges the boundaries of convex polyhedra.

   The main idea of the algorithm in to project the vertices and the edges of $Q_{|Q|}$ onto the boundary of $P_{|P|}$. The projected edges of $Q_{|Q|}$ are then intersected with the edges of $P_{|P|}$. These are the edges of the refined polyhedron of $P_{|P|}$. After the faces of the refined polyhedron are found, its vertices, edges, and faces are projected onto $Q_{|Q|}$, to form its refined polyhedron.

4. Induce the merged vertex–neighborhood graph found in (3) onto P, to form a polyhedron, $\tilde{P}$. The polyhedron $\tilde{P}$ is a refinement of P, since its vertex–neighborhood graph is isomorphic to the merged one, and thus a refinement of P's vertex–neighborhood graph, and it is congruent to P.

5. Repeat the previous phase for Q, creating a new polyhedron, $\tilde{Q}$, which is a refinement of Q.

At the end of this process $\tilde{P}$ and $\tilde{Q}$ are refinements of P and Q respectively. Moreover, their vertex–neighborhood graphs are isomorphic. Therefore, $\tilde{P}$ and $\tilde{Q}$ can be used for establishing a correspondence between P and Q.

The next sections elaborate on each of the above phases of the algorithm. Section 3 focuses on the realization algorithm (steps (1)–(2)). Section 4 describes the merging process of two given vertex–neighborhood graphs, and explains how the merged graph can be induced back onto the original polyhedra, to produce their refinements (steps (3)–(5)).

3 The Polyhedron Realization Algorithm

Given a zero–genus polyhedron, P, we present an algorithm for constructing a zero–genus convex polyhedron, $P_{|P|}$, that realizes P. We assume that the input polyhedron has triangular faces. If not, we triangulate the faces in linear time. The algorithm consists of two phases: simplification and re-attachment.

1. **Simplification** – Simplify the vertex–neighborhood graph by removing a low–degree vertex from it and re-triangulating the resulting graph. This operation is repeated until a 4–clique graph, representing a tetrahedron, results. During this phase, a hierarchy of planar graphs, $G_{|P|} = G(P), G_{|P-1|}, ..., G_4 = K_4$, is constructed, such that G_i has i vertices for $4 \leq i \leq |P|$.

2. **Re-attachment** – Given the above graph hierarchy, proceed bottom up, and construct a convex polyhedron, P_i, for each graph G_i, so that $G(P_i)$ and G_i are isomorphic, for $4 \leq i \leq |P|$. This procedure first constructs a tetrahedron, P_4, so that $G(P_4) \cong G_4 = K_4$. Suppose that a convex polyhedron, P_i, is constructed, such that $G(P_i) \cong G_i$. P_i is then used, jointly with G_{i+1}, to construct a convex polyhedron, P_{i+1}, such that $G(P_{i+1}) \cong G_{i+1}$, $4 \leq i < |P|$.

Once the convex polyhedron $P_{|P|}$ is constructed, our goal is achieved: $G(P_{|P|}) \cong G_{|P|} = G(P)$.

Note that the simplification phase somewhat resembles the way the Dobkin–kirkpatrick hierarchy [8] [9] is constructed. However, while in our algorithm a single vertex is removed at each step, in the Dobkin–kirkpatrick hierarchy an independent set of vertices is removed. A major difference between the algorithms is in the re-attachment phase. In the Dobkin–kirkpatrick hierarchy the polyhedron is known to be convex, and thus the vertices need not be re-positioned. In our case, re-positioning is a major consideration. We elaborate on each of the two phases below.

3.1 Phase 1: Simplifi cation

Given a triangular graph, G_i, representing a polyhedron, we describe in this section an algorithm for constructing a triangular graph, G_{i-1}, with one vertex less. To do it, we repeat the following two steps.

1. Find a vertex, v, of degree 3, 4, or 5, and remove it from the graph.

2. Re-triangulate the resulting face.

This procedure is graphically described in Figure 1. It is repeated until a 4–clique results. The proof of correctness of the simplification phase distinguishes between three cases according to the degree of the vertex removed. If a vertex of degree three is removed, it is sufficient to “close” the hole created with a face. Since the graph contains at lest four vertices, it cannot have multiple edges. In case the vertex is of degree four we show in Lemma 3.2 that one of the two diagonals of the hole created can be added. Finally, in case the vertex is of degree five we show in Lemma 3.3 that the $5-$gon hole created can be triangulated by adding two diagonals emanating from one vertex. We summarize in Theorem 3.4.

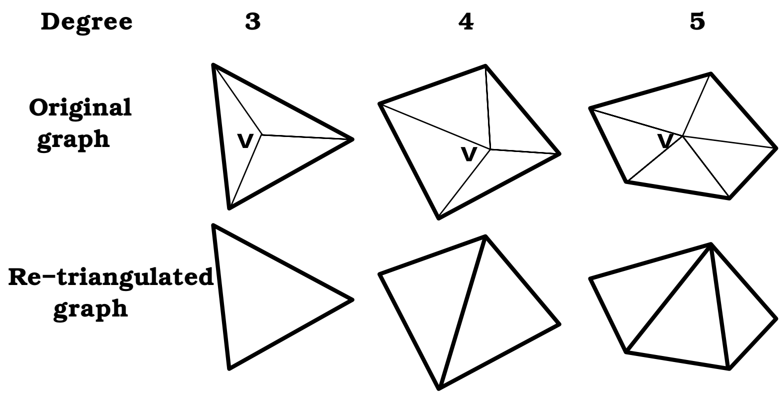


Figure 1: Simplification

Lemma 3.1 *A planar triangular graph must contain at least one vertex of degree smaller than six.*

Lemma 3.2 *If a vertex v, of degree four, is removed from a planar triangular graph, G_i, then (1) The neighbors of v in G_i cannot be all inter-connected (in the new graph), and (2) A diagonal can be added, between two of v's neighbors, so that the graph remains planar.*

Proof: Suppose, on the contrapositive, that the four vertices form a clique, contradicting (1). Together with v, the five vertices used to form a clique before the removal of v. Since a five–vertex clique cannot be planar, it contradicts our assumption that we started from a planar graph.

Let us denote the four neighbors of v as v_1, v_2, v_3, and v_4, so that v_n and $v_{(n+1)mod4}$ are neighbors. Since we started from a triangulated graph, and v was connected to all four vertices, the edges v_1-v_2, v_2-v_3, v_3-v_4, v_4-v_1

exist. Since they are not all connected to each other, one diagonal in the quadrilateral does not exist in the graph. It is obvious that the missing diagonal can be added so that planarity is not violated. □

Lemma 3.3 *If a vertex v, of degree five, is removed from a planar triangular graph, G_i, then (1) There exist at most three diagonals between the neighbors of v in G_i and (2) Two diagonals, which share a common vertex, can be added between v's neighbors so that the graph remains planar.*

Proof: We show first that there can exist at most three diagonals connecting the five vertices after we remove the vertex v. Suppose on the contrapositive that there are at least four diagonals. In this case, at most one pair of vertices, v_k and v_j, are not connected by a diagonal in the resulting pentagon. Consider four vertices, among which only one is v_k or v_j. They are all inter–connected by diagonals in the resulting pentagon. In the original graph, these four vertices, together with v, formed a 5–clique. This contradicts planarity.

Having eliminated the possibility of four or five diagonals, we assume that there are three diagonals, and show that there exists a vertex through which none of the diagonals pass. Suppose on the contrapositive that the diagonals cover all the vertices. There must exist a vertex through which two diagonals pass, otherwise three diagonals would cover six vertices. Without loss of generality, assume it is v_5, and its diagonal–neighbors are v_2 and v_3. See Figure 2. By our assumption, the third diagonal must connect the remaining two vertices v_1 and v_4.

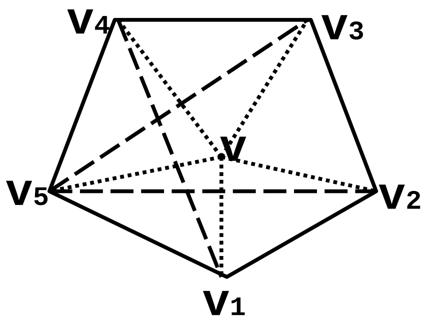


Figure 2: A pentagonal face with three diagonals covering all the vertices.

We now consider the original graph, before the detachment. We partition the vertices into two groups, $A = \{v_1, v_3, v_5\}$ and $B = \{v, v_2, v_4\}$. Each member of A is connected to each member of B, and each member of B is connected to each member of A. Thus, the original graph contains a fully-connected bipartite graph known as $K_{3,3}$, and therefore planarity is contradicted.

If there exist only one or two diagonals, they cannot cover all five vertices, and a vertex through which two new diagonals can pass, exists. It is not hard to show that the two missing diagonals can be added in a planar fashion. □

Theorem 3.4 *Given a triangular planar graph G_i, whose number of vertices is at least four, a vertex v of degree three, four, or five can be found and removed, and the resulting graph can be re-triangulated so that the graph remains planar.*

Proof: By Lemma 3.1, there exists at least one vertex of degree smaller than six. Since the graph is triangular, the degree cannot be two. Thus, there exists at least one vertex of degree three, four, or five, which can be detached. If the detached vertex is of degree three, the hole left is a triangle, so that it is sufficient to close it with a face. Obviously, the graph is still planar. Since there are at least four vertices, the face intersects other triangles along an edge, a vertex or an empty set. Otherwise (the vertex is of degree four or five), by Lemmas 3.2 and 3.3, diagonals can be added so that the graph remains planar. The resulting graph is also triangular. To see this, we distinguish between the different cases. If a vertex of degree three is detached, the former neighbors of v form a triangle. If a vertex of degree four is detached, a quadrilateral is formed, and by Lemma 3.2 a diagonal can be added to form two triangles. Finally, if a vertex of degree five is detached, a pentagon is formed, and by Lemma 3.3 two adjacent diagonals can be added to form three triangles. □

Lemma 3.5 *The simplification procedure can be repeated until there are four vertices left. In this case the graph is necessarily the* $1-$*skeleton of a tetrahedron.*

Figures 3 – 4 illustrate the simplification phase by providing a few snapshots of an epcot being simplified, starting from a 50-vertex epcot (to the left) and ending with a tetrahedron (to the right). Note that even though the algorithm works on planar graphs, and not on the three–dimensional structures, for clarity, the snapshots shown below illustrate the three–dimensional structures of the intermediate objects.

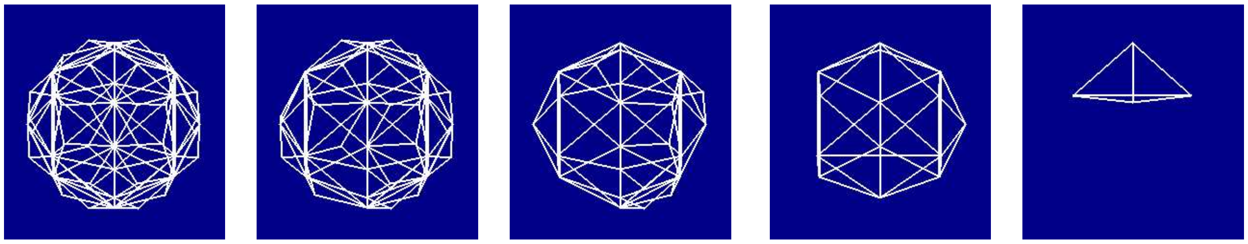


Figure 3: Epcot simplification – wire-frame

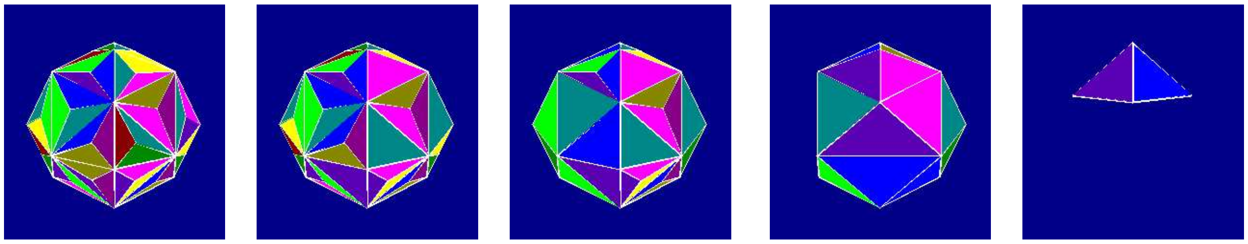


Figure 4: Epcot simplification - faceted shading

3.2 Phase 2: Re-Attachment

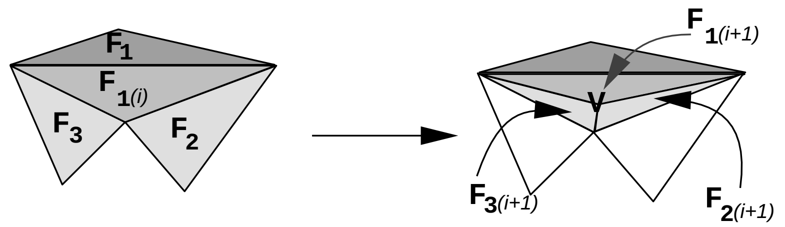
We are given a hierarchy of vertex–neighborhood graphs, $G_{|P|} = G(P), G_{|P|-1}, ..., G_4$, which is the output of the simplification phase. The goal is to construct a sequence of convex polyhedra, $P_{|P|}, P_{|P|-1}, ..., P_4$, such that $G(P_i)$ is isomorphic to G_i, $4 \leq i \leq |P|$.

The re-attachment phase proceeds bottom up. A convex polyhedron, P_4, for G_4, is first constructed. Since G_4 is a 4-clique, it can be realized by a tetrahedron. Suppose that a convex polyhedron, P_i, has been already constructed for the vertex–neighborhood graph G_i. We show next how P_i can be used, along with G_{i+1}, to construct a convex polyhedron P_{i+1}, whose vertex–neighborhood graph is isomorphic to G_{i+1}, $4 \leq i < |P|$. At the end of this process, $P_{|P|}$ is the realized P.

The major consideration is how to add the vertex v, which was detached during the simplification phase, to P_i, in order to form P_{i+1}. We distinguish between three cases according to the degree of v.

In case v is of degree three, during the simplification process its cone of faces was replaced by a triangular face, $F_{1(i)}$. Thus $F_{1(i)}$ in P_i needs to be replaced by a cone of three faces in P_{i+1}, as shown in Figure 5. Denote the three neighboring faces of $F_{1(i)}$ in P_i, F_1, F_2 and F_3.

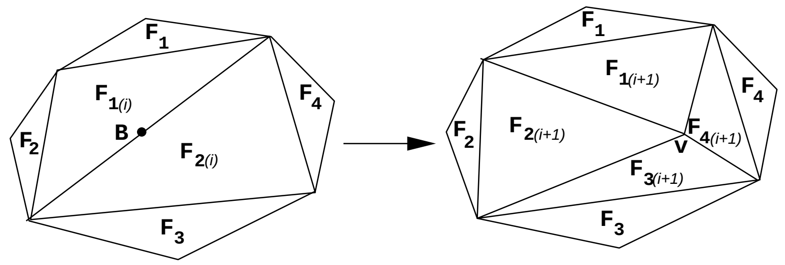
In case v is of degree four, By Lemma 3.2, G_i was created by removing v from the graph G_{i+1} and adding a diagonal, thus forming two new faces $F_{1(i)}$ and $F_{2(i)}$, as illustrated in Figure 6. Therefore, $F_{1(i)}$ and $F_{2(i)}$ need



(a) P_i (b) P_{i+1}

Figure 5: Attachment of v of degree three to P_i

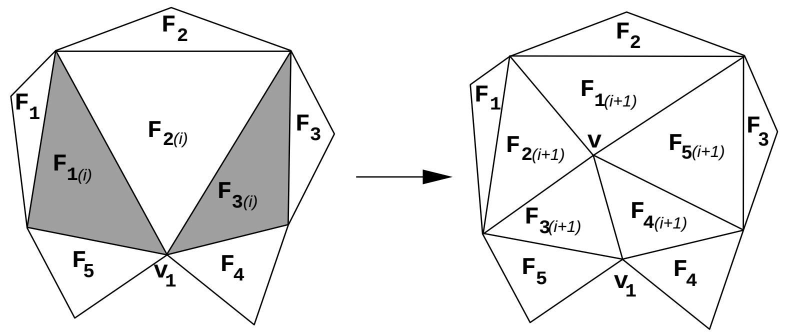
to be replaced by the cone of four faces. Denote the four neighboring faces of $F_{1(i)}$ and $F_{2(i)}$ in G_i, F_1, F_2, F_3, and F_4.



(a) P_i (b) P_{i+1}

Figure 6: Attachment of v of degree four to P_i

In case v is of degree five, By Lemma 3.3, G_i was created by removing v from the graph G_{i+1} and adding two adjacent diagonals, thus forming three new faces $F_{1(i)}$, $F_{2(i)}$ and $F_{3(i)}$, as illustrated in Figure 7. Therefore, $F_{1(i)}$, $F_{2(i)}$ and $F_{3(i)}$ need to be replaced by the cone of five faces. Denote the five neighboring faces of the new faces in G_i, F_1, F_2, F_3, F_4 and F_5.



(a) P_i (b) P_{i+1}

Figure 7: Attachment of v of degree five to P_i

By the induction hypothesis, a convex polyhedron P_i whose vertex–neighborhood graph is G_i has been created. We need to show how to add v to P_i in order to form P_{i+1}. Let V_i be the set of vertices of P_i and let $\mathcal{F}_i$ be $\{F_{j(i)} | 1 \leq j \leq d-2\}$, where d is the degree of the removed vertex (i.e., the faces created during the simplification stage). We need to add a point v such that the convex hull of $V_i \bigcup v$ contains all the faces of P_i not in $\mathcal{F}$ and all the faces formed by v and the edges of the boundary of $\mathcal{F}$ (i.e., a cone with apex v).

We say that a point v is above (resp. below) a face F if that point is in the half–space of the supporting plane of F that does not (resp. does) contain the polyhedron. In order to find a point v that satisfies the above requirements, v needs to be above all the faces in $\mathcal{F}$ and below all the faces adjacent to $\mathcal{F}$. In this case, by to beneath–beyond convex hull algorithm algorithm [22], v should be added to the convex hull, and the triangulation faces should be detached and replaced by the new faces.

It remains to find such a v. We first describe the algorithm for positioning v, and next prove with a series of lemmas the correctness of the positioning algorithm. The cases of v of degrees three or four are relatively easy, and are proved in Lemmas 3.6–3.7. The case where v is of degree five is more involved and is proved in Lemmas 3.8–3.9.

To find v, we first define a ray r along which v is located, as described below. The ray r is defined by its base B, and its direction N. To find B and r, we follow the procedure below.

1. **Determine B and N:**
   There are three possible cases:
   (a) if v is of degree three, as in Figure 5, then
       $B =$ the center of mass of the face $F_{1(i)}$.
       $N =$ the normal of $F_{1(i)}$.
   (b) if v is of degree four, as in Figure 6, then
       $B =$ the midpoint of the edge shared by $F_{1(i)}$ and $F_{2(i)}$.
       $N =$ the average of $F_{1(i)}$'s normal and $F_{2(i)}$'s normal.
   (c) if v is of degree five, as in Figure 7, then
       $v_1 =$ the vertex through which the two triangulation edges pass.
       $I_p =$ the intersection point of the planes on which $F_{1(i)}, F_2$ and $F_{3(i)}$ reside.
       $B =$ the midpoint of the segment $v_1 - I_p$.
       $N =$ the average of the normals of $F_{1(i)}$ and $F_{3(i)}$.
2. **Determine r:**
   Let r be a ray whose base is B and whose direction is N.
3. **Transform the polyhedron if needed:**
   If v is of degree five and B is below $F_2(i)$ then
   Apply the appropriate projective transformation as described below.
   Recalculate r as described above.

So far we described the creation of r. We now explain how v is located along r. Consider the neighboring faces of v's cone of faces, $\{F_j \mid 1 \leq j \leq d\}$, where d is the degree of v. Consider also the planes on which these faces reside. Calculate the intersection points between these planes and the line that r defines $\{I_j = Plane(F_j) \cap Line(r) \mid 1 \leq j \leq d\}$. Find the d factors $\{t_j \mid I_j = B + t_j N,\ 1 \leq j \leq d\}$. Let

$$t = \begin{cases} 1 & \text{if } \forall j, t_j < 0 \text{ or } t_j = \infty \\ 0.5 * \min\{t_j \mid t_j > 0\} & \text{otherwise} \end{cases} \tag{1}$$

Position v at $B + tN$. Remove the faces $\{F_{j(i)} | 1 \leq j \leq d-2\}$, and construct d new faces between v and its neighbors, as illustrated in Figures 5–7. This completes the construction of P_{i+1}.

Lemma 3.6 *In case v is of degree three, it is positioned correctly. That is, v is above $F_{1(i)}$ and below F_j, for $\{j|1 \leq j \leq 3\}$.*

Proof: Since t is positive, v is above $F_{1(i)}$. It is left to show that v is below the neighbors of $F_{1(i)}$. We distinguish between the faces according to the sign of t_j. If t_j is positive, v is above F_j if and only if $t > t_j(> 0)$. If t_j is negative, v is above F_j if and only if $t < t_j(< 0)$. Our choice of t prevents v from being above F_j. In the first case this is because $t < t_j$, and in the second case this is because t and t_j do not have the same sign. If t_j is ∞, all the points on the ray r are on the same side of F_j. Since r intersects the polyhedron, the points of it are obviously below F_j. Therefore the only face above which v is $F_{1(i)}$. Since by our construction it is removed, the resulting P_{i+1} is convex. □

Lemma 3.7 *In case v is of degree four, it is positioned correctly.*

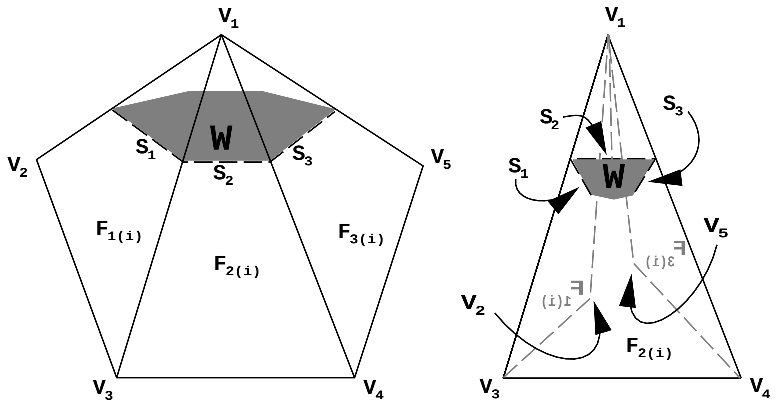
Proof: We need to show that v is above $F_{1(i)}$ and $F_{2(i)}$, and that it is below F_1, F_2, F_3 and F_4. Since N is the sum of the two normals of $F_{1(i)}$ and $F_{2(i)}$, it is clearly in acute angles to both. For this reason, any point on r is above both faces, and thus for these faces every choice of a positive t is acceptable. Moreover, since t does not have both the same sign as t_j and a larger absolute value, then similarly to the case of degree three, v is below $\{F_1, F_2, F_3 F_4\}$. The case of $t_j = \infty$ is also similar. □

In case v is of degree five, there are two possibilities. If we are lucky, B is above $F_{2(i)}$. In this case, B can be used to construct P_{i+1}, as was done in the case of degree four. Otherwise, we need to transform the polyhedron P_i prior to the positioning of B. We discuss the required projective transformation below.

Lemma 3.8 *In case v is of degree five and B is above $F_{2(n)}$, v is positioned correctly.*

Proof: The proof is similar to the previous case. Along with the assumption that B is above $F_{2(n)}$, we get the required result. □

The projective transformation: There are two possible configurations, as shown in Figure 8. Figure 8(a) illustrates an "open" configuration in which the sum of dihedral angles between $F_{1(i)}$, $F_{2(i)}$ and $F_{2(i)}$, $F_{3(i)}$ is larger than 180°. In Figure 8(b), illustrating a "closed" configuration, this sum is less than 180°.



(a) open (b) closed

Figure 8: Possible configurations

Recall that we assume that B is below $F_{2(i)}$. We describe below the transformation for the case where the configuration is "closed". The case of an "open" configuration is similar.

To define the transformation, we find a plane, W (the grey plane in Figure 8), which intersects P_i only in the faces adjacent to v_1. Such a plane can be easily found. The intersections of the plane with faces of the polyhedron are segments on W, denoted as S_1, S_2, S_3 (corresponding to $F_{1(i)}, F_{2(i)}, F_{3(i)}$ respectively). We first define a planar transformation on this plane, and then extend the required transformation to three dimensions.

After finding W, we apply an affine transformation Aff such that: (1) Aff transforms W to the plane $Z = 0$. (2) Aff transforms S_1 to the Y axis, to part of the segment $((0,0,0),(0,1,0))$. (3) Aff transforms S_3 to the X axis, to part of the segment $((0,0,0),(1,0,0))$. (4) Aff transforms the intersection of the lines on which S_1, S_3 reside (denoted as I_{S_1,S_3}), to the origin. The situation on $Aff(W)$ (i.e., on $Z = 0$) is illustrated in Figure 9(a).

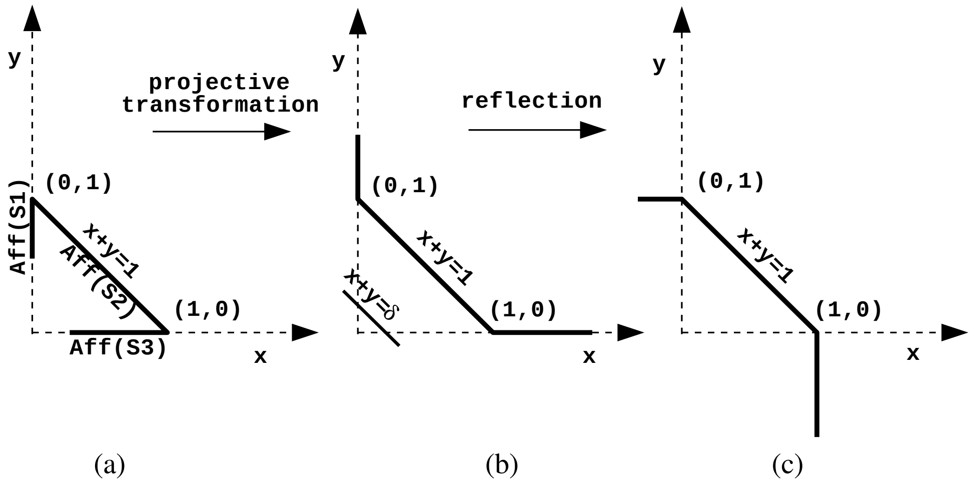


Figure 9: The projective transformation

To complete the definition of the affine transformation, it is left to specify the reciprocal image of the Z axis. To do it, we define a new plane, a *separating plane*, that (1) separates I_{S_1,S_3}, the intersection of the lines on which S_1 and S_3 reside, from the polyhedron and (2) is parallel to S_2. Such a plane can be easily found. Once it is found, we can define the reciprocal image of the Z-axis to be a line on the separating plane which is not on W. From now on, we assume that the affine transformation has already been applied.

Figure 9(c) illustrates the intuition behind the projective transformation we are seeking. It should "open" the faces $F_{1(i)}$ and $F_{3(i)}$. And in two dimensions (on $Z = 0$), it should "open" the segments $Aff(S_1)$ and $Aff(S_3)$. In other words, the purpose of the projective transformation is to transform $Aff(S_1)$ and $Aff(S_3)$ such that the point of intersection of their lines is transformed to the other side of $Aff(S_2)$ (and in three dimensions, to the other side of $F_{2(i)}$). Our intention is to 'open" the configuration and to transform B to reside above $F_{2(i)}$.

In two dimensions, two planar transformations are defined as illustrated in Figure 9. The first is a planar projective transformation (Figure 9(b)), and the second is a planar reflection around the line $x + y = 1$ (Figure 9(c)). The planar projective transformation is defined by $T(V) = \frac{V(1-\delta)}{V_x+V_y-\delta}$ where V is a two–dimensional vector, and δ is small enough to separate the origin from $P_i \cap W$. To illustrate that this transformation performs the required opening, note that: (1) The line $x + y = 1$ is unchanged by the transformation. (2) The segment $((0,\delta),(0,1))$ is transformed to $((0,\infty),(0,1))$. (3) The open segment $((\delta,0),(1,0))$ is transformed to $((\infty,0),(1,0))$. The extension to three dimensions is obtained by $T(V) = \frac{V(1-\delta)}{V_x+V_y-\delta}$ where the equation $x + y = \delta$ represents the separating plane.

Lemma 3.9 *The above projective transformation, T, transforms I_p, the point of intersection of the three planes on which the faces $F_{1(i)}, F_2, F_{3(i)}$ reside, and thus B, from one side of the plane of $F_{2(i)}$ to the other side.*

Proof: It can be easily shown that when T is applied to the convex polyhedron P_i, a convex polyhedron results, if the polyhedron resides on one side of the plane defined by the denominator of the projective transformation. This requirement is satisfied by our choice of the separating plane. The resulting polyhedron also has an isomorphic vertex graph, because the vertices of the transformed convex hull are images of the vertices of the original convex hull. Moreover, the transformation T preserves both the boundary of P_i (which is is transformed to the boundary of $T(P_i)$), and the intersection points between planes. It remains to be shown that this intersection, $T(I_p)$, resides above $T(F_{2(i)})$.

Let L_F be the line of intersection of the two faces $F_{1(i)}$ and $F_{3(i)}$. The three points v_1, I_{S_1,S_3}, and I_p reside on this same line, L_F. Now we examine $T(L_F)$ more closely. T is continuous on L_F, except for the intersection of L_F with the separating plane. This intersection is a single point, since the points on L_F were established to be on both sides of the separating plane — v_1 on the same side as the polyhedron, I_{S_1,S_3} on the other side. Moreover, I_{S_1,S_3} and I_p reside on the same side of the separating plane (since we could choose W so that I_p is farther away from I_{S_1,S_3} on L_F).

T is continuous on the whole side of the separating plane which contains I_{S_1,S_3} and I_p. We conclude that $T(v_1)$ cannot be between $T(I_{S_1,S_3})$ and $T(I_p)$, since the segment between I_{S_1,S_3} and I_p is transformed continuously. Thus $T(I_{S_1,S_3})$ and $T(I_p)$ are on the same side of $T(L_F)$ relative to $T(v_1)$.

I_{S_1,S_3} is transformed by T from one side to the other side of $T(S_2)$, and thus of $T(F_{2(n)})$, because the intersection of S_1 and S_3 before the projective transformation is at the origin, and after applying T at $(1, 1, 0)$. Since I_p and I_{S_1,S_3} were transformed to the same side of $T(L_F)$ relative to $T(v_1)$, they were also transformed to the same side of $T(F_{2(n)})$, thus correcting the position of I_p. $\square$

Theorem 3.10 *Given a convex polyhedron P_i, after re-attaching v as described above, the resulting polyhedron P_{i+1} is convex.*

Proof: By our construction, there are three possible cases: v of degree three, four, or five. If v is of degree three, by Lemma 3.6, the resulting polyhedron is convex. If v is of degree four, by Lemma 3.7, the resulting polyhedron is convex. If v is of degree five, there are two cases. If B is above $F_{2(n)}$, by Lemma 3.8, the resulting polyhedron is convex. Otherwise, we apply the projective transformation as defined above. By Lemma 3.9, after applying this transformation, the case is reduced to the previous case, and thus a convex polyhedron can be constructed. $\square$

Theorem 3.11 *The realization algorithm described above runs in $O(N^2)$ time, where N is the number of vertices in the polyhedron.*

4 Merging Vertex Neighborhood Graphs

Given two convex polyhedra $P_{|P|}$ and $Q_{|Q|}$, our goal is to construct two convex polyhedra $\tilde{P}$ and $\tilde{Q}$, such that: (1) The vertex–neighborhood graphs of $\tilde{P}$ and $\tilde{Q}$ are isomorphic to each other. (2) $\tilde{P}$ is a refinement of $P_{|P|}$. (3) $\tilde{Q}$ is a refinement of $Q_{|Q|}$. The algorithm works as follows.

1. **Initialize $\tilde{P}$ and $\tilde{Q}$:**
   - Let $\tilde{P} \leftarrow P_{|P|}$ and $\tilde{Q} \leftarrow Q_{|Q|}$.
   - Move the centers of mass of $\tilde{P}$ and $\tilde{Q}$ into the origin.
2. **Update the vertex set of $\tilde{P}$:**
   For each vertex v of $\tilde{Q}$
   - Shoot a ray from the origin through v.
   - If the ray does not intersect $\tilde{P}$ in a vertex, add the intersection to the vertex set of $\tilde{P}$.
3. **Create the edge set of $\tilde{P}$ and update its vertex set:**
   For each edge e of $\tilde{Q}$

- Create the plane Pl that passes through e and through the origin O.
- The intersection of Pl and $\tilde{P}$ is a two–dimensional convex polygon. Sort the vertices according to their order in the polygon.
- The polygon consists of two polygonal arcs between the vertices that correspond to the endpoints of e. Represent the edge e in $\tilde{P}$ by the shorter arc.
- Traverse the polygonal arc, adding its vertices and its edges to the sets of vertices and edges of $\tilde{P}$ respectively.
- Replace every edge of $\tilde{P}$ which is split this way by its parts.

4. **Create the face set of** $\tilde{P}$ (given the vertices and the edges found above).
5. **Update the vertex set of $\tilde{Q}$:**
   For each vertex w of $\tilde{P}$:
   - Shoot a ray from the origin through w.
   - If the ray does not intersect $\tilde{Q}$ in a vertex, add the intersection to the vertex set of $\tilde{Q}$.
6. **Create the edge set of $\tilde{Q}$:**
   For each edge e of $\tilde{P}$
   - Create an edge between the two vertices in $\tilde{Q}$, which correspond to the vertices of e.
7. **Create the face set of $\tilde{Q}$.**
8. **Determine the location of the vertices of one polyhedron on the surface of the other:**
   This mapping is done using barycentric coordinates.

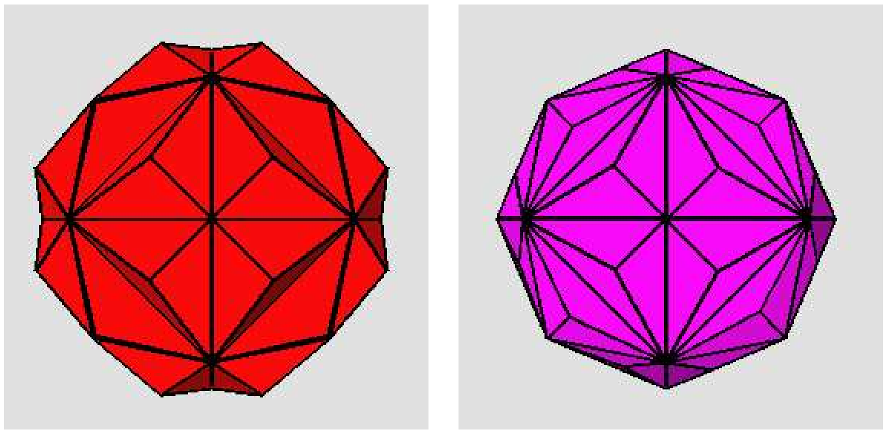
5 Implementation and Results

The algorithm is implemented in C and runs on SGI workstations. Though the complexity of the realization process is $O(N^2)$, without the projective transformation, the complexity is $O(N)$. The projective transformation is seldom necessary. As a result, the algorithm is very efficient and it runs very fast.

Figures 10 – 11 show a couple of results of the realization algorithm. In Figure 10 an epcot and its realized epcot are being presented. In Figure 11 a deformed cube and its realized polyhedron is shown. The cube was created by “pushing” one face inside, so that it is ϵ-away from its opposite face. This face cannot remain planar, so we allow it to “bend”, and we re-triangulate the resulting polyhedron. The deformed-cube example, though small in size, illustrates our realization algorithm very well. The cube does not belong to any of the classes supported by previous algorithms. It is non-convex, not star-shaped, not an object of revolution, and not an extruded object.

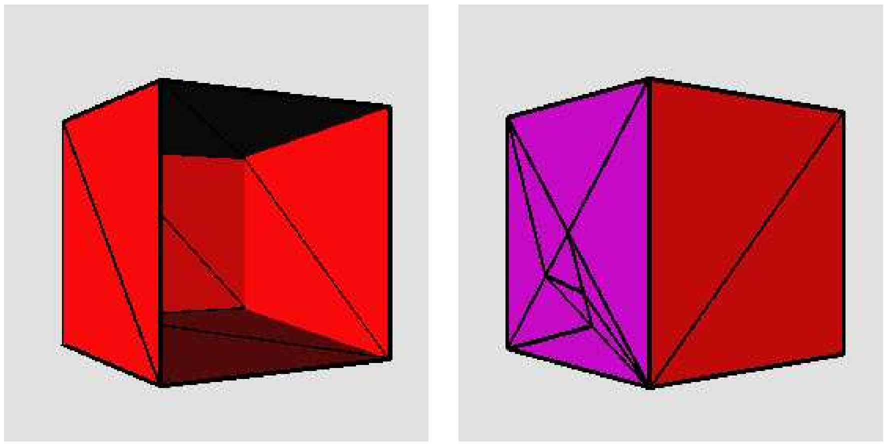
Figure 12 illustrates the results of the merge algorithm. In Figure 12(a) the vertex–neighborhood graphs of a house and an icosahedron are merged on the surface of the icosahedron. In Figure 12(b) two deformed cubes are merged on the surface of one of them. The pair of cubes have opposite faces pushed in.

Figures 13–17 present a few of morph sequences created by utilizing the realization algorithm. Figures 13–14 show the transformation between a pair of deformed cubes with opposite faces pushed in. During interpolation, the deformed face is pushed out, while the opposite one is pushed in. We use both wire–frame and faceted shading in order to illustrate the above process. Figures 15 – 16 transform a house into an icosahedron. While Figure 15 displays a few snapshots from the morph sequence in a smooth shading, Figure 16 shows some snapshots in a faceted shading. Note how the stairs and the roof disappear in the icosahedron’s surface. Finally, Figure 17 transforms the house into a bottle. None of the objects is star-shaped.



(a) An epcot (b) A realized epcot

Figure 10: Realization of an epcot



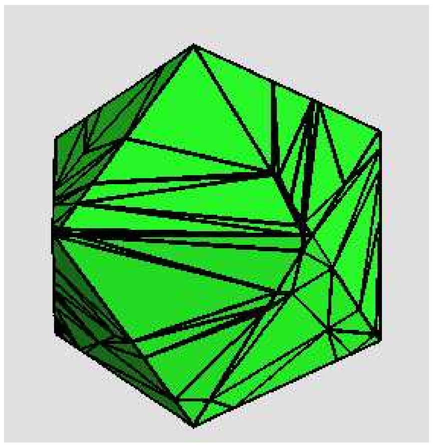
(a) A deformed cube (b) A realized deformed cube

Figure 11: Realization of a deformed cube

6 Conclusion

This paper presents a new algorithm for realizing a three–dimensional zero–genus polyhedron. Our algorithm is general, efficient, and easy to implement. We demonstrated how the algorithm can be used for finding a correspondence between two three–dimensional polyhedra for metamorphosis.

There are a few directions for future research. One interesting direction engages human intervention. The current algorithm involves almost no user control. While an automatic technique is desirable in many applications, at certain times, it is beneficial to provide finer controls. Second, an intriguing problem is to find an algorithm for”inflating” non zero–genus polyhedra. Finally, a challenging problem is to find a morph algorithm between polyhedra that can guarantee that self–intersection does not occur during interpolation.


(a) A house and an icosahedron

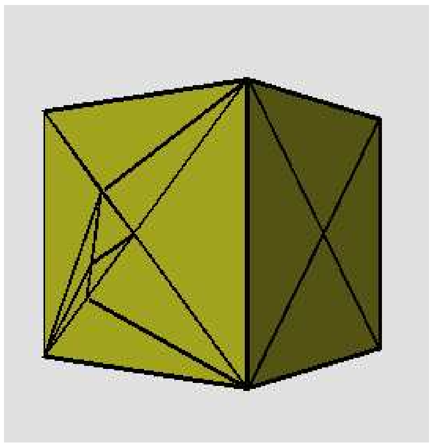

(b) Two deformed cubes

Figure 12: Merging the vertex–neighborhood graphs

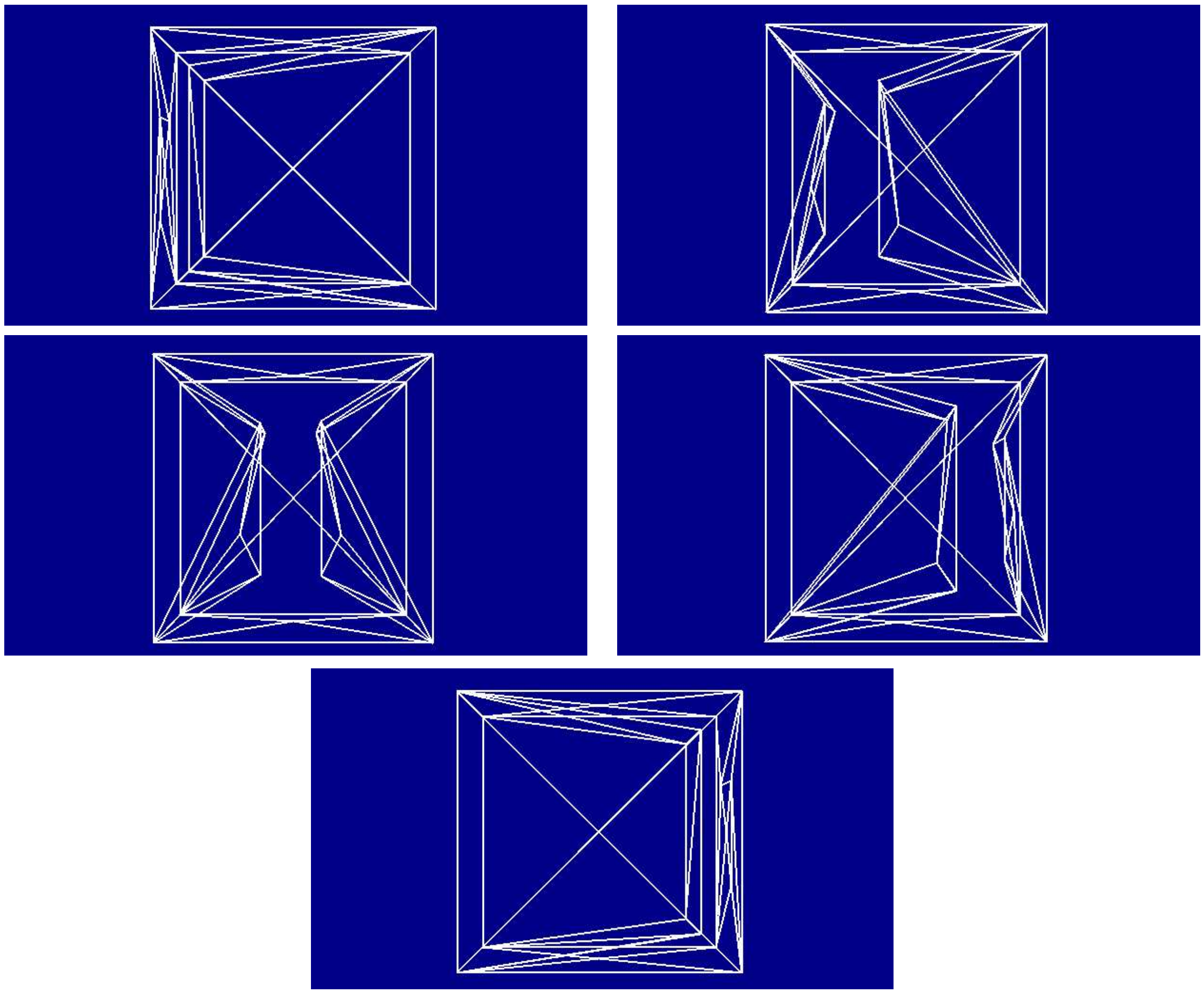


Figure 13: Shape transformation between the deformed cubes – wire–frame

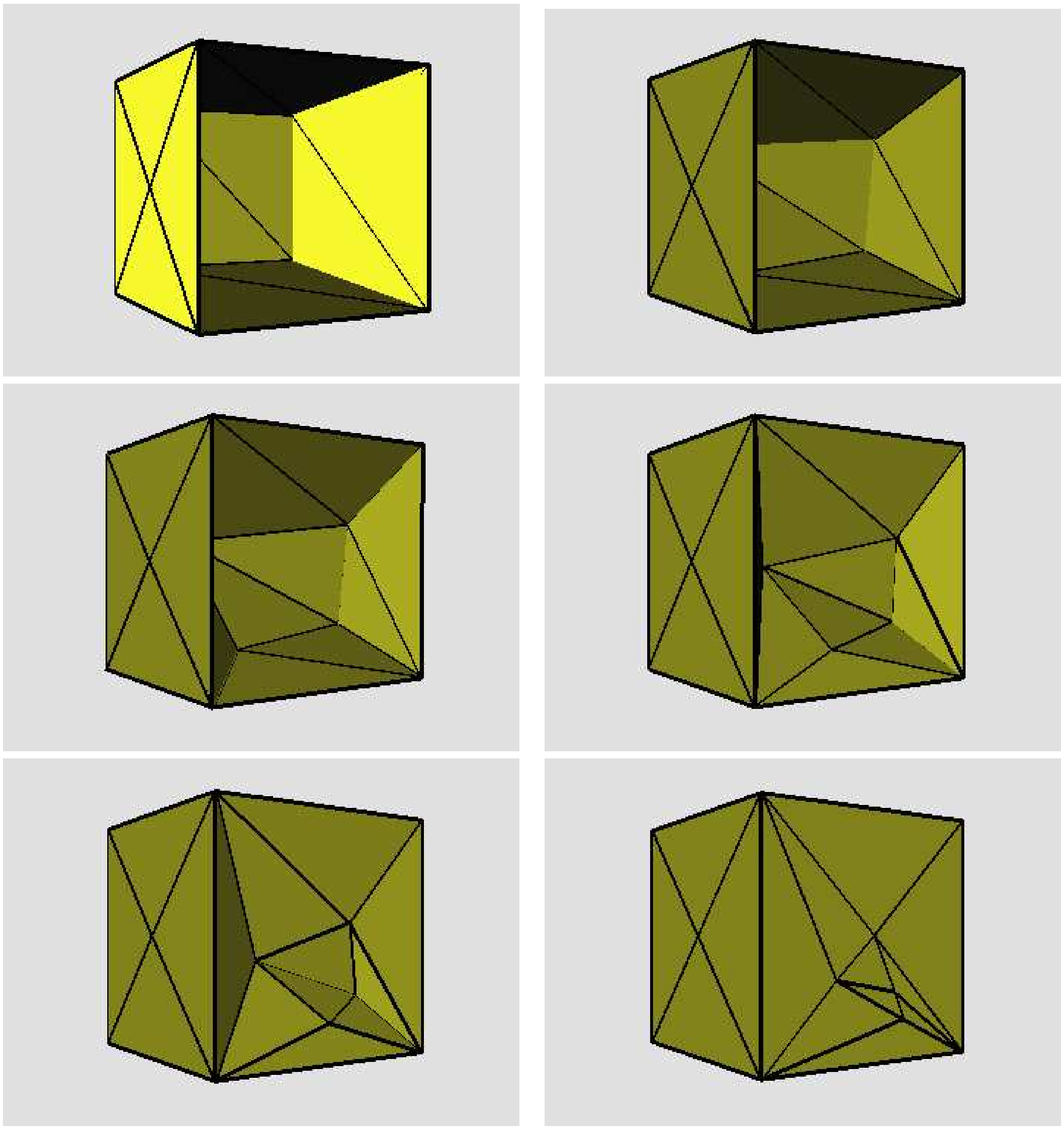


Figure 14: Shape transformation between the deformed cubes – faceted shading

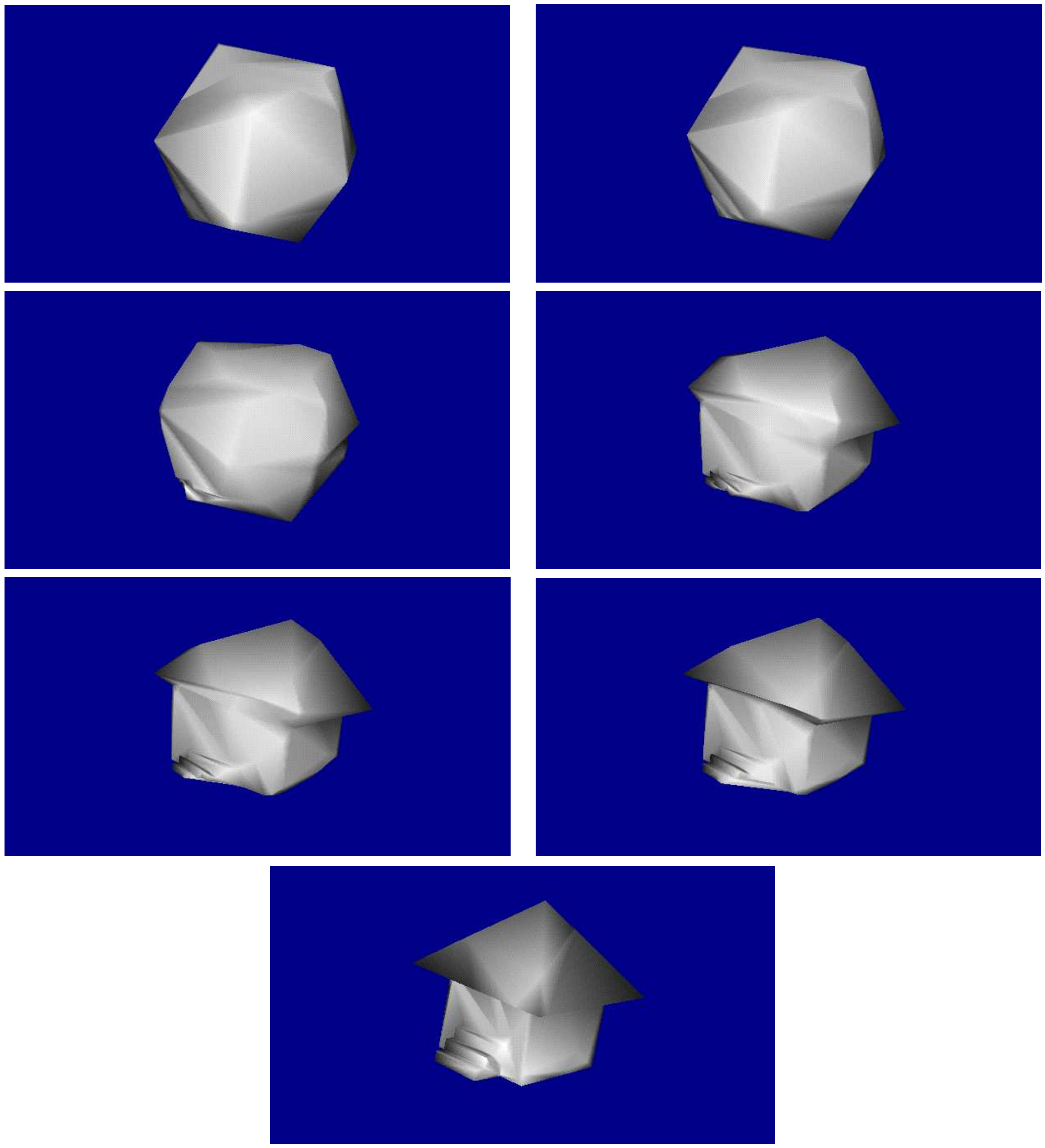


Figure 15: Shape transformation between a house and an icosahedron - smooth shading

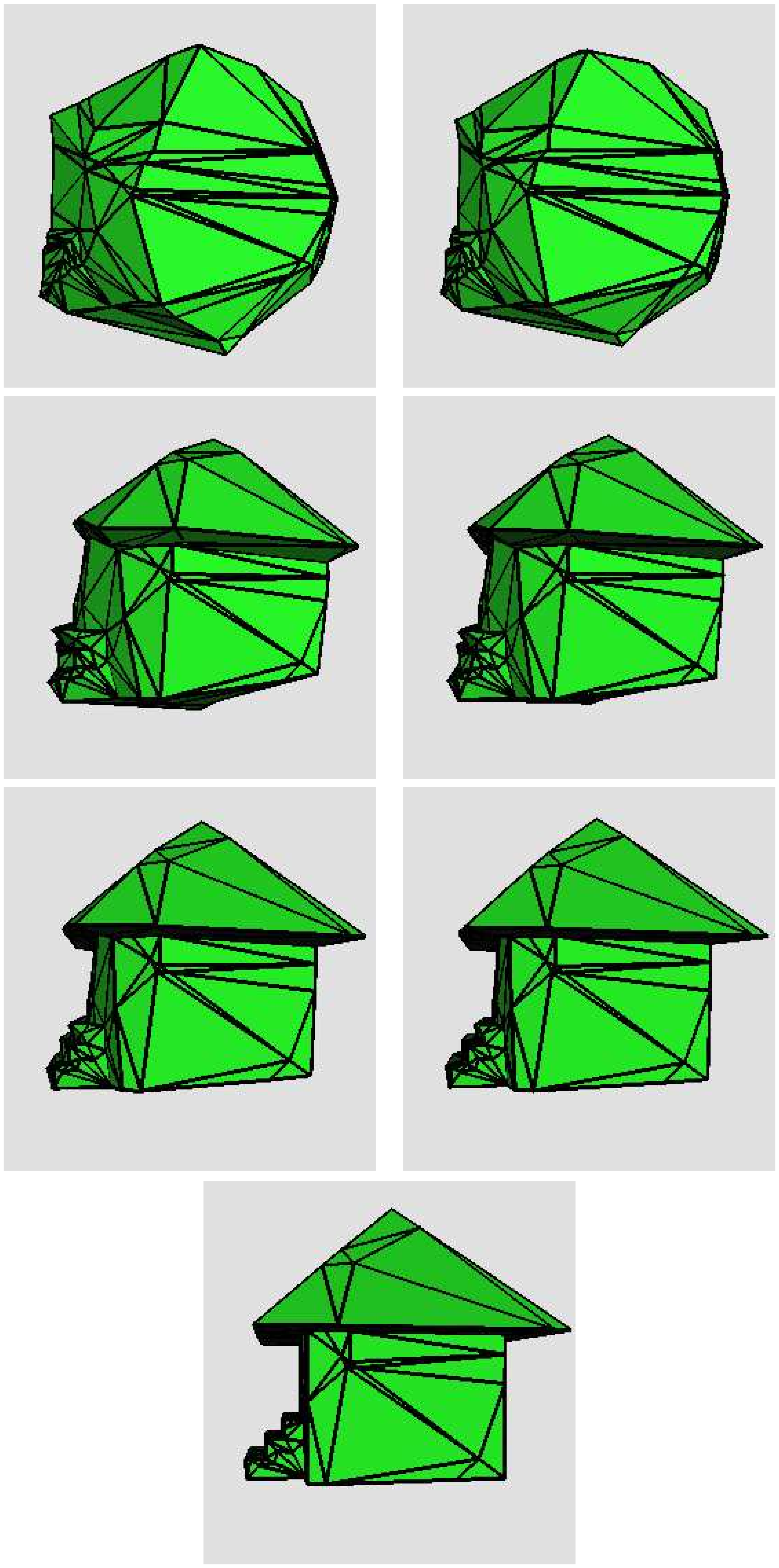


Figure 16: Shape transformation between a house and an icosahedron – faceted shading

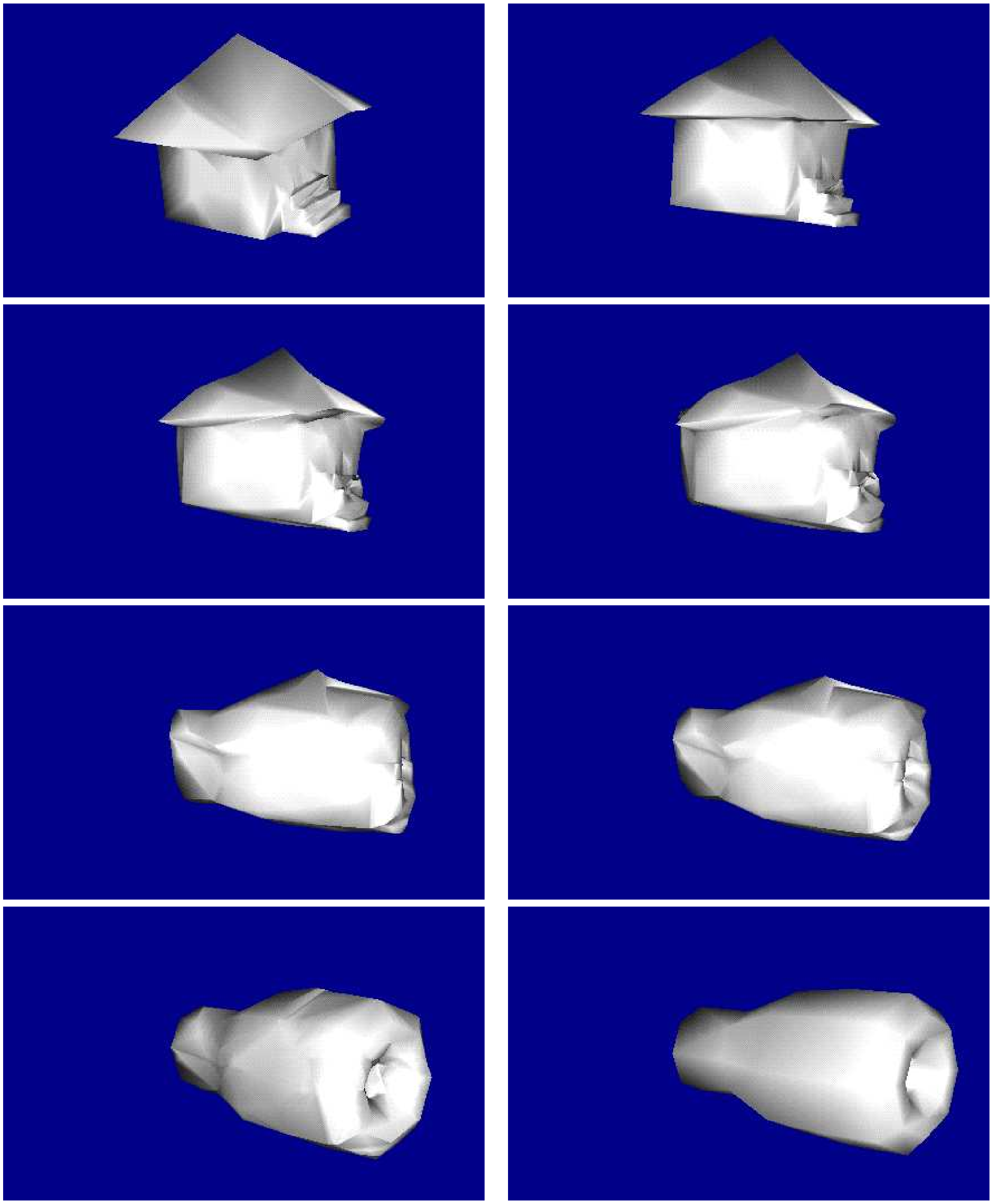


Figure 17: Shape transformation between a house and a bottle

References

[1] B. Aronov, R. Seidel, D. Souvaine. On compatible triangulations of simple polygons. *Computational Geometry: Theory and Applications* 3, 1993, pages 27–35.

[2] T. Beier and S. Neely. Feature–based image metamorphosis. In *Proceedings of SIGGRAPH '92*, volume 26(2), pages 35–42. June 1992.

[3] E. Bethel and S. Uselton. Shape distortion in computer–assisted keyframe animation. In State of the Art in Computer Animation N. Magnenat-Thalmann and D. Thalmann, Eds., Springer-Verlag, NY, 1989, pages 215–224.

[4] E. Carmel and D. Cohen-Or. Warp–guided object–space morphing. *The Visual Computer,* 13:(9+10), pages 465–478, 1997.

[5] E. Chen and R. Parent. Shape averaging and its applications to industrial design. *IEEE Computer Graphics and Applications*, 9(1):47–54, January 1989.

[6] D. Cohen-Or, D. Levin and A. Solomovici. Contour blending using warp–guided distance field interpolation. In *Proceedings of Visualization '96*, pages 165–172, October 1996.

[7] D. Cohen-Or, D. Levin and A. Solomovici. Three dimensional distance field metamorphosis. In *ACM TOG*, April 1998.

[8] D. Dobkin and D. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *J. of Algorithms*, 6, pages 381–392, 1985.

[9] D. Dobkin and D. Kirkpatrick. Determining the separation of preprocessed polyhedra – a unified approach. *ICALP*, pages 400–413, 1990.

[10] E. Goldstein and C. Gotsman. Polygon morphing using a multiresolution representation. *Graphic Interface '95*, 247–254, 1995.

[11] B. Grunbaum. Convex Polytopes. *Interscience Publishers*, 1967.

[12] L. Guibas and J. Hershberger. Morphing simple polygons. *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, 267–276, 1994.

[13] T. He, S. Wang, A. Kaufman. Wavelet–based volume morphing. Proceedings of Visualization '94, R. D. Bergeron, A. E. Kaufman, Eds. October 1994, pages 85 – 91.

[14] T. Hong, N. Magnenat-Thalmann, and D. Thalmann. A general algorithm for 3–D shape interpolation in a face-based representation. In *Proceedings of Graphics Interface '88*, June 1988, pages 229–235.

[15] J.F. Hughes. Scheduled Fourier volume morphing. *Computer Graphics*, 26(2):43–46, July 1992.

[16] T. Kanai, H. Suzuki and F. Kimura. 3D geometric metamorphosis based on harmonic maps. *Proceedings of Pacific Graphics '97*, 97–104, October 1997.

[17] A. Kaul and J. Rossignac. Solid–interpolation deformations: Construction and animation of pips. In *Proceedings of EUROGRAPHICS'91*, pages 493–505, September 1991.

[18] J.R. Kent, R.E. Parent, and W.E. Carlson. Establishing correspondences by topological merging: A new approach to 3D shape transformation. In *Proceedings of Graphics Interface 91*, July 1991, pages 271 – 278.

[19] J.R. Kent, W.E. Carlson, and R.E. Parent. Shape transformation for polyhedral objects. *Computer Graphics*, 26(2):47–54, July 1992.

[20] A. Lerios, C. D. Garfinkle, and M. Levoy. Feature-based volume metamorphosis. In *Proceedings of SIGGRAPH '95*, pages 449–456. ACM, August 1995.

[21] R.E. Parent. Shape transformation by boundary representation interpolation: a recursive approach to establishing face correspondences. *Journal of Visualization and Computer Animation*, 3:219–239, 1992.

[22] F.P. Preparata and M.I. Shamos. Payne. Computational geometry – an introduction. *Springer–Verlag*, 1985.

[23] B.A. Payne and A.W. Toga. Distance field manipulation of surface models. *IEEE Computer Graphics and Applications*, 12(1):65–71, January 1992.

[24] T.W. Sederberg, P. Gao, G. Wang, and H. Mu. 2–D shape blending: An intrinsic solution to the vertex path problem. In *Proceedings of SIGGRAPH '93*, volume 27, pages 15–18. ACM, August 1993.

[25] M. Shapira and A. Rappoport. Shape blending using the star–skeleton representation. *IEEE Computer Graphics and Applications*, 15:44–50, March 1995.

[26] Y.M. Sun, W. Wang, and F.Y.L. Chin. Interpolating polyhedral models using intrinsic shape parameters. In S.Y. Shin and T.L. Kunii, editors, *Proceedings of the Third Pacific Conference on Computer Graphics and Applications, '95*, pages 133–147. World Scientific, 1995.

[27] C. Terzides. Transformational design. *Knowledge Aided Architectural Problem Solving and Design*. NSF Project #SMC 8609893, June 1989.

[28] G. Wolberg. Digital Image Warping. *IEEE Computer Society Press*. 1990.