IASS 2018 BOSTON, JULY 16-20
CREATIVITY IN STRUCTURAL DESIGN

Proceedings of the IASS Annual Symposium 2018
Creativity in Structural Design
July 16-20, 2018, MIT, Boston, USA
Caitlin Mueller, Sigrid Adriaenssens (eds.)

Constrained Manipulation of Polyhedral Systems

Andrei NEJUR*, Masoud AKBARZADEH[a]

*University of Pennsylvania School of Design, Polyhedral Structures Laboratory
Pennovation Center, 3401 Grays Ferry Ave.
Philadelphia, PA, 19146
nejur@upenn.edu

Abstract

This paper presents a method for the manipulation of groups of polyhedral cells that allows geometric transformation while preserving the planarity constraints of the cells and maintaining the equilibrium direction of the edges for the reciprocity of the form and force diagrams. The paper expands on previously investigated single-cell manipulations and considers the effects of these transformations in adjacent cells and the whole system. All the transformations discussed in this paper maintain the initial topology of the input system. The result of this research can be applied to both form and force diagrams to investigate various geometric transformations resulting in convex or complex (self-intersecting) polyhedra as a group. The product of this research allows intuitive user interaction in working with form and force diagrams in the early stages of geometric structural design in 3D.

Keywords: 3d Graphic Statics, Polyhedron, Manipulation, Form, Force, 3DGS

1. Introduction

3D Graphic Statics (3DGS) based on reciprocal polyhedral diagrams is one of the promising structural design methods that has recently been introduced and is currently being developed by multiple researchers around the world (Akbarzadeh [1,2,3], Konstantatou [6] McRobie [9]). This method relies on polyhedral systems to describe equilibrium of complex force configurations in three dimensions. Modeling and manipulating polyhedral geometry in the context of 3D Graphic Statics and reciprocal polyhedral diagrams, either as the form or force diagram, are not a trivial tasks. The construction of dual-reciprocal convex polyhedral diagrams was addressed in [1]. However, the research was limited to convex cells and did not provide much insight into polyhedral manipulation. Although polyhedral reconstruction is a well-researched topic with results covering multiple field applications (Demaine and O'Rourke [4], Ikeuchi [5], Moni [10]) most previous research has addressed single cell systems. Recently Lee at al [7], have shown a method for constructing single polyhedra based on face normal directions and target face areas. Previously, the same researchers, in [8] presented a workflow for modifying multiple polyhedrons as a 3DGS force diagram but without conserving the reciprocal relationship with the form diagram. To the extent of our knowledge no method for manipulating groups of polyhedrons with constant normal direction for the faces has been proposed prior to the writing of this paper.

1.1. Problem statement

The reciprocal relationship between a polyhedral system and its dual in 3DGS, is the equivalent of an equilibrium of forces (represented by the first system) inside a structural form (represented by the dual). This equilibrium is comprised of two components, a topological and a geometrical one. The topological component ensures a one to one pairing between the number of forces and the elements of the form. The geometry component further refines the relationship by enforcing perpendicularity between primal (force) faces and dual (form) edges according to the conventions proposed by Rankine [11]. The topological component of the relationship, although not trivial, is relatively straight-forward to establish. The geometric component can be addressed second and is usually more complicated to establish and much more prone to breaking in the event of unconstrained change occurring in any of the polyhedral

structures. The process of establishing the geometrical link of the reciprocal relationship usually requires iteration over the topology [1] or the calculation of a stress function as a higher dimension polytope [6].

Often, the methods used to establish the reciprocal relation prohibit further exploration and constraint based transformation of the resultant form. A transformation based on a full reconstruction of the system can be attempted. But depending on the speed of the process and the complexity of the data, the manipulation can be a very cumbersome and often impossible task especially if it needs to be repeated multiple times. By examining the geometry rules that make these reciprocal constructs possible, it is obvious that they are not simply singular events, but instances of a solution space. This means that if the geometry rules are respected and enforced, a process for the exploration of a solution space for a polyhedral system, locked into a reciprocal relationship with its dual, can be created.

1.2. Objectives

The present research aims to establish a workflow that would allow for the manipulation of a polyhedral system, in a state of static equilibrium, without breaking the reciprocal relationship with its dual, and while keeping all of the faces of the cells planar. Moreover, the method emphasizes on preserving the convexity of the cells during and after the transformation. See Fig.1 for an example.

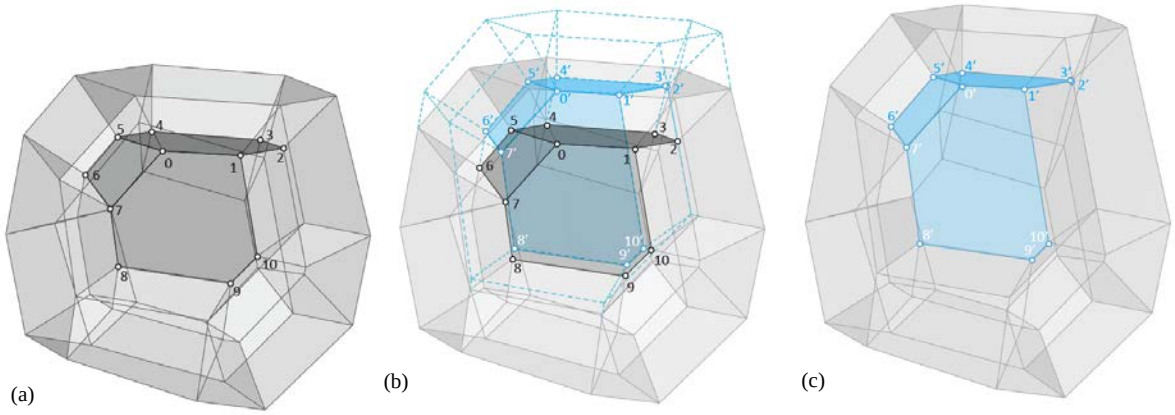


Figure 1. Transformation of a simple polyhedral form.

2. Theoretical framework.

The start of the transformation can be one, or a set of vertices in the group of polyhedral cells. Any input for the manipulation of the polyhedral group can be viewed as a translation of a set of vertices. In Fig. 1, the offset of faces $f_{(0,1,2,3,4,5)}$, $f_{(0,1,10,9,8,7)}$ and $f_{(0,7,6,5)}$ is the result of the translation of vertex v_0 to position $v_{0'}$. Any vertex or set of vertices from these faces can be considered an input for the transformation process. The transformation works on a graph representation of the polyhedral group. In the graph representation of the polyhedral group, the nodes of the graph are the vertices of the polyhedrons and the edges of the graph are the edges of the polyhedrons. Creating the sequence of geometrical transformations in the polyhedral system is equivalent to converting the graph into a tree graph. We consider the root of the tree graph, the first moved vertex in the polyhedral cluster. The input vertices are moved to the new locations and from them the transformation is followed from vertex to vertex via the edges until all vertices in the tree have been parsed. We call this tree graph a *transformation path*. For each complete transformation of a polyhedral group, based on the same input of moved vertices, the transformation paths are not unique. Each path produces a unique transformation result for a given set of inputs. A single vertex transformation (the root) is enough to start the path. Any other input vertices will have to be in positions compatible with the transformation path for the transformation to succeed. The transformation process works in steps starting from the root and expands outwards, calculating each new transformed vertex based on its connections in the tree of transformations and in the original polyhedral cluster graph.

2.1. Transformation propagation

This concept ensures that at every step, the transformation is propagated. It uses the connectivity of the original polyhedron topology to extend the transformation from the translated vertex, to all the edges that connect to it and from there, to all the faces that contain those edges. By attempting to fully determine those edges, new vertices are translated and thus the transformation propagates further.

2.1.1. Vertices registering edges and faces

Every moved vertex (if parallelism is preserved) causes a number of edges to be reconstructed as a direction in space anchored to the moved vertex point. Those edges, even though not fully reconstructed, in turn, trigger the reconstruction of all the faces that contain them. In Fig. 2.c, a vertex is moved from point p_0 to $p_{0'}$. As a result, all original edges passing through vertex v_0 namely $e_{(0,1)}$, $e_{(0,2)}$ and $e_{(0,4)}$, are partially reconstructed as parallel directions anchored to $p_{0'}$. For the purpose of this discussion we will use the notations for the transformed edges as $e_{(0',1')}$, $e_{(0',2')}$ $e_{(0',4')}$ event though points $p_{1'}$ $p_{2'}$ and $p_{4'}$ have not yet been determined, and some of them might end up in a different position than the one presented here when the full transformation of the polyhedron group is completed. What is important in our case is the direction of the edge in space and its anchoring position, point $p_{0'}$. We will call these partially-reconstructed, transformed edges, *partial edges*. In a similar fashion, all faces containing the edges mentioned above will be named using the points that uniquely identify them both in their original and their transformed state. Two partial edges sharing a point define the partial face as $f_{(1',0',2')}$ or $f_{(2'0',4')}$.

2.1.2. Closing the transformation loop

All those partial elements need to be fully determined moving forward, so the immediate goal is to find the position for the vertices at the other end of the partial edges. Therefore, any determined vertex has two main roles: a) to complete multiple partial edges and add to multiple partial faces that eventually get completed too, and b) to generate a new set of partial edges and faces that ensure the transformation loop continues until all the elements of the polyhedral group are parsed.

2.2. Transformation prioritization

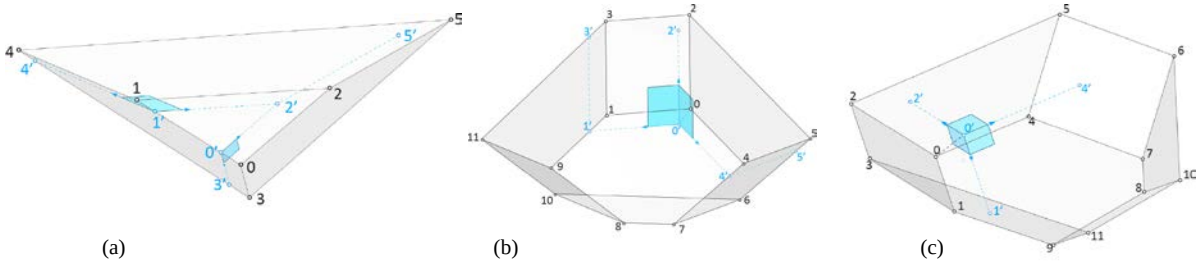
The second concept deals with controlling the perpetuation of a change inside the polyhedral cluster for the scope of maintaining a coherent transformation path. A coherent transformation path is a tree on the polyhedral group graph that is bound by a set of priority rules. These rules ensure that a fully transformed edge will stay parallel to its original direction, even though its vertices are calculated in different branches of the path at very different steps of the transformation. Transformation prioritization is also necessary to obtain additional geometric information for the calculation of some vertices. For this, some elements must wait in a queue until additional transformed geometry becomes available for the calculation. All transformed vertex calculations are achieved through intersections. Transformation prioritization works by examining all available intersections at any given step of the process and sorting them in three main categories to determine the order they will be worked in.

2.2.1. Intersection types and priority

All intersections are either between two coplanar partial edges, or between a partial edge (edge direction) and an original face-plane. We call these coplanar line intersections, *in-face* intersections and the line-plane intersections, *out-of-face* intersections. For the purpose of prioritizing, we also need to distinguish between two subtypes of *in-face* intersections. First, we have intersections between two partial edges in the same face. We call those *active intersections* because both edges actively seek to pinpoint their other vertex location. Second, we have intersections between one partial or active edge and one original untransformed edge. We name those *passive intersections*. In order to keep the transformation process coherent, the *active intersections* must be completed first, *passive intersections* second and *out-of-face intersections* third.

An active intersection example is shown in Fig. 2.a. The transformation of two vertices v_0 and v_1 in the same face, triggers the creation of two coplanar partial faces $e_{(0',2')}$ and $e_{(1'2')}$ that connect to the same topological yet undetermined vertex $v_{2'}$. If they would not be recognized as *active* both edges could be

completed through individual intersections with $e_{(1,2)}$ and $e_{(0,2)}$ respectively. This would create and ambiguous determination for $v_{2'}$ with two transformed positions.



(a) (b) (c)

Figure 2. Intersection types.

Figure 2.b shows multiple typical *passive in-face* intersections. Starting from transformed $v_{0'}$ the position for $v_{1'}$ and $v_{4'}$ are determined by intersecting *partial edges* with coplanar *original edges*. One step further $v_{3'}$ and $v_{5'}$ are determined in the same way. It should be noted that *passive intersections* have medium priority and can be pursued only if there are no *active intersections* available.

Out-of-face intersections are exemplified in Fig. 2.c. This type of intersection can be instantiated by a transformed vertex moved outside of all original faces topologically connected to its original self. In this case, the position of any of the transformed vertices $v_{1'}$, $v_{2'}$ or $v_{4'}$ can only be determined through a line-plane intersection between a *partial edge* and the plane of an original face topologically connected to the original position of the vertex about to be determined. Only one *out-of-face* intersection can be determined at a time, since the vertex position resulting from the operation is always resting on one original face plane. As a result, all subsequent intersections will be *in-face* and thus have higher priority than all other remaining *out-of-face* ones.

2.3. The implementation.

In order to test the viability and robustness of the described workflow, the previously described concepts and rules were implemented as part of the 3DGS research and development framework, created at the Polyhedral Structures Lab at University of Pennsylvania. The framework was developed as an extension for the popular 3d modeling program Rhinoceros. All the figures presented in this paper are directly derived from the explorations undertaken with the implemented solution. The algorithm is written as a modified BFS (Breadth First Search), with multiple inner loops and an expansion behavior based on the *transformation prioritization* rules described in the previous sections. In a similar fashion to the theoretical workflow decomposed above, the algorithm starts from a given set of vertex translations and reconstructs *partial edges and faces*. The step by step completion of those elements through various intersection techniques yields additional transformed vertices. Those vertices bring more edges and faces in the transformation process closing a loop, that only ends when there are no more elements to be transformed. Figure 3 shows a high-level overview of the algorithm's most important steps.

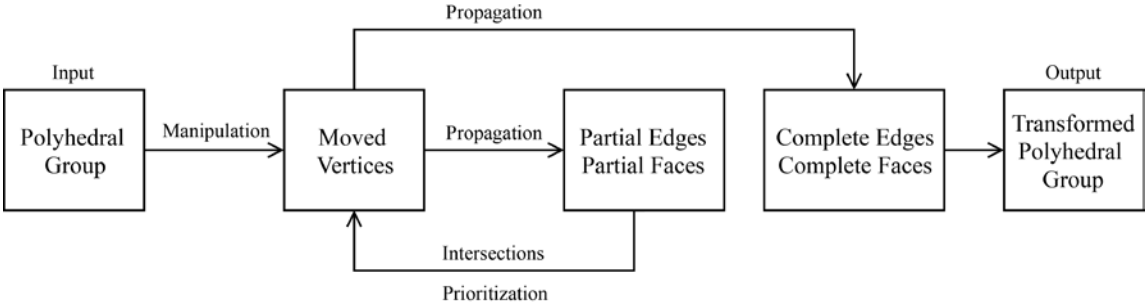


Figure 3. Overview of the implemented algorithm

The implemented version of the algorithm is based on a custom data structure capable to encode the various stages of the transformation. The tool is capable to transform large polyhedral sets with hundreds of cells and thousands of vertices while giving live visual feedback to the user.

2.4. A transformation example.

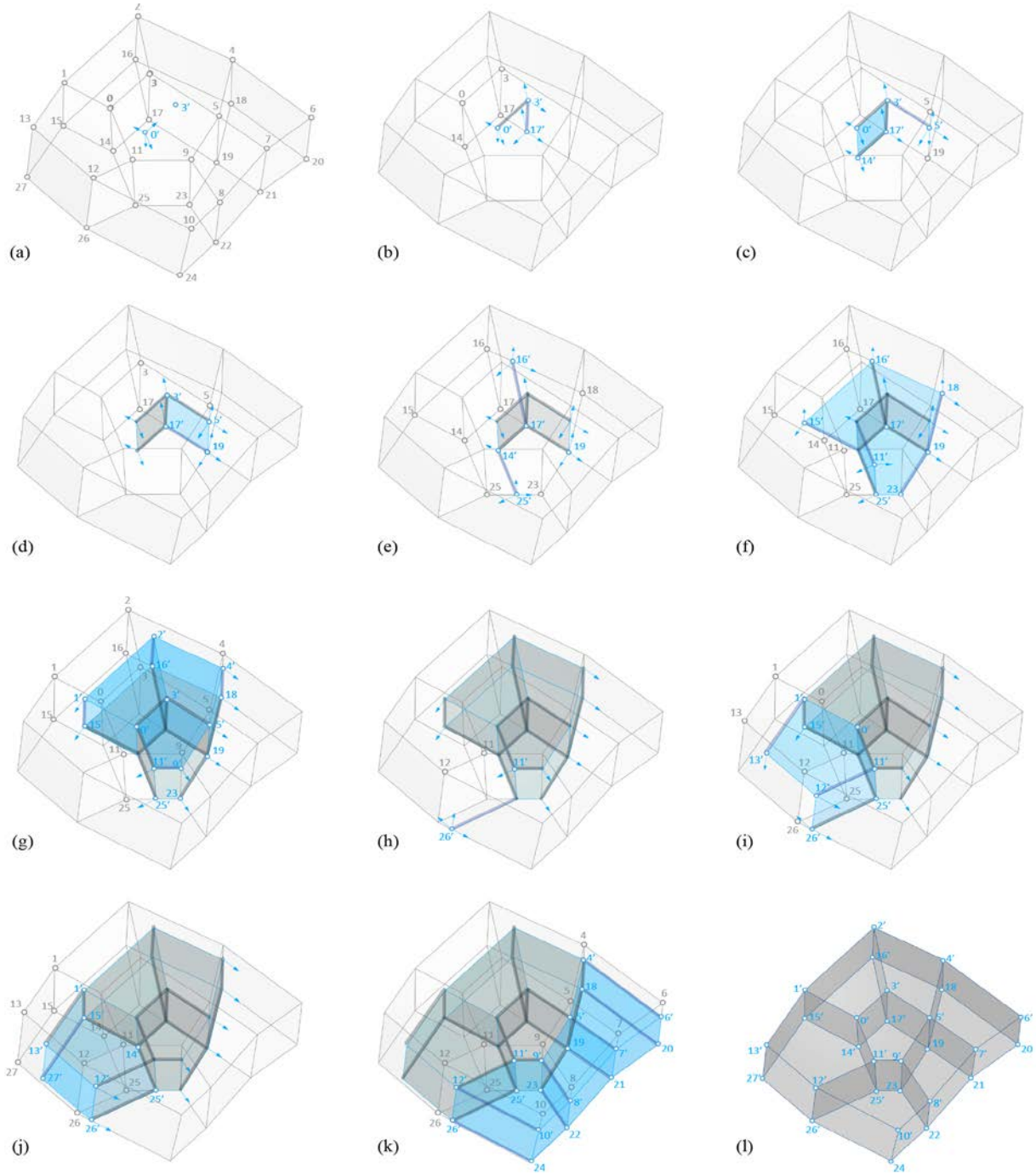


Figure 4. A typical polyhedral group transformation in steps.

Figures 4.a to l show the step by step process of transformation for a typical simple polyhedral cluster. The cluster has 7 cells, grouped in one single layer for reasons pertaining to process visibility. The gray line background plus the gray points and gray notations, refer to the original geometry of the polyhedral cluster. The blue overlay, including the blue dots and the blue notations with apostrophe, refers to the

transformed geometry. The blue fill tracks the geometry completed in the current step. The gray fill refers to the fully transformed faces completed in the previous steps. At each step the blue arrows show the position and direction of the available partial edges. The dark gray thick line overlay depicts the transformation path walked so far by the process and the dark blue line shows the current step.

The transformation starts from the translation of vertex v_0 into $v_{0'}$ (see Fig. 4.a). All other 27 vertex positions are calculated starting from here, using the procedures and rules described in the previous section. Since point at $p_{0'}$ is placed inside of a cell and not in any particular position on a face-plane or along an edge, the only available intersections for the first step are *out-of-face* intersections. The first *out-of-face* intersection is computed by intersecting *partial edge* $e_{(0',3')}$ with the face plane of $f_{(3,17,19,5)}$ to get the location for the translation of v_3 into $v_{3'}$. In the next step, pictured in Fig. 4.b, the transformation of v_3 in $v_{3'}$ propagates to all edges topologically connected to v_3. From all the potential intersections added, two can be classified as *in-face passive* intersections. Those are the ones corresponding to the topological connections to v_5 and v_{17}. The *passive in-face* intersections have precedence according to the rules of transformation prioritization, since all other available intersection are *out-of-face*. In Fig. 4.b partial edge $e_{(3',17')}$ has already intersected $e_{(17,19)}$ to produce transformed vertex $v_{17'}$. Evaluating all partial edges present in the system at this step, shows that now we have an active in plane intersection. The intersection occurs between two *partial edges* pointing towards a new position for original vertex v_{14}. Both edges are in the face-plane of *partial face* $f_{(0',3',17')}$. Since this is the only *active in-face* intersection in the waitlist at the moment, we can operate it and thus get the position for $v_{14'}$. The operation also completes $f_{(0',3',17',14')}$ highlighted in blue in Fig. 4.c and produces two other *partial edges* anchored in the same $v_{14'}$. Having exhausted all *in-face active* intersections, the transformation proceeds with another *passive* intersection visible in Fig. 4.c, to find a position for $v_{5'}$. In Fig. 4.d, two *partial edges* anchored in transformed $v_{17'}$ and $v_{5'}$ forming an active intersection, come together in the original position for v_{19}. This means v_{19} is parsed but not translated by the transformation process. In the same figure, newly completed face $f_{(3',17',19,5')}$ is highlighted.

Over the next figures, the film of the transformation is accelerated with multiple operations depicted in each image. The process switches between *passive in-face* intersections used to add new geometrical information to the mass of transformed parts inside polyhedral group and *active in-face* intersections to complete transformed faces and expand the transformed group of elements. For instance, Fig. 4.e and h depict new positions for transformed $v_{16'}$, $v_{25'}$ and $v_{26'}$ and the associated new partial edges. Figures 4.f, g, i and j, show the completion of multiple partial faces (depicted in blue) through *active in-plane* intersections. Intersections shown, also produce positions for moved vertices and new *partial edges* that sometimes result in a continuous active intersection streak as exemplified in Fig. 4.i and j. The final steps of the transformation are shown in Fig. 4.k. Here, the last vertices are computed and if necessary translated. Figure 4.l depicts the final form of the polyhedral system after all its vertices have been parsed.

2.5. Applications

From a 3D Graphic Statics point of view, the algorithm's utility has been explored mainly in the realm of the form. The technique's main merit is the ability to explore multiple geometries of a form diagram that remain reciprocal to a force polyhedral group. This exploration can empirically or algorithmically determine the degrees of freedom of certain geometries and thus inform the user on how to proceed with the design from an early stage. Any polyhedral geometry, provided it has some degrees of freedom, can lend itself to being transformed while remaining reciprocal to its dual counterpart. Using this, select elements of the geometry can be resized, supports or applied forces can be moved and the parts of the system can be translated or otherwise manipulated with regard to the whole. Figure 5 shows the local manipulation of a simple shell through the translation of vertices. Figure 5.a shows the result of three consecutive vertex translations that move both supports and applied force locations, completely transforming the look of the shell. In Fig. 5.b the first set of transformations are fully processed and a new transformation is initiated through the translation of a vertex from the upper part of the shell towards the bottom. This transformation reduces the thickness of the shell without altering the overall distribution of the supports and applied forces. The final geometry of the shell is shown in Fig. 5.c.

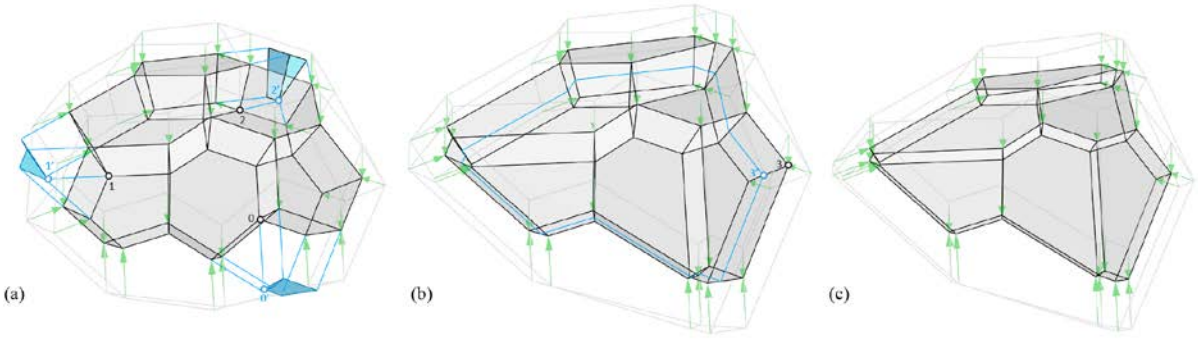


Figure 5. Local manipulation of a dual layer shell

Although the manipulation of form polyhedral diagrams has been the scope of this paper, the presented workflow and algorithm are equally as potent in transforming the force diagram in a 3DGS reciprocal relationship. For the force, the manipulation can be used to set values for certain exterior face areas, thus effectively transforming the boundary conditions of the whole system. For the interior faces, constrained manipulation of the faces can control the transformation of the load path. As a result, within the same loading conditions, similar support placement and identical topology, multiple ways of load distribution can be investigated.

Even though convexity of the faces and cells was presented as a constraint in the objectives of this research, the resulting tool is limited to convexity neither in the input nor in the results. Working with a force polyhedral system, the tool can be very useful in switching from pure compression or tension (convex) to compression-tension combined by transforming some convex faces into concave or complex ones. See example in Fig. 5. Vertex v_0 is moved beyond the convexity limits of the system resulting in a number of complex (self-intersecting) faces and cells and also a number of flipped edges and faces with a negative length or area. Figure 5.a shows the transformation of the geometry, up to the limit of convexity for all elements. In Fig. 5.b, the same point translation goes beyond the limit of convexity and some edges (shown in red) are flipped. Some faces also flip and their shape can be followed using the point notations. Figure 5.c shows the full final geometry of the polyhedral group.

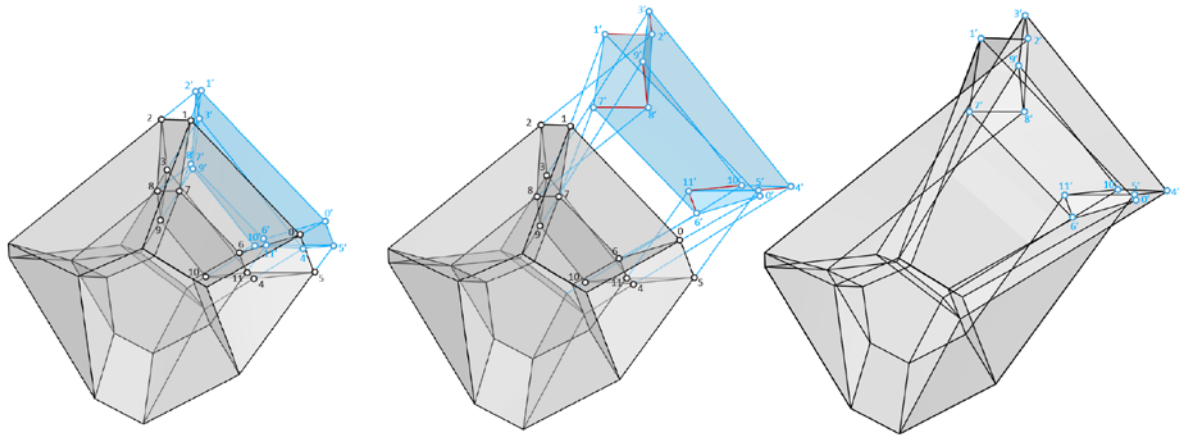


Figure 6. A transformation example that goes beyond the convexity limits

3. Conclusions

We have shown a fully geometric solution for the constrained manipulation of polyhedral groups inside a reciprocal relationship with their dual and demonstrated its ability to transform polyhedral groups while keeping the original normal direction of all faces unaltered.

Limitations

There are several limitations to the transformations possible using the presented algorithm. First, all manipulations are dependent on the degrees of freedom of the polyhedral group. This is in general inversely proportional to the percentage of triangular faces in the group. Some overly constrained systems (with many triangular faces) will only be uniformly scaled due to formal rigidity. Also depending on the degrees of freedom, local manipulations of the geometry can have global effects. This can result in unwanted change in other parts of the system. Second, an exact quantification of the transformation results is not yet possible. While the user can set specific values for the input manipulations, the results, due to the complex nature of the geometrical interactions are not always predictable. To address this, more research is needed. Third, even though any input transformation can be translated into points, a proper user interface helping with that is yet to be created. At the moment, complex manipulations, involving multiple synchronized offsets for faces or edges, are still problematic to test even though they are theoretically possible.

Future research

These are just a few of the envisioned uses for the algorithm presented in the previous pages. As stated before, one of the main benefits for the described tool is the exploration of manipulation possibilities for polyhedral clusters locked in a reciprocal relationship with their dual. Because of this, the present research should also be understood as a tool and a vehicle for the investigation of the possible manipulations in polyhedral groups. In order to fully understand the possibilities that this tool opens for 3DGS research in particular and polyhedral geometry in general further work is needed.

References

[1] M. Akbarzadeh, T. V. Mele, and P. Block, “On the equilibrium of funicular polyhedral frames and convex polyhedral force diagrams”, *Computer-Aided Design*, vol. 63, pp. 118–128, 2015.

[2] M. Akbarzadeh, T. V. Mele, and P. Block, “3D Graphic Statics: Geometric construction of global equilibrium”, *IASS 2015:Future Visions*, Amsterdam, 2015.

[3] M. Akbarzadeh, “Three Dimensional Graphical Statics using Polyhedral Reciprocal Diagrams,” dissertation, Zurich, 2016.

[4] E. D. Demaine and J. O'Rourke, Geometric folding algorithms: linkages, origami, polyhedra. Cambridge University Press, 2008.

[5] K. Ikeuchi, “Recognition of 3-D Objects Using the Extended Gaussian Image.”, *IJCAI*, pp. 595-600, 1981

[6] M. Konstantatou, A. McRobie, “Reciprocal construction using conic section and Poncelet duality”, *IASS 2016: Spatial Structures in the 21st Century*, Tokyo, 2016.

[7] J. Lee, T. V. Mele, and P. Block, “Area-controlled construction of global force polyhedra.”, *IASS 2017: Interfaces - Architecture. Engineering. Science*, Hamburg, 2017.

[8] J. Lee, T. V. Mele, and P. Block, “Form finding explorations through geometric transformations and modifications of force polyhedrons”, *IASS 2016: Spatial Structures in the 21st Century*, Tokyo, 2016.

[9] A. McRobie, "The geometry of structural equilibrium", *Royal Society Open Science*, vol. 4, no. 3, pp. 160759, Mar. 2017.

[10] S. Moni, “A closed-form solution for the reconstruction of a convex polyhedron from its extended Gaussian image”, *Proceedings. 10th International Conference on Pattern Recognition*, 1990

[11] W. J. M. Rankine, “XVII. Principle of the equilibrium of polyhedral frames,” *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 27, no. 180, pp. 92–92, 1864.