Computer-Aided Design 141 (2021) 103068

Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Algebraic 3D Graphic Statics: Constrained Areas

Masoud Akbarzadeh [a,c,*], Márton Hablicsek [a,b]

[a] Polyhedral Structures Laboratory, School of Design, University of Pennsylvania, Philadelphia, USA
[b] Department of Mathematics, Leiden University, Leiden, Netherlands
[c] General Robotic, Automation, Sensing and Perception (GRASP) Lab, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, USA

ARTICLE INFO

Article history:
Received 23 May 2020
Received in revised form 11 May 2021
Accepted 17 May 2021

Keywords:
Algebraic three-dimensional graphic statics
Polyhedral reciprocal diagrams
Geometric degrees of freedom
Static degrees of indeterminacies
Tension and compression combined polyhedra
Constraint manipulation of polyhedral diagrams

ABSTRACT

This research is a continuation of the Algebraic 3D Graphic Statics Methods that addressed the reciprocal constructions in an earlier publication (Hablicsek et al. 2019). It provides algorithms and (numerical) methods to geometrically control the magnitude of the internal and external forces in the reciprocal diagrams of 3D/Polyhedral Graphic statics. 3D graphic statics (3DGS) is a recently rediscovered method of structural form-finding based on a 150-year old proposition by Rankine and Maxwell in Philosophical Magazine. In 3DGS, the form of the structure and its equilibrium of forces are represented by two polyhedral diagrams that are geometrically and topologically related. The areas of the faces of the force diagram represent the magnitude of the internal and external forces in the members of the form diagram. The proposed method allows the user to control and constrain the areas and edge lengths of the faces of general polyhedrons that can be convex, self-intersecting, or concave in a group of aggregated polyhedral cells. In this method, a quadratic formulation is introduced to compute the area of a face based on its edge lengths only. This quadratic function is then turned into a linear formulation to facilitate the non-trivial computation of reciprocal polyhedral diagrams. The approach is applied to force diagrams, including a group of polyhedral cells, to manipulating the face geometry with a predefined area and the edge lengths. The method is implemented as a multi-step algorithm where each step includes computing the geometry of a single face with a target area and updating the polyhedral geometry. One of the remarkable results of this framework is to control the construction of the zero-area faces as proposed by McRobie (2017b). The zero-area faces represent a member with zero force in the form diagram. This research shows how self-intersecting faces, including the zero-area faces, can be constructed with additional edge constraints in a group of polyhedral cells without breaking the reciprocity of the form and force diagrams. Thus, it provides more hints on the generalization of the principle of the equilibrium of polyhedral frames. It also suggests a design approach where the boundary conditions and internal forces of compression-only systems can be manipulated to the design systems with both compression and tensile forces with no change in the geometry or the faces' planarity of the form diagram.

1. Introduction

Recently, geometry-based structural design methods, known as Graphic Statics, have been extended to 3D dimensions based on various approaches, among them those based on a historical proposal by Rankine and Maxwell in Philosophical magazine will be the subject of this article [1–11].

In this method which is called *3D Graphical Statics using Reciprocal Polyhedral Diagrams*, the equilibrium of the forces in a single node is represented by a closed *polyhedron* or a polyhedral *cell* with planar faces (Fig. 1). Each face of the force polyhedron is perpendicular to an edge in the form diagram, and the magnitude of the force in the corresponding edge is equal to the area of the face in the force polyhedron. The sum of all area-weighted normals of the cell must equal zero that can be proved using the divergence theorem [7,12,13]. In some cases, a cell can have *complex faces* (self-intersecting), which have multiple enclosed regions (Fig. 2b). The direction and the magnitude of the force corresponding to a complex face can be determined by summing the area-weighted normals of all of the enclosed regions (Fig. 2b, c). As a result, the direction of the internal force in the members of the structure might flip based on the direction of the face of a single force cell. In the context of the paper, we will use the '3DGS' acronym in place of 3D/polyhedral graphic statics and it

* Corresponding author at: Polyhedral Structures Laboratory, School of Design, University of Pennsylvania, Philadelphia, USA.
E-mail addresses: masouda@upenn.edu (M. Akbarzadeh), m.hablicsek@math.leidenuniv.nl (M. Hablicsek).

https://doi.org/10.1016/j.cad.2021.103068
0010-4485/

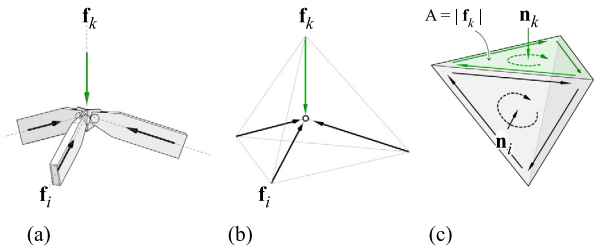

Fig. 1. (a) A 3D structural joint with an applied force and internal forces in its members; (b) the form diagram/bar-node representation of the same joint in the context of 3DGS; and (c) the force diagram/polyhedron representing the equilibrium of the same node in 3DGS.

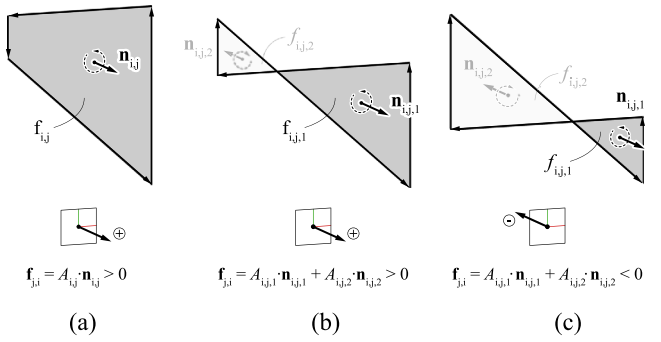

Fig. 2. From left to right: (a) A convex face with a positive force direction (out of the page); (b) a complex face with two enclosed regions and a positive net force direction; (c) and a complex face with two enclosed regions and a negative net force direction.

should not be mistaken by the vector-based 3D graphic statics [14,15].

Multi-layered funicular form-finding

Funicular structural forms carrying the applied loads in the form of pure tensile/compressive axial forces maximize the structural performance and minimize the use of materials. The internal structure of a bone is a classic example where material follows the principal stress directions and forms delicate lattice structures [16]. The 3D graphic statics discussed in this paper may be used to design the topology and the equilibrium geometry of funicular polyhedral systems. These funicular networks are quite similar to the *Thrust-Network Analysis (TNA)* proposed by Block and Ochsendorf [17]. The main difference is that TNA uses polygonal reciprocal diagrams [4,18], whereas 3DGS uses polyhedral reciprocal diagrams. Consequently, the thrust networks of TNA methods are two-manifolds (single layer), but the network generated by 3DGS can be two-manifolds or multi-layered polyhedral networks based on the topology of the force diagram.

Assigning the design loads

Prescribing the applied loads is an essential part of using 3D/polyhedral graphic statics. In this regard, three possible scenarios might be considered in the design process: the applied loads are the self-weight of the system; the applied loads are the combination of the self-weight/dead load and live load in the system, and the applied loads are significantly larger than the self-weight of the system.

The funicular polyhedral structures can be single-layer or multi-layered systems. In the case of a single-layer/two-manifold system, the self-weight can be calculated as the tributary area for each node multiplied by the thickness of the shell and compared with the area of the face corresponding to the applied load in the force diagram. Thrust Network Analysis (TNA) is an excellent example for this approach [17,19,20]. Each force diagram can describe the equilibrium of an infinite family of solutions of funicular forms due to the geometric degrees of freedom of the form diagram [17,21]. The tributary area can be different for each solution. Thus, matching the self-weight with the applied loads in the form-finding process should be achieved in multiple steps. In each step after form-finding, the tributary area will be calculated and compared with the area of the face of the applied load in the force diagram. The area of the face should be updated accordingly using iterative methods [2,5] or algebraic methods, which will be discussed in this paper. Once the difference between the self-weight and the area of the applied load face is minimized, the form-finding process for the self-weight of the structure may stop. The self-weight for multi-layered funiculars can be calculated based on segments of the structure under the 2D extrusion of the tributary area of an external load where the applied load is exerted. The best example for this scenario is the design of *Hedracrete* structure where the weight of the spatial members connected to an external node of a funicular polyhedral structure is used as the applied load for the system [22].

The second scenario where both live load and dead load are considered can be addressed: first, a funicular form resulting from the self-weight can be found. Subsequently, an additional funicular form resulting from the live load can be superimposed on the initial form to create a pushover funicular solution for both loading cases. The *Salginatobel* bridge designed by Robert Maillart is an excellent example where the funicular forms from both self-weight and live load were used to derive the final geometry of the bridge [23,24].

The third approach is often used in the design of frames or trusses where the self-weight is negligible compared to the externally applied loads. The design of the Eiffel tower could be used as a good example where the applied load other than gravity is used in the design process, which made the self-weight of the structure insignificant to be included in the form-finding process [25].

In addition, in geometry-based form-finding methods, it is assumed that the stiffness of the construction material is almost infinite. I.e., the construction material has a very large modulus of elasticity and a very small Poisson's ratio, and therefore, the deflection in the system is negligible. That assumption is, in fact, the main reason that such techniques were quite powerful in the design of masonry structures [26]. Moreover, the joints in the structural form are also rotation free with no moment resistance.

In reality, the modulus of elasticity of common construction materials such as steel or concrete is not infinite. Thus the self-weight might cause deformation in the system, particularly in the structural forms designed for applied loads much larger than the self-weight of the system. In such scenarios, if the self-weight is not considered in the form-finding process, the global deformation of the system under the self-weight should be measured using Finite Element Analysis to avoid excessive internal stress. Adding moment-resistance stiffness at the location of the joints for the frame structures could also help to resist an internal bending moment caused by deflection in the system.

Structural efficiency of the systems

In the funicular forms resulted from 3DGS, if the stiffness is not infinite then the efficiency of the structural system becomes a function of material property and load-bearing capacity. The structural efficiency of funicular polyhedral structures designed by 3DGS has recently been investigated in multiple

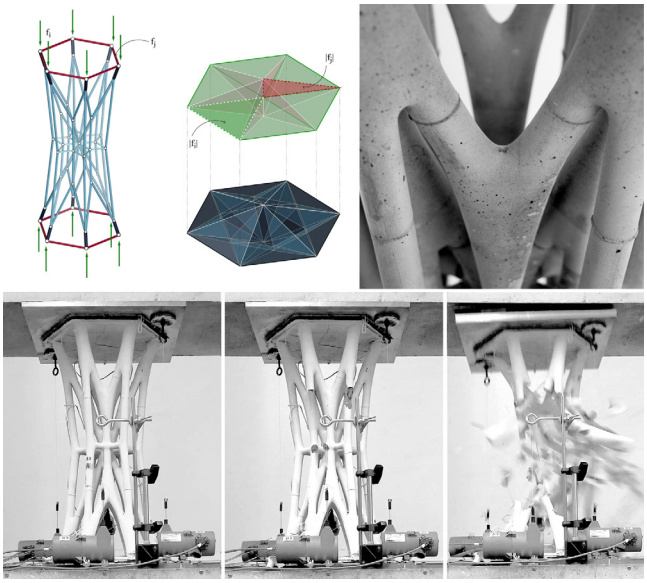

Fig. 3. Top: form and force diagram for a cylindrical sample design by using 3DGS; and bottom: the final compression-only prototype tested to failure [27].

studies [27,28]. For instance, a small-scale prototype with dimensions proportional to the standard concrete cylindrical compression test was constructed using in-situ high-performance, self-consolidating concrete. The 12 kg (26.45 *lbs*) specimen could take 240 kN load (2000 times more than its self-weight). The experimental results revealed the *resiliency* of the system as the local buckling did not immediately cause the global failure of the system due to the internal indeterminacy (states of self-stress) of the funicular geometry. The structural efficiency of the system was also verified as all remaining members of the specimen simultaneously collapsed under their maximum strength [27] (Fig. 3). Bolhassani et al. [22] includes further discussions on the efficiency of the system compared with a conventional concrete frame structure.

Fabrication rationalization

The polyhedral constraints of the method and the resulting funicular forms with planar faces might seem restricting in the design process. Yet, the polyhedral geometries of the resulting structural forms facilitate their construction using flat-sheet materials. For instance, 3DGS can be used for the design and construction of compression-only glass shells with planar faces [21]. Therefore, the methods of 3D/polyhedral graphic statics can combine form-finding and fabrication rationalization in one step.

A valuable teaching tool

The geometric relationships between the form and force diagrams in an interactive environment help students intuitively understand the internal force flow in structural systems and funicular forms. This property has been very well demonstrated using the methods of 2D graphic statics [29,30]. For instance, a designer can change the magnitude of the applied forces by geometrically adjusting the force diagram and observe the resulting change in the form of the structure and its internal forces. The algebraic implementation of the methods of graphic statics proposed by Van Mele and Block [31] was an important step in developing an educational platform for students to explore the structural forms and the geometry of their force equilibrium interactively. This geometric relationship can teach students what parameters control their design and how they can deliberately modify/optimize them to achieve specific design criteria. There is currently no educational platform for students to learn the principles of 3DGS presented in this paper. The graphic statics' procedural methods are insufficient to develop such a platform, and algebraic formation is needed for the development of interactive tools.

Another limitation of using 3DGS for educational purposes is the necessary preparation to understand the reciprocal relationship between the polyhedral cells and the geometry of the funicular forms. Some might find the polyhedral representation of forces as not intuitive and explicit as the 2D methods of graphic statics. This observation is accurate: the relationship between the faces of the force diagram and the edges of the form diagram might not be trivial to understand and reflect upon instantly for an untrained eye. This obstacle could be overcome in the future using more comprehensive visualization methods and educational packages. Particularly, including examples which relate 2D graphic statics methods to polyhedral graphic statics could be quite beneficial.

Like many other new concepts and methodologies, practicing procedural methods and reviewing a series of examples can overcome this obstacle. In this regard, a helpful approach is to use Minkowski sum in transforming the form and force diagrams into each other as suggested by McRobie [7]. Both form and force diagrams must be constructed first and added together using Minkowski sum to make an interactive transformation. An algebraic formulation can play an essential role in facilitating this process.

There are currently many tools and packages available for structural form-finding [32–34], and the paper intends not to provoke competition between the use of 3DGS and other tools and methods available for structural form-finding. Instead, authors rely on valuable characteristics of the methods of 3DGS. Understanding the geometric relationship between the funicular form and its force distribution suffices to find the spatial funicular forms. Thus, further research in the mathematical relationship between the form and force diagrams in 3D is inevitable and will provide resources for researchers who intend to learn and contribute to the computational implementation of the methods of 3DGS.

Application in material science

The cellular/polyhedral structures have numerous applications in materials with specific micro-architecture [35]. For instance, in bio-medical research, the porosity of such structures reduces the weight and increases the biodegradability of an implant. In a recent study, the application of 3D/polyhedral graphic statics has been investigated to design the cellular solids' architecture. It has been shown that subdividing the cells of the force diagram with respect to particular axes can translate buckling-prone cellular funicular structures of 3DGS to shellullar [36] funicular systems with a very low density of the material, which can be used to design highly efficient structural systems in large and small scales [37].

Exploiting the potential of 3DGS in design and engineering requires the ability to manipulate the geometric diagrams without breaking the reciprocity between the two and instantly observe the effect of the change in the other diagram. The existing computational tools for the design and manipulation of the reciprocal polyhedrons of 3DGS are quite limited. Moreover, controlling and optimizing the magnitude of forces by changing the areas of the faces needs efficient algorithms.

1.1. Related works

In 2016, Akbarzadeh [2] showed that the reciprocal diagrams of 3DGS can be constructed in a procedural (step-by-step) approach in a parametric software by assigning constraints between reciprocal components of each diagram that allows simultaneous control over the geometry of both diagrams. This method is extremely time-consuming and tedious for structures with a large number of nodes and members. Akbarzadeh et al. [13] developed a computational algorithm that could receive convex polyhedral cells as a primal, and construct its reciprocal diagram iteratively within a certain tolerance defined by the user (see also [38]). The main limitation of this method is that it cannot deal with (non-convex) self-intersecting polyhedrons or explore tension and compression equilibrium. Moreover, controlling the areas of the faces was computationally quite expensive. In 2018, Lee et al. [6] proposed a method called *Disjointed Force Polyhedra* where the equilibrium of the system was computed by constructing a single convex polyhedron for each node using Extended Gaussian Image algorithm [39–41] and matching the areas of the shared faces [6,42]. This method allows the control of the areas of the convex cells, but it breaks the reciprocity between the two diagrams. Moreover, it cannot control the areas of the self-intersecting faces. Recently, Hablicsek et al. [43] developed an algebraic formulation relating the geometry of the reciprocal polyhedral diagrams using a linear system of equations. This method can directly construct the dual from a given primal in one step. Although the previous formulation could immediately construct the reciprocal polyhedral diagrams, it did not provide any insight into how to control the areas of the faces corresponding to the magnitude of the forces in the form diagram. Moreover, geometrically constrained constructions were also not addressed.

1.2. On the importance of algebraic formulation

The algebraic formulation of the graphic statics methods can reveal the true potentials of the method in design. For instance, in a previous study, the authors showed that the algebraic relationship between the form and force allows exploring multiple equilibrium configurations for a single force distribution, which was not trivial to compute using other approaches. For instance, for a single convex force distribution of Fig. 4, a compression-only synclastic funicular form, as well as two other anticlastic configurations with combined tension and compression members, can be found [44]. Such properties can be explored by utilizing the form diagram's geometric degrees of freedom computed using the algebraic approach. The previously explored iterative methods could not lead to such explorations. Thus the algebraic formulation provides insight into the possibilities of using the methods of 3DGS in design and construction, which could remain unexplored otherwise.

On the limitations of 3D/polyhedral graphic statics

The 3D/polyhedral graphic statics methods discussed in this paper are based on the reciprocal polyhedral diagrams proposed by Rankine [1] and Maxwell [4]. That is, the reciprocity between the form and force diagram is induced by polyhedral systems. Consequently, the resulting equilibrium states are limited to polyhedral systems or configurations that can be translated into polyhedral systems by the inclusion of zero-bar elements or tetrahedralization [45]. Nevertheless, the 3D/polyhedral graphic statics' planarity constraints significantly facilitate the construction of structures designed using this method. The application of these methods in other sciences and their unique educational

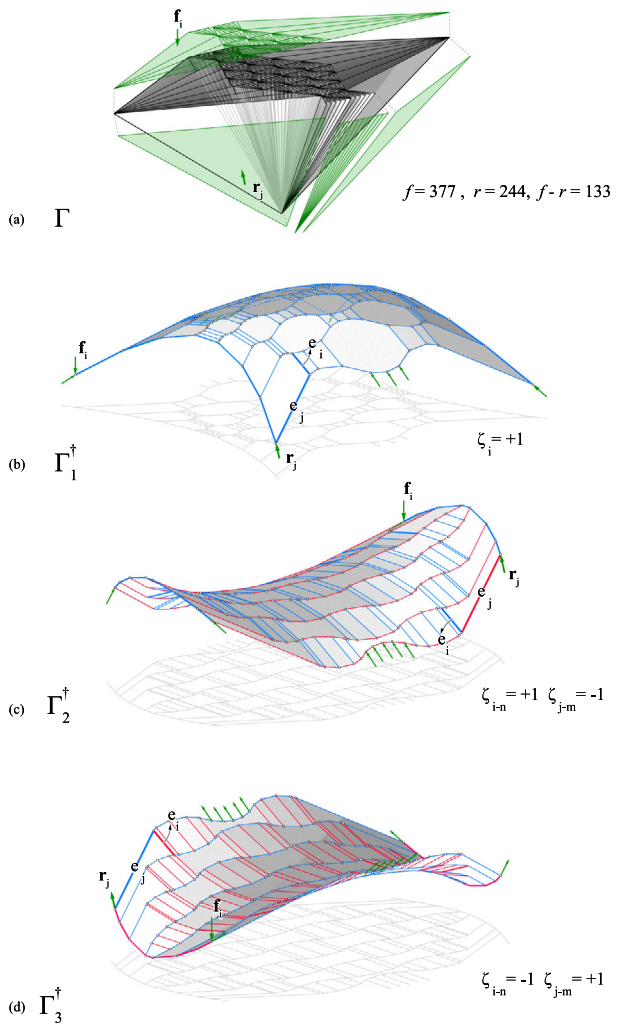

Fig. 4. (a) A force diagram consisting of convex polyhedral cells with 377 number of faces resulting in (b) a (synclastic) compression-only form with 133 degrees of freedom; (c) and (d) two different (anticlastic) shells with both tension and compression members by assigning both negative and positive values to the edges of the form (b) [44].

advantages justifies further research in their relevant theoretical approaches. In addition, the relatively challenging readings of polyhedral diagrams and their relationship to the funicular structural form should also be considered as a limitation that needs to be addressed in broadening the use of this technique among the practitioners and designers.

Another limitation of the methods of 2D and 3D graphic statics, in general, is that the application of external loads on the internal nodes of the structure is not allowed. This limitation can be explained by visiting the algebraic methods for 2D graphic statics proposed by Van Mele and Block [31]. In the mentioned approach, it is required that the primal graph be *planar* meaning that it can be drawn on a 2D plane without crossing edges. Connecting an external load or a free edge to the internal vertices of a form diagram results in a non-planar graph and overlapping spaces. Thus the reciprocal diagram cannot be found using the proposed method. The same limitation applies in polyhedral reciprocal diagrams. Adding a free-edge as an applied load to the form will result in overlapping the topological /decomposed spaces, and thus the reciprocal diagram cannot be found using the

proposed methodology of this paper. In contrast, other numerical form-finding methods such as force density and dynamic relaxation methods may overcome that limitation in the form-finding process (see, for instance, [33,34]).

1.3. Contributions

This paper provides a robust algebraic method to construct polyhedrons with assigned areas and edge lengths of their faces, from which its reciprocal dual can be constructed as the structural form. The formulation introduced in this paper relates the areas of the faces of the polyhedral system to its edge lengths allowing the combination of this method with the previous algebraic formulation to control the areas of the faces. Moreover, the methods of this research can compute the areas of self-intersecting faces with constraint edges, which has never been addressed in the literature previously. Specifically, this approach can construct zero-area, self-intersecting faces in the system, where the sum of the signed areas of a self-intersecting face is zero. The existence of such faces in the force diagram allows the removal of the forces in the boundary or internally and therefore, describes internal force equilibrium that previously was not possible using reciprocal polyhedral diagrams.

The paper is organized as follows. A quadratic formulation to compute the area is introduced for a single face based on its edge lengths (Section 2.3). Then, a methodology is described to manipulate the geometry of the face with a predefined area and edge lengths (Section 2.4). Subsequently, the geometry of the polyhedron is updated with the newly changed faces (Section 2.5). This approach is a multi-step algorithm, where each step includes the computation of the geometry of a single face and an update of the polyhedral geometry. In the end, the dual structural form is updated with the new magnitude of the internal or external forces (Section 2.6). Alongside the theory, we provide the computational setup in Section 3 describing the main algorithms in detail. Finally, Section 4 shows the application of this method in the design of funicular structures with zero force members or reactions in the boundary conditions.

1.4. Nomenclature

We denote the algebra objects of this paper as follows; matrices are denoted by bold capital letters (e.g. $\mathbf{A}$); vectors are denoted by lowercase, bold letters (e.g., $\mathbf{v}$), except the user input vectors which are represented by the Greek letters (ν and ξ). Table 1 encompasses all the notations used in the paper.

Table 1
Nomenclature for the symbols used in this paper and their corresponding descriptions.

Topology	Description
Γ	Primal diagram
$\Gamma^\dagger$	Dual, reciprocal diagram
v	# of vertices of Γ
e	# of edges of Γ
f	# of faces of Γ
c	# of cells of Γ
$v^\dagger$	# of vertices of $\Gamma^\dagger$
$e^\dagger$	# of edges of $\Gamma^\dagger$
$f^\dagger$	# of faces of $\Gamma^\dagger$
Matrices	
$\mathbf{M}_f$	Area matrix of the face f
$\mathbf{E}_f$	Equilibrium matrix of the face f
$\mathbf{L}_f$	Matrix of predefined edge lengths of the face f
$\mathbf{E}_p$	Equilibrium matrix of the polyhedral system p
$\mathbf{E}^\dagger$	Equilibrium matrix of the dual
$\mathbf{L}_p$	Matrix of predefined edge lengths of the polyhedron p
$\mathbf{B}_f$	Constraint matrix of the face f
$\mathbf{B}_p$	Constraint matrix of the polyhedron p
$\mathbf{B}_p^+$	Moore–Penrose inverse of $\mathbf{B}_p$
$\mathbf{RREF}_f$	RREF of $(\mathbf{B}_f \vert \mathbf{b}_f)$
$\mathbf{E}^\dagger$	Equilibrium matrix of the dual diagram
$(\mathbf{E}^\dagger)^+$	Moore–Penrose inverse of $\mathbf{E}^\dagger$
Vectors	
$\mathbf{n}$	Consistent unit normal vector
$\mathbf{q}$	Vector of edge lengths
$\mathbf{u}_j$	Direction vector of edge vector $\mathbf{e}_j$
$\mathbf{b}_f$	Constraint vector for the face f
$\mathbf{l}_f$	Vector of predefined edge lengths of the face f
$\mathbf{b}_p$	Constraint vector for the polyhedron p
$\mathbf{l}_p$	Vector of predefined edge lengths of the polyhedron p
$\mathbf{q}_{nci}$	Vector of nci edge lengths of a face f_i
$\mathbf{q}_{fix}$	Vector of fixed edge lengths of a face f_i
$\mathbf{q}_{nfd}$	Vector of nfd edge lengths of a face f_i
$\mathbf{q}_{ci}$	Vector corresponding to the edge length of the ci edge
$\mathbf{e}_j^\dagger$	Edge vector of $e_j^\dagger$ in $\Gamma^\dagger$
$\mathbf{u}_j^\dagger$	Direction vector of edge vector $\mathbf{e}_j^\dagger$
$\mathbf{q}^\dagger$	Vector of edge lengths of $\Gamma^\dagger$
Parameters	
ν	Parameter for the MPI method to solve Eq. (30)
ξ	Parameter for the MPI method to solve Eq. (33)
Other	
q_i	Signed length of the edge $\mathbf{e}_i$
$\vert\mathbf{e}_i\vert$	Length of the edge $\mathbf{e}_i$
O	Centroid of a face
$h_{i,j}$	(signed) distance of v_j from e_i
H_i	Average (signed) distance of the v_j from e_i
A_f	Area of face f
$\mu_{i,j}$	The scalar ratio between $\mathbf{e}_i \times \mathbf{e}_j$ and $\mathbf{n}$
$\eta_{i,j}$	The scalar ratio between $\mathbf{u}_i \times \mathbf{u}_j$ and $\mathbf{n}$
$GDoF_f$	Geometric Degrees of Freedom of face f
$CGDoF_f$	Constrained Geometric Degrees of Freedom of face f
e_{fix}	(user)-selected fixed edges of a face
e_{ind}	List of independent edges of a face
e_{nfd}	List of *nfd* edges of a face
e_{nci}	List of *nci* edges of a face
e_{fix}^p	(user-)selected fixed edges of the polyhedron
e_{ci}	Critical independent edge
q_{ci}	Signed length of the e_{ci} edge
r	Rank of $\mathbf{RREF}_f$

2. Research methods

2.1. Overview

The first step in the methodology is to link the areas of the faces of the polyhedral system to their edge lengths. The definition of the area based on the edge lengths will result in a quadratic function per face of the polyhedron. As a result, controlling the areas of the faces of the polyhedral system requires to solve a complex system of non-homogeneous quadratic equations simultaneously. This complex system of quadratic equations is usually solved using quadratically constrained quadratic programming. However, to the knowledge of the authors, these methods usually fail in computing self-intersecting/compression – and – tension combined systems. Moreover, the objective of our research is to provide a methodology to control the areas of the faces without perturbing the system drastically. Therefore, in this section, we provide a simple methodology to solve these quadratic equations sequentially by preserving the main geometric features of both the form and force diagrams.

In this method, there are two types of equation; (i) the quadratic equations that compute the areas of the faces based on the edge lengths; and (ii) the linear equations that provide the geometry of the faces of the polyhedrons with user-defined edge lengths as constraints. The quadratic equations of the faces are solved using the linear equations around the edges of each face with constrained edge lengths. Each quadratic equation for a face area has as many variables as the number of edges of the face which results in a variety of significantly different solutions (see Fig. 10a–d for illustration). However, we can control the solution space by reducing the number of variables to one. This allows us to find a solution for the quadratic equation with a limited geometric perturbation in the system.

In the following sections, we introduce the steps to develop the quadratic equation to compute the area of a face of a polyhedral system based on the edge lengths, and then we develop the non-homogeneous linear equation system describing the equilibrium equation for the face with predefined edge lengths. In the end, we show how to solve the equation system and how to recompute the geometry of the form.

2.2. Linear equilibrium equations for a polyhedral system

In a previous paper, Hablicsek et al. [43] showed how to write the equilibrium equations for a system of polyhedral cells with planar faces. For each face f_i, we can write an equation based on its edge lengths that shows the closeness of the face. The term *edge length* is used in two occasions in this paper: (a) a scalar which represents the *signed* length of an edge vector $\mathbf{e}_i$ and is shown by q_i both in the formulation and the text; and (b) the actual edge length of the vector $\mathbf{e}_i$ which is always positive and denoted by $|\mathbf{e}_i|$. By choosing a normal vector for each face f_i, we can obtain a consistent edge orientation. We denote the unit direction vector $\mathbf{u}_j$ corresponding to the edge vector $\mathbf{e}_j$ of Fig. 5a. Since each face provides a closed loop of edges, the sum of the edge vectors has to be the zero vector. Thus, we obtain a vector equation for the edge lengths q_j of $\mathbf{e}_j$ as

$$\sum_{e_j} \mathbf{u}_j q_j = \mathbf{0} \tag{1}$$

where the sum runs over the edges e_j of the face f_i. We remark that the edge lengths q_j can take both *negative and positive values*; changing the length of an edge to its negative means that we change the direction of the edge to the opposite direction without changing the direction of the unit vector.

Thus, each face f_i of the polyhedron p provides three equations for the edge lengths, one equation for each of the x-coordinates, y-coordinates, and z-coordinates. Thus, Eq. (1) can be described by a $[3 \times e]$ matrix, $\mathbf{E}_{f_i}$

$$\mathbf{E}_{f_i}\mathbf{q} = \mathbf{0} \tag{2}$$

where $\mathbf{q}$ denotes the vector of the edge lengths of the polyhedron. This equation describes the geometry of the face.

Similarly, we can obtain a $[3f \times e]$ matrix, $\mathbf{E}_p$ describing the geometry of the entire network. Here f denotes the number of faces and e denotes the number of edges in the network. In other words, we have a linear equation system

$$\mathbf{E}_p\mathbf{q} = \mathbf{0} \tag{3}$$

where $\mathbf{q}$ denotes (again) the vector of edge lengths of the polyhedron p. Each solution of the linear equation system (3) represents a network, whose edges are parallel to their associated edges of the original network with different edge lengths.

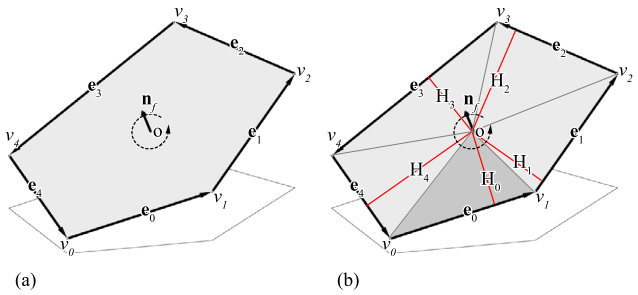

Fig. 5. (a) Vertices and edge vectors of the face f with a normal direction $\mathbf{n}_f$; and (b) dividing the face into triangles with a base $\mathbf{e}_i$ and the height H_i from the centroid of the vertices.

2.3. Quadratic equation system for the area of a face

In this section, we explain, how to develop a quadratic system of equations for a face f_i of a polyhedral network based on the edge vectors of the face after O'Rourke [46].

Consider a face f_i with k vertices: $v_0, v_1, \ldots, v_{k-1}$. We denote the edge e_i by the vertices v_i and v_{i+1}. Let $\mathbf{n}$ be a chosen unit normal vector of the face f_i. Using the right-hand rule, the normal $\mathbf{n}$ provides the direction of the edges. We denote the directional edges by $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{k-1}$ vectors. For the sake of simplicity, in the following explanation, we use a cyclic order of the edges, meaning e_k and e_{k+1} will also denote the edges e_0 and e_1, etc.

We can compute the area of the face f_i by:

- dividing the face f_i into k triangles given by the e_i and the geometric center (centroid) of vertices, O; and
- computing the area of each triangle by computing its height H_i which is the distance from O to the edge e_i (Fig. 5b).

Average height

The height H_i, the perpendicular distance of O from the edge e_i, is the average of the signed projected distances $h_{i,j}$ of the vertices $v_0, v_1, \ldots, v_{k-1}$ from e_i (Fig. 6d).

For instance, $h_{0,2}$, $h_{0,3}$, and $h_{0,4}$ are the signed distances of the vertices v_2, v_3 and v_4 from the edge e_0. The $h_{0,0}$ and $h_{0,1}$ are zero since the vertices v_0 and v_1 lay on edge e_0. Therefore, the area A_i of the triangle O, v_i and v_{i+1} can be written as

$$A_i = \frac{1}{2}|\mathbf{e}_i|H_i = \frac{1}{2k}|\mathbf{e}_i|\sum_{j=0}^{k-1} h_{i,j},$$

and the total area of the face equals

$$A_f = \sum_{i=0}^{k-1} A_i = \frac{1}{2k}\sum_{0\le i,j\le k-1} |\mathbf{e}_i|h_{i,j}.$$

We are looking for a formula for the area A_i based on the edge vectors. Thus, let us compute the $h_{i,j}$ based on the edge lengths of the face f_i. Recall, that the vertices v_i and v_{i+1} are on the edge e_i, hence, $h_{i,i}$ and $h_{i,i+1}$ are zero (see Fig. 6a, d). The first vertex that can contribute non-trivially is v_{i+2}, and the height, $h_{i,i+2}$, can be computed by constructing the triangle of the vertices v_i, v_{i+1} and v_{i+2} (Fig. 6a). We denote the signed area of this triangle by $A_{i,i+2}$ that can be computed using the two following methods:

- the area can be found by the height $h_{i,i+2}$ (Fig. 6a):

$$A_{i,i+2} = \frac{1}{2}|\mathbf{e}_i|h_{i,i+2}. \tag{4}$$

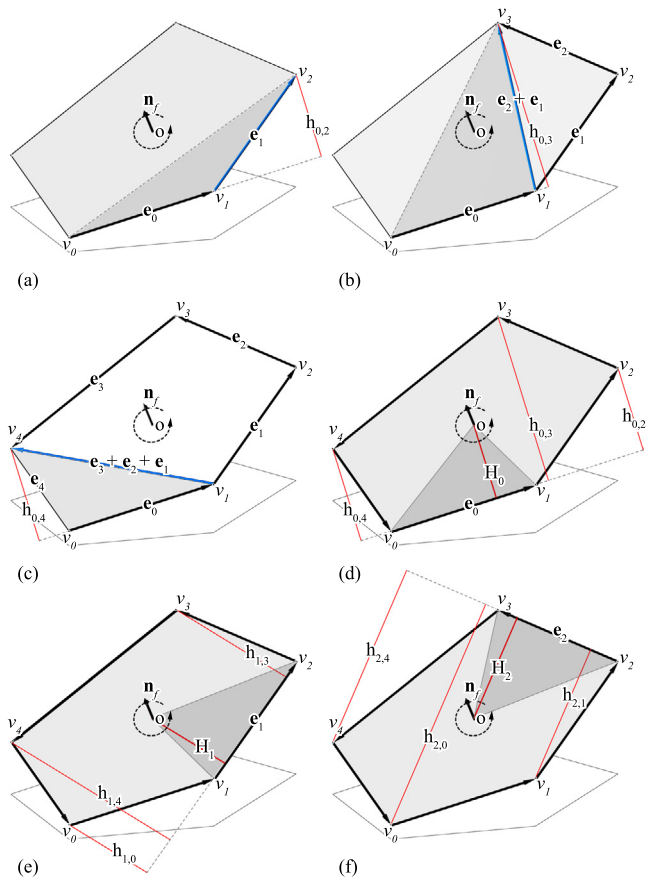

Fig. 6. Finding the average height H_i for each edge $\mathbf{e}_i$ by constructing triangles starting from (a) $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$, and (b) $\mathbf{e}_i$ and $\mathbf{e}_{i+1}+\mathbf{e}_{i+2}$ until we find all heights of the vertices in (c) and (d); and repeating the same process for other edges to find all H_is in (e) and (f).

- also, the cross product of $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ provides the signed area:

$$A_{i,i+2}\mathbf{n} = \frac{1}{2}(\mathbf{e}_i \times \mathbf{e}_{i+1}) \tag{5}$$

Note that the sign of the area in Eq. (4) is defined by the $h_{i,i+2}$ where it can only be negative in a concave or a self-intersecting polygon.

From Eqs. (4) and (5), we get the following equation of vectors

$$\frac{1}{2}(\mathbf{e}_i \times \mathbf{e}_{i+1}) = \frac{1}{2}|\mathbf{e}_i|h_{i,i+2}\mathbf{n}. \tag{6}$$

We would like to solve this equation for the scalar $h_{i,i+2}$. However, we have two vectors on the sides of the equation above. On one hand, it is not allowed to divide vectors by vectors if their directions are neither exactly the same nor opposite. On the other hand, the vectors $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ are perpendicular to $\mathbf{n}$. Hence, the cross product of vectors $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ is either parallel or opposite to $\mathbf{n}$. Therefore, there exists a scalar $\mu_{i,i+1}$ so that

$$\mu_{i,i+1}\mathbf{n} = \mathbf{e}_i \times \mathbf{e}_{i+1}.$$

Thus, we can think of $\mu_{i,i+1}$ as

$$\mu_{i,i+1} = \frac{\mathbf{e}_i \times \mathbf{e}_{i+1}}{\mathbf{n}}. \tag{7}$$

Going back to Eq. (6), using the notations introduced above, we obtain a formula for $h_{i,i+2}$:

$$h_{i,i+2} = \frac{\mathbf{e}_i \times \mathbf{e}_{i+1}}{|\mathbf{e}_i|\mathbf{n}} = \mu_{i,i+1}\frac{1}{|\mathbf{e}_i|}. \tag{8}$$

Eq. (8) is not of the form of a linear equation, because the scalar $\mu_{i,i+1}$ depends on the lengths of the edges $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$. By dividing both sides of Eq. (7) by the edge lengths of e_i and e_{i+1}, the scalar $\mu_{i,i+1}$ will become

$$\eta_{i,i+1} = \frac{1}{|\mathbf{e}_i| \cdot |\mathbf{e}_{i+1}|} \cdot \mu_{i,i+1}. \tag{9}$$

The scalar $\eta_{i,i+1}$ will only depend on the directions of $\mathbf{e}_i$ and $\mathbf{e}_{i+1}$ and not on the edge lengths. Using this notation, we can write $h_{i,i+2}$ as

$$h_{i,i+2} = \eta_{i,i+1}|\mathbf{e}_{i+1}|. \tag{10}$$

In Eq. (10), the height is expressed as a product of a scalar $\eta_{i,i+1}$ that depends only on the direction of the edges, and the edge length of $\mathbf{e}_{i+1}$. Therefore, we obtained the height $h_{i,i+2}$ based on the linear function of the edge length.

In the next step, we will compute $h_{i,i+3}$. Consider the triangle with vertices v_i, v_{i+1}, v_{i+3} (see Fig. 6b). Note that the vector from v_{i+1} to v_{i+3} is the vector

$$\mathbf{e}_{i+1} + \mathbf{e}_{i+2}.$$

We compute the area $A_{i,i+3}$ of the triangle using two methods. The area can be computed by using the height $h_{i,i+3}$ (Fig. 6b):

$$A_{i,i+3} = \frac{1}{2}|\mathbf{e}_i|h_{i,i+3}.$$

Also, the cross product of $\mathbf{e}_i$ and $\mathbf{e}_{i+1}+\mathbf{e}_{i+2}$ provides the signed area:

$$A_{i,i+3}\mathbf{n} = \frac{1}{2}(\mathbf{e}_i \times (\mathbf{e}_{i+1} + \mathbf{e}_{i+2})).$$

As a consequence, we obtain the following relationship.

$$\frac{1}{2}(\mathbf{e}_i \times (\mathbf{e}_{i+1} + \mathbf{e}_{i+2})) = \frac{1}{2}|\mathbf{e}_i|h_{i,i+3}\mathbf{n}$$

We combine this equation with Eq. (6) to obtain

$$\mathbf{e}_i \times \mathbf{e}_{i+2} + |\mathbf{e}_i|h_{i,i+2}\mathbf{n} = |\mathbf{e}_i|h_{i,i+3}\mathbf{n} \tag{11}$$

Again, we would like to divide Eq. (11) by the vector $\mathbf{n}$ to solve the equation for the scalar $h_{i,i+3}$. As before, the vectors $\mathbf{e}_i$ and $\mathbf{e}_{i+2}$ are perpendicular to $\mathbf{n}$, hence $\mathbf{e}_i \times \mathbf{e}_{i+2}$ is either parallel or opposite to the direction of $\mathbf{n}$. We, again, introduce the scalar $\mu_{i,i+2}$ satisfying

$$\mu_{i,i+2}\mathbf{n} = \mathbf{e}_i \times \mathbf{e}_{i+2}.$$

With this notation, Eq. (11) becomes

$$\mu_{i,i+2} + |\mathbf{e}_i|h_{i,i+2} = |\mathbf{e}_i|h_{i,i+3},$$

so we obtain a recursive formula

$$h_{i,i+3} = h_{i,i+2} + \frac{1}{|\mathbf{e}_i|}\mu_{i,i+2}. \tag{12}$$

The scalar $\mu_{i,i+2}$ depends on the lengths of the edge vectors $\mathbf{e_i}$ and $\mathbf{e_{i+2}}$, however $\eta_{i,i+2}$ defined below does not:

$$\eta_{i,i+2} = \frac{1}{|\mathbf{e}_i| \cdot |\mathbf{e}_{i+2}|}\mu_{i,i+2}.$$

The scalar $\eta_{i,i+2}$ only depends on the directions of the edges. Using this notation and Eqs. (10), (12) becomes a linear expression for $h_{i,i+3}$

$$h_{i,i+3} = h_{i,i+2} + \eta_{i,i+2}|\mathbf{e}_{i+2}| = \eta_{i,i+1}|\mathbf{e}_{i+1}| + \eta_{i,i+2}|\mathbf{e}_{i+2}| \tag{13}$$

Repeating this process for the rest of the edges (Fig. 6c) results in a formula for $h_{i,i+l}$

$$h_{i,i+l} = \sum_{1\le m\le l-1} \eta_{i,i+m}|\mathbf{e}_{i+m}|.$$

Finally, we compute the average of these heights, H_i by repeating the same process for all the edges of the face (Fig. 6e,f):

$$H_i = \frac{1}{k}\sum_{j=2}^{k-1} h_{i,i+j} = \frac{1}{k}\sum_{j=2}^{k-1}\sum_{1\le m\le j-1} \eta_{i,i+m}|\mathbf{e}_{i+m}| \tag{14}$$

$$= \frac{1}{k}\sum_{j=1}^{k-2}(k-j-1)\eta_{i,i+j}|\mathbf{e}_{i+j}| \tag{15}$$

As a consequence, we obtain a quadratic formula for the area of the face f in the edge lengths of the edges of f

$$A_f = \frac{1}{2}\sum_{i=0}^{k-1}|\mathbf{e}_i|H_i = \frac{1}{2k}\sum_{i=0}^{k-1}\sum_{j=0}^{k-2}(k-j-1)\eta_{i,i+j}|\mathbf{e}_i|\,|\mathbf{e}_{i+j}|. \tag{16}$$

Quadratic form

The next step is to turn the quadratic equation (16) into a quadratic form with a matrix. Note, that we can compute the coefficients

$$(k-j-1)\eta_{i,i+j}$$

in Eq. (16) without knowing the edge lengths. As a result, we can formulate the right-hand side of Eq. (16) in a quadratic form given by a matrix $\mathbf{M}_f$, whose entries are given by the coefficients:

$$\mathbf{M}_{f,ij} = \begin{cases} (k-j+i-1)\eta_{i,j} & \text{if } j > i \\ (i-j-1)\eta_{i,j} & \text{if } j < i \\ 0 & \text{if } i = j. \end{cases} \tag{17}$$

Thus, we can rewrite Eq. (16) in a quadratic form

$$A_f = \frac{1}{2k}\mathbf{q}^T\mathbf{M}_f\mathbf{q} \tag{18}$$

where $\mathbf{q}$ is the column vector of the edge lengths

$$\mathbf{q} = (|\mathbf{e}_1|, |\mathbf{e}_2|, \ldots, |\mathbf{e}_k|)^T.$$

Symmetric matrix

Usually, the matrix $\mathbf{M}_f$ is not a symmetric matrix. However, the computations may become simpler if the matrix is symmetric. Indeed, the matrix $\mathbf{M}_f$ can be turned into a symmetric matrix. Since

$$\left(\mathbf{q}^T\mathbf{M}_f\mathbf{q}\right)^T = \mathbf{q}^T\mathbf{M}_f^T\mathbf{q}$$

the quadratic form given as

$$\frac{1}{2k}\mathbf{q}^T\mathbf{M}_f^T\mathbf{q}$$

also computes the area of the face.

As a consequence,

$$A_f = \frac{1}{2k}\mathbf{q}^T\frac{\mathbf{M}_f + \mathbf{M}_f^T}{2}\mathbf{q},$$

and in this case, the corresponding matrix

$$\frac{\mathbf{M}_f + \mathbf{M}_f^T}{2} \tag{19}$$

is symmetric.

For the sake of computational simplicity, from now on, we assume that the matrix $\mathbf{M}_f$ appearing in Eq. (16) is always symmetric.

Note that in the formulation, if the edge vectors need to be computed, the notation q_i is used which can take positive or negative values. This means that the reconstructed face can have negative edge length and its corresponding edge vector $\mathbf{e}_i$ may flip to the opposite direction (compared to its initial orientation). In all formulations, the notation $|\mathbf{e}_i|$ represents the actual length of the vector that is always positive.

2.4. Computing the face geometry for a target area

In this section, we develop a method to reconstruct a given face f_i by constraining particular user-defined edge lengths and the target area for the face. To give a general overview of our approach, consider the face f_i of Fig. 5a. Initially, without any constraint, we have five unknowns which are the edge lengths of the five edges e_{0-4}. There are three equilibrium equations based on Eq. (2), in $-x$, $-y$, $-z$ around the face f_i, and one of them is redundant [43]. As a consequence, the dimension of the possible solutions, i.e. the possible faces, satisfying the equilibrium equations is

$$e - 2 = 3.$$

Instead of solving the quadratic area equation (18) for three unknowns, we constrain two of them and solve the quadratic equation for only one unknown. Using this technique, we can significantly reduce the complexity of finding a solution for this quadratic equation. This provides additional design possibilities for the user, as we either allow the user to define up to two edge lengths out of three or we use the existing edge lengths for two edges and compute the area based on the last unknown edge.

In general, our goal is to simplify solving the quadratic equation of the face by solving it for only one edge length.

Computing $GDoF_f$ using RREF

The dimension of the possible solutions for the geometry of the face is called the Geometric Degrees of Freedom ($GDoF_f$). In fact, $GDoF_f$ describes the dimension of the family of polygons with edges parallel to the edges of face f which is always equal to:

$$e - 2.$$

The GDoF is also equal to the number of independent edges in each face. In fact, the lengths of the independent edges can define the lengths of the rest of the edges and the geometry of the face [44]. The independent edges can be found using the Reduced Row Echelon form method (RREF).

Specifying the edge lengths for the $e - 2$ independent edges yields a unique solution for the geometry of the face. In other words, a unique solution to Eq. (2) is obtained with preassigned edge lengths for the $e - 2$ independent edges. Although, the equilibrium equations (Eq. (2)) are satisfied, there is no guarantee that the area equation, Eq. (18), is satisfied as well.

However, we can specify $e-3$ independent edges. In that case, we will have infinitely many solutions given by the edge length of the last independent edge that, in fact, provides the possibility to solve the area equation.

This method provides a solution to recompute the geometry of the face with a given target area. However, the objective of the research is to construct the geometry of a face with a given target area and user-defined edge lengths.

Constrained Geometric Degrees of Freedom ($CGDoF_f$)

The user-defined edge lengths provide linear equations for the edge lengths that are in general non-homogeneous. As a consequence, the dimension of the solution space for possible geometries of the face may decrease significantly.

The user may over-determine the system, for instance, by assigning too many edge lengths. To avoid this problem, we compute the dimension of the constrained solution space, called the Constrained Geometric Degrees of Freedom (CGDoF), using RREF.

The result of this method classifies the edges into the following classes:

- the fixed edges, e_{fix}: the edges chosen by the user with predefined edge lengths (these edges are always dependent edges of the equation system);
- the non-fixed dependent edges, e_{nfd}: the dependent edges which are not predefined by the user, and
- the independent edges, e_{ind}.

To solve the quadratic equation for the area, we reduce the number of independent edges e_{ind} to one, by assigning the existing edge length for all independent edges except one. The last remaining independent edge is called the critical independent edge, e_{ci}, the length of which we find using the quadratic equation. This method will be described in detail in the next sections.

Defining the constrained equations for a face

In Section 2.3, we expressed the area of a face polygon based on a quadratic form of the edge lengths. Now, we develop linear, non-homogeneous constrained equations describing the geometry of the face with preassigned lengths for certain edges of the face.

We can write the edge $\mathbf{e}_i$ with a predefined length q_i as a constraint vector equation in the following way:

$$\mathbf{l}_i^T \mathbf{q} = q_i \tag{20}$$

where $\mathbf{l}_i$ is the $[e \times 1]$ column vector whose entries are all zero (0) except at the index of e_i where it is one (1).

Similarly, multiple constraints, i.e other fixed edge lengths, can be written as a matrix equation

$$\mathbf{L}_f \mathbf{q} = \mathbf{l}_f \tag{21}$$

where the rows of $\mathbf{L}_f$ are the row vectors $\mathbf{l}_i^T$ and $\mathbf{l}_f$ is the vector whose entries are the q_i, the predefined edge lengths.

Together with the equilibrium equations of (2), we obtain all the linear equations describing the linear constraints which results in the constraint equation system

$$\mathbf{B}_f \mathbf{q} = \mathbf{b}_f \tag{22}$$

where the matrix $\mathbf{B}_f$ is obtained by stacking the matrices $\mathbf{E}_f$ and $\mathbf{L}_f$

$$\mathbf{B}_f = \begin{pmatrix} \mathbf{E}_f \\ \mathbf{L}_f \end{pmatrix},$$

and the vector $\mathbf{b}_f$ is obtained as stacking the zero vector and the vector $\mathbf{l}_f$ together

$$\mathbf{b}_f = \begin{pmatrix} \mathbf{0} \\ \mathbf{l}_f \end{pmatrix}$$

We call the matrix $\mathbf{B}_f$ the constraint matrix and the vector $\mathbf{b}_f$ the constraint vector.

Analyzing the constraint equation system (RREF)

The constraint equation system, Eq. (22), is, in general, a non-homogeneous, linear equation system. The solution space of this equation system is often not a linear subspace but rather the empty set or an affine subspace of the possible solution space $\mathbb{R}^e$. Here, e denotes the number of edges. The dimension of this affine subspace is the Constrained Geometric Degrees of Freedom of the face ($CGDoF_f$). The $CGDoF_f$ is the geometric degrees of the face after applying the edge constraints by the user.

It is also possible to have no solutions for Eq. (22). In this case, we say that the $CGDoF_f$ is $-\infty$. If there exists a solution to Eq. (22), then the $CGDoF_f$ is a non-negative integer, which is the dimension of the affine subspace formed by the solutions. When the $CGDoF_f$ is zero, the constraint equations have a unique solution. If the $CGDoF_f$ is positive, then there are many significantly different solutions to the constraint equations.

In order to compute the $CGDoF_f$, we use the reduced row echelon form ($\mathbf{RREF}_f$) of the matrix obtained from the constraint matrix, $\mathbf{B}_f$, and the constraint vector $\mathbf{b}_f$:

$$\left(\mathbf{B}_f \mid \mathbf{b}_f\right).$$

The $CGDoF_f$ can be easily computed from this reduced- row-echelon form, but the following two possibilities might occur:

- If there exists a row of $\mathbf{RREF}_f$, so that the last entry is one, but all other entries are zero:

  $$\left(0 \quad \ldots \quad 0 \mid 1\right)$$

  then, the constraint equation, Eq. (22) has no common solution. In this case, the $CGDoF_f$ is $-\infty$, and the user needs to modify the constraints and/or release some of the constrained edges from their input.
- Otherwise, we have at least one solution. In this case, the $CGDoF_f$ equals $e - r$ where e is the number of edges of the face and the number of columns of the constraint matrix $\mathbf{B}_f$, and r is the rank of the $\mathbf{RREF}_f$. This rank equals the number of pivots, and as a consequence, in this case, it also equals the rank of $\mathbf{B}_f$.

Solving the area equation

The main idea of manipulating the edge lengths of the face to obtain a required area is to solve the area equation (18) by reducing the number of unknown edges to a single unknown edge length.

We reduce the unknowns by finding all the independent edges of the constraint equation system (22) and assigning either the current values or a user-defined values to them.

From now on, we assume that there exists a solution to Eq. (22). The columns in $\mathbf{RREF}_f$ corresponding to the pivots are called the pivotal columns. The non-pivotal columns correspond to the so-called independent edges whose lengths can be manipulated freely. Once the values for the independent edges are set (possibly by the user), there is a unique solution to Eq. (22).

The edges corresponding to the pivotal columns are the edges which depend on the independent ones. These edge lengths will be updated so that Eq. (22) is satisfied.

Our method for solving the area equation (18) is to set as many values of the lengths of the independent edges as possible. In this case, it is one less than the number of independent edges: $e-r-1$. The length of the last independent edge, q_{ci}, is the length of the *critical independent* edge, e_{ci}. This length will be treated as a variable for which we will solve Eq. (18).

To solve Eq. (18), we organize the edges according to the form of $\mathbf{RREF}_f$ into vectors:

- the vector corresponding to the critical edge is defined as an $[e \times 1]$ column vector $\mathbf{q}_{ci}$:

  $$\mathbf{q}_{ci,i} = \begin{cases} q_{ci} & \text{if } i \text{ is the index of the } e_{ci} \\ 0 & \text{otherwise} \end{cases}$$

  Here q_{ci}, the edge length of the e_{ci} edge, is the unknown and we will solve the area equation for q_{ci};

- the vector of the edge lengths of the *non-critical independent* edges, e_{nci}, is defined as $\mathbf{q}_{nci}$ which is an $[e \times 1]$ column vector:

$$\mathbf{q}_{nci,i} = \begin{cases} q_i & \text{if } i \text{ is the index of an } e_{nci} \text{ edge} \\ 0 & \text{otherwise} \end{cases}$$

  where q_i are the current edge lengths of the e_{nci} edges;
- Similarly, the vector of the edge lengths of the fixed /predefined edges, e_{fix} is defined by $\mathbf{q}_{fix}$ as

$$\mathbf{q}_{fix,i} = \begin{cases} q_i & \text{if } i \text{ is the index of a user-selected edge} \\ 0 & \text{otherwise} \end{cases}$$

  where q_i denote the length of the user-selected edge. The indices of fixed edges are indices of some of the pivotal columns. These q_i are fixed in the beginning of the problem and will not be updated;
- Finally, the vector of edge lengths of the *non-fixed dependent* edges, e_{nfd} is defined by vector $\mathbf{q}_{nfd}$ as

$$\mathbf{q}_{nfd,i} = \begin{cases} q_i & \text{if } i \text{ is the index of an } e_{ndf} \\ 0 & \text{otherwise} \end{cases}$$

  The edge lengths q_i of the e_{nfd} edges can be computed from the lengths of the edges corresponding to e_{ci}, e_{nci}, and e_{fix}. The edge lengths will be updated in order to satisfy the area equation.

After setting up the notations, we begin to solve the area equation (18).

Since any edge is either e_{ci}, e_{nci}, e_{fix} or e_{nfd}, we have an equality

$$\mathbf{q} = \mathbf{q}_{ci} + \mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{q}_{nfd}. \tag{23}$$

Moreover, the lengths of the e_{nfd} depend linearly on the lengths of the e_{ci}, e_{nci} and e_{fix}, hence there exist an $[e \times e]$ square matrix, $\mathbf{D}$ and vectors $\mathbf{d}$, $\mathbf{g}$ so that:

$$\mathbf{q}_{nfd} = \mathbf{D}\mathbf{q}_{nci} + \mathbf{d}q_{ci} + \mathbf{g} \tag{24}$$

The matrix $\mathbf{D}$ and the vectors $\mathbf{d}$ and $\mathbf{g}$ can be computed from the RREF form easily as follows.

The matrix $\mathbf{D}$ is a matrix whose entries are mostly 0 except at the entries corresponding to the columns of e_{nci} and to the rows of the e_{nfd}, where the value has the opposite sign than the corresponding value in the $\mathbf{RREF}_f$ matrix of $\left(\mathbf{B}_f \mid \mathbf{b}_f\right)$.

The vector $\mathbf{d}$ is a vector whose entries are mostly zero (0) except at the entries corresponding to the column of the e_{ci} edge and to rows of the dependent edges, where the value has the opposite sign than the corresponding value in the $\mathbf{RREF}_f$ matrix of $\left(\mathbf{B}_f \mid \mathbf{b}_f\right)$.

The vector $\mathbf{g}$ is the contribution coming from the fixed edges. This is a vector whose entries are mostly zero (0), except for entries corresponding to the indices of the e_{nfd} edges, when the entry is the last entry of the corresponding row of the $\mathbf{RREF}_f$ matrix $\left(\mathbf{B}_f \mid \mathbf{b}_f\right)$.

We simplify Eq. (23) slightly. Consider the $[e \times e]$ square matrix $\mathbf{Id}_{nci}$, which is the identity restricted to the e_{nci} edges, and 0 elsewhere. Since,

$$\mathbf{Id}_{nci}\mathbf{q}_{nci} = \mathbf{q}_{nci}$$

we have that

$$\mathbf{q}_{nfd} + \mathbf{q}_{nci} = \mathbf{D}'\mathbf{q}_{nci} + \mathbf{d}q_{ci} + \mathbf{g}$$

where $\mathbf{D}'$ is the matrix $\mathbf{D} + \mathbf{Id}_{nci}$.

Thus, by Eq. (23), we have

$$\mathbf{q} = \mathbf{q}_{ci} + \mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{d}q_{ci} + \mathbf{g}.$$

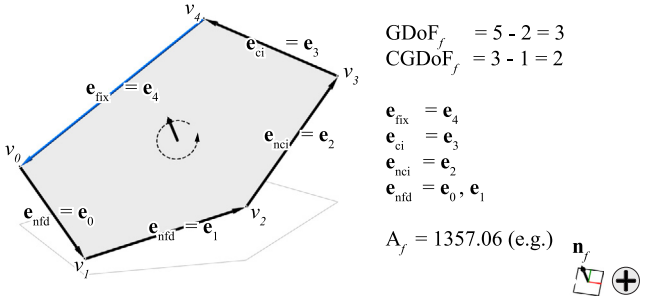

Fig. 7. A sample face showing the edge vectors, its normal direction and the choices of e_{fix}, e_{cr}, e_{nci}, and e_{nfd}.

Let us denote $\mathbf{d}'$ by the vector obtained by adding a 1 to the vector $\mathbf{d}$ at the index of the critical edge, i.e $\mathbf{d}'q_{ci} = \mathbf{q}_{ci} + \mathbf{d}q_{ci}$. Then, we have

$$\mathbf{q} = \mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{d}'q_{ci} + \mathbf{g} \tag{25}$$

Now, we can solve the area equation (18) by plugging in the right-hand side of Eq. (25) into $\mathbf{q}$: the quantity

$$\frac{1}{2k}\left(\mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{d}'q_{ci} + \mathbf{g}\right)^T \mathbf{M}_f \left(\mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{d}'q_{ci} + \mathbf{g}\right) \tag{26}$$

computes the area of the face, A_f. Rearranging the terms, we obtain a quadratic equation for q_{ci}:

$$aq_{ci}^2 + bq_{ci} + c = 0 \tag{27}$$

where

$$a = \mathbf{d}'^T\mathbf{M}_\mathbf{f}\mathbf{d}'$$

and

$$b = 2\mathbf{d}'^T\mathbf{M}_\mathbf{f}\left(\mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{g}\right)$$

and

$$c = \left(\mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{g}\right)^T \mathbf{M}_\mathbf{f}\left(\mathbf{D}'\mathbf{q}_{nci} + \mathbf{q}_{fix} + \mathbf{g}\right) - 2kA_f$$

We can solve this quadratic equation (using the quadratic formula) to obtain possibly two solutions for q_{ci}:

$$q_{ci} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Number of solutions

Depending on the target area, A_f, we might have different number of solutions for Eq. (27). It is possible to have no solution, a unique solution, or two significantly different solutions (see Fig. 8).

Depending on the sign of A, a large positive or a small negative prescribed area ensures that we have multiple (two) solutions.

2.4.1. Updating the edges of the face

Once we computed the length q_{ci} of the e_{ci} edge, we can update the lengths of the dependent edges using Eq. (24). Now, all the lengths of the edges are computed. The face corresponding to the edge lengths has the required area and the edges of the face satisfy the constraint equation system, Eq. (22) while only the lengths of the e_{nfd} edges and the length of the e_{ci} were manipulated (Fig. 10).

The above discussion is summarized in Algorithm 2 in Section 3.2.

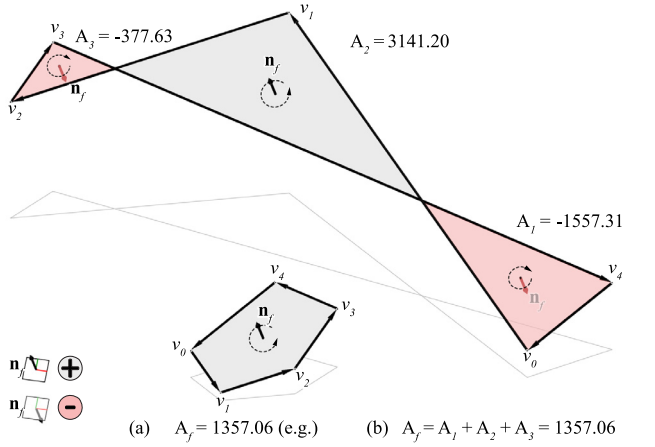

Fig. 8. There are at least two significantly different geometries that represent the same area of a polygon.

2.4.2. Example

Consider the pentagon of Fig. 5. The coordinates of the vertices of the pentagon are the following: $v_0 = (0, 0, 0)$, $v_1 = (6.79, -18, -9)$, $v_2 = (35.3, -21.5, -1)$, $v_3 = (55.7, -8.9, 14.5)$, and finally $v_4 = (36.3, 8.9, 18.7)$. The matrix $\mathbf{M}_f$ for this example is the following:

$$\begin{pmatrix} 0 & 2.872 & 1.85 & -0.506 & 0 \\ 0 & 0 & 1.893 & 1.358 & -0.38 \\ -0.925 & 0 & 0 & 2.97 & 0.577 \\ 1.012 & -0.679 & 0 & 0 & 2.722 \\ 2.986 & 0.761 & -0.288 & 0 & 0 \end{pmatrix}$$

Here, the order of the edges is given as e_0, e_1, e_2, e_3 and e_4. Hence, the matrix is a $[5 \times 5]$ matrix. Its elements were computed using Eq. (17), for instance,

$$\mathbf{M}_{f,12} = 3\frac{|\mathbf{e}_0 \times \mathbf{e}_1|}{|\mathbf{e}_0| \cdot |\mathbf{e}_1|}.$$

We remark also that most entries of the matrix are positive, however, we can see that $\mathbf{M}_{f,14}$ has to be negative, since the vectors $\mathbf{e}_0$, $\mathbf{e}_3$ and $\mathbf{n}$ have negative orientation with respect to the local coordinate of the surface (Fig. 7).

The user selects e_4 as the fixed edge and assigned area of the face to be zero.

The matrix $\mathbf{B}_f$ is given as

$$\begin{pmatrix} 0.319 & 0.956 & 0.714 & -0.731 & 0.867 \\ -0.847 & -0.117 & 0.445 & 0.663 & -0.215 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

and the vector $\mathbf{b}_f$ is

$$\begin{pmatrix} 0 \\ 0 \\ 41.78 \end{pmatrix}.$$

Here the first two rows describe the equilibrium equations, Eq. (2), — for the x- and y-coordinates. The third equation is the constraint equation, Eq. (21), for the fixed edge whose length is 41.78. In our case, the fixed edge corresponds to the last entry.

The $\mathbf{RREF}_f$ matrix is

$$\left(\begin{array}{ccccc|c} 1 & 0 & -0.659 & -0.709 & 0 & -16.602 \\ 0 & 1 & 0.966 & -0.529 & 0 & 43.441 \\ 0 & 0 & 0 & 0 & 1 & 41.78 \end{array}\right). \quad (28)$$

As a consequence, the $CGDoF_f$ can be computed as $e - r = 5 - 3 = 2$.

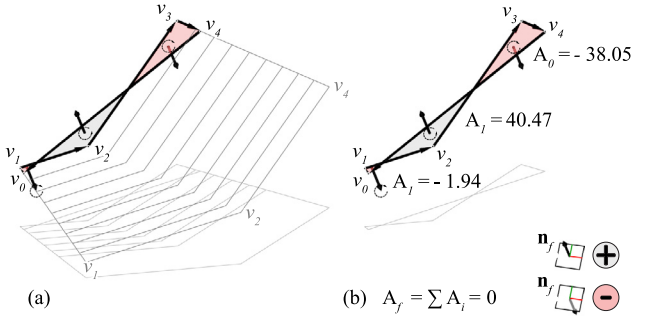

Fig. 9. (a) Multiple steps to compute the new face geometry with the pre-assigned area (for visualization purposes only); and (b) the computed face geometry with zero area.

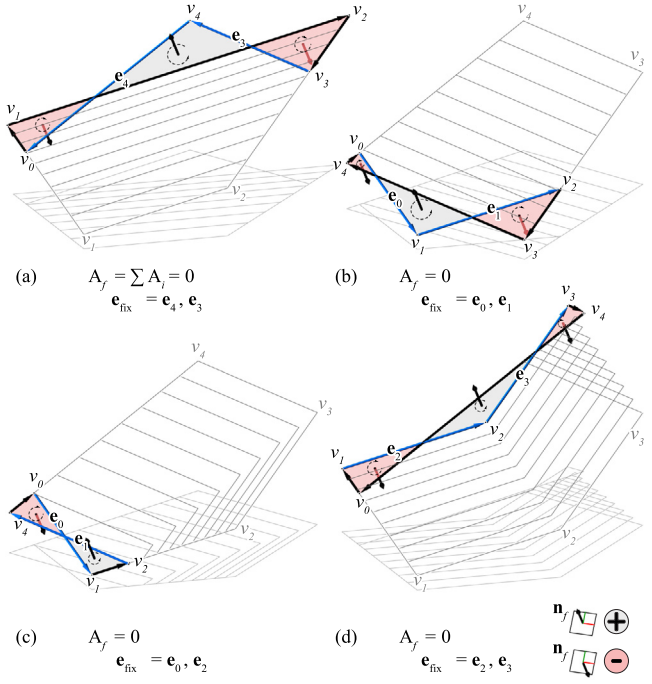

Fig. 10. (a) to (d) Various zero-area computation for a starting face with the area A_f and different chosen fixed edges.

Moreover, we can identify the e_{ci}, e_{nci}, e_{fix} and e_{nfd} edges from the RREF matrix, (28) as follows.

The fixed edge e_4 corresponds to the fifth entry. Since, the $CDGoF_f$ is equal to 2, we have two more $(5 - 2 - 1 = 2)$ dependent edges. These are the e_{nfd} edges, e_0 and e_1, given by the other pivotal columns. The e_{ci} edge was chosen to be the edge corresponding the fourth entry, e_3. Finally, the e_{nci} edge is the remaining edge, e_2 for which we solve the quadratic area equation (see Fig. 7).

Now, we compute the coefficients of Eq. (27) to solve for the edge length, q_{ci} of the e_{ci} edge. Here, the target area, A_f is zero.

$a = 1.796$, $b = 390.646$, $c = 1898.751$

We obtain two solution for Eq. (27)

$q_{ci} = -4.974$ and 212.535.

As a consequence, we get two significantly different solutions for the geometry of this face. The updated (self-intersecting) face corresponding to $q_{ci} = -4.974$ is shown in Fig. 9.

2.5. Computing the polyhedral geometry for target areas

In this process, a user would select multiple internal/external faces and edges of a polyhedral system and would assign target areas for each face and edge lengths for each edge to compute the new geometry of the polyhedron. Computing the geometry of a system of polyhedral cells with pre-assigned areas and fixed edge lengths in one step is a complex task. We propose a multi-step, inductive process to tackle this problem.

2.5.1. Prescribed area for one face

In each step, we compute the geometry of a single face f_i with the assigned area as described in Section 2.4. Then we update the polyhedron with the new edge lengths using Moore–Penrose Inverse (MPI) Method [43,47,48] and move to the next face and repeat this process until there is no face left to change.

First, we identify the fixed edges from the list e_{fix} which lie on the face f_i with prescribed edge lengths, and use RREF method to solve the quadratic area Eq. (18) described in Section 2.4.

In the next step, to preserve the new geometry of the face f_i, we consider all the newly generated edges of the face as fixed edges for the entire polyhedral system, i.e., we update the list of fixed edges e_{fix} for the entire polyhedron with the newly computed edge lengths of f_i.

2.5.2. Non-homogeneous equation system for a polyhedron

Similarly to Section 2.4, Eq. (22), the linear constraint equations for a polyhedral system can be described by two different kinds of linear equations. First, we have the equilibrium equation system, Eq. (3) describing the topology of the polyhedral system. Second, we have the linear constraints coming from the prescribed edge lengths.

As a result, we obtain an equation system that describes the equilibrium of the polyhedron with the prescribed edge lengths of the fixed edges:

$$\mathbf{B}_p\mathbf{q} = \mathbf{b}_p \tag{29}$$

where the constraint matrix

$$\mathbf{B}_p = \begin{pmatrix} \mathbf{E}_p \\ \mathbf{L}_p \end{pmatrix}$$

is built from the equilibrium matrix of the polyhedron $\mathbf{E}_p$ (see Eq. (3)) and the constraint equations $\mathbf{L}_p\mathbf{q} = \mathbf{l}_p$ coming from the fixed edges (see Eq. (21)). Similarly, the vector

$$\mathbf{b}_p = \begin{pmatrix} \mathbf{0} \\ \mathbf{l}_p \end{pmatrix}$$

is obtained from the edge vector of the fixed edges $\mathbf{l}_p$.

2.5.3. Solving non-homogeneous equation systems using MPI

Now, we propose to solve Eq. (29) using the Moore–Penrose Inverse (MPI) method. The MPI method is a technique to solve a general non-homogeneous equation system of the matrix form

$$\mathbf{B}_p\mathbf{q} = \mathbf{b}_p \tag{30}$$

where the matrix $\mathbf{B}_p$ and the vector $\mathbf{b}_p$ are given.

We represent MPI of the matrix $\mathbf{B}_p$ by $\mathbf{B}_p^+$ that satisfies the following matrix equations

$$\mathbf{B}_p\mathbf{B}_p^+\mathbf{B}_p = \mathbf{B}_p \quad \text{and} \quad \mathbf{B}_p^+\mathbf{B}_p\mathbf{B}_p^+ = \mathbf{B}_p^+.$$

Assume that the vector $\mathbf{b}_p$ is of the form $\mathbf{B}_p\mathbf{q}$ for some vector $\mathbf{q}$. Multiplying the first equality by $\mathbf{q}$, we have

$$\mathbf{B}_p\mathbf{B}_p^+\mathbf{B}_p\mathbf{q} = \mathbf{B}_p\mathbf{q}.$$

As a consequence, we obtain

$$\mathbf{B}_p\mathbf{B}_p^+\mathbf{b}_p = \mathbf{b}_p. \tag{31}$$

Therefore, if a solution to Eq. (30) exists, then Eq. (31) has to be satisfied. Similarly, if Eq. (31) is satisfied, then the vector $\mathbf{B}_p^+\mathbf{b}_p$ is a solution to Eq. (30) This provides an effective tool to check whether Eq. (30) has a solution or not.

From now on, we assume that Eq. (31) holds, in other words, we assume that Eq. (30) has a solution. In this case, any vector $\mathbf{q}$ of the form

$$\mathbf{q} = \mathbf{B}_p^+\mathbf{b}_p + (\mathbf{Id} - \mathbf{B}_p^+\mathbf{B}_p)\nu \tag{32}$$

solves the linear equation system Eq. (30) where $\mathbf{Id}$ is the identity matrix and ν is any column vector of the right dimension. In fact, these are all the solutions to Eq. (30). Summarizing the above discussion, we have at least one solution to Eq. (30) if and only if Eq. (31) holds for $\mathbf{b}_p$. Moreover, if there is a solution to Eq. (30), then all solutions have the form of Eq. (32) [47,48].

In Eq. (32), the parameter ν is freely chosen by the user to control the solution. In our examples, we take ν to be the initial edge lengths of the polyhedron, resulting in a solution to Eq. (32) which is the new geometry for the initial polyhedron with the prescribed area for face f_i. In this case, the new edge lengths are the best fit (least squares) to the initial edge lengths. Also, in many cases, only certain parts of the polyhedron change significantly (see Figs. 11 and 14).

Another approach could be to take ν to be the vector whose entries are all 1, in this case, we get a solution with well-distributed edge lengths.

2.5.4. Updating the polyhedral geometry with multiple prescribed face areas

The previously discussed method can compute the geometry of a polyhedral system with multiple faces with prescribed areas in an inductive process. In each step, we update the geometry of the polyhedron using Eq. (32) with the newly computed face whose edge lengths are added to the list of fixed edges. The new edge lengths change the constraints equations $\mathbf{L}_p\mathbf{q} = \mathbf{l}_p$ to compute the polyhedral geometry.

We summarize the process in Algorithm 3.

2.6. Updating the internal forces in the dual diagram

Let us call the starting diagram the *primal*, Γ, and the reciprocal perpendicular polyhedron *dual*, $\Gamma^\dagger$ (Fig. 12a, b). The vertices, edges, faces, and cells of the primal are denoted by v, e, f, and c respectively, and the ones of the dual are super-scripted with a dagger (†) symbol (Fig. 12a,b).

Since the face $f_i^\dagger$ is a closed polygon, the sum of the edge vectors $\mathbf{e}_j^\dagger$ should be zero. Hence, we obtain a vector equation similar to Eq. (1)

$$\sum_{f_j} \mathbf{u}_j^\dagger q_j^\dagger = \mathbf{0}$$

where the sum runs over the attached faces f_j of the edge e_i of the primal Γ; $\mathbf{u}_j^\dagger$ denotes the unit directional vector corresponding to the edge vector $\mathbf{e}_j^\dagger$; and $q_j^\dagger$ denotes the edge length of $\mathbf{e}_j^\dagger$ in the dual $\Gamma^\dagger$.

Similarly, as before, each vector equation for the face of the dual diagram yields three linear equations for the edge lengths, and we obtain a linear equation system for the edge length vector $\mathbf{q}^\dagger$ which can be described by a $[3e \times f]$ matrix that we call the equilibrium matrix of the dual diagram, $\mathbf{E}^\dagger$,

$$\mathbf{E}^\dagger\mathbf{q}^\dagger = \mathbf{0}. \tag{33}$$

In [43], three different methods were described to generate the dual diagram from Eq. (33). In this paper, we choose to use the Moore–Penrose Inverse (MPI) method to initially construct

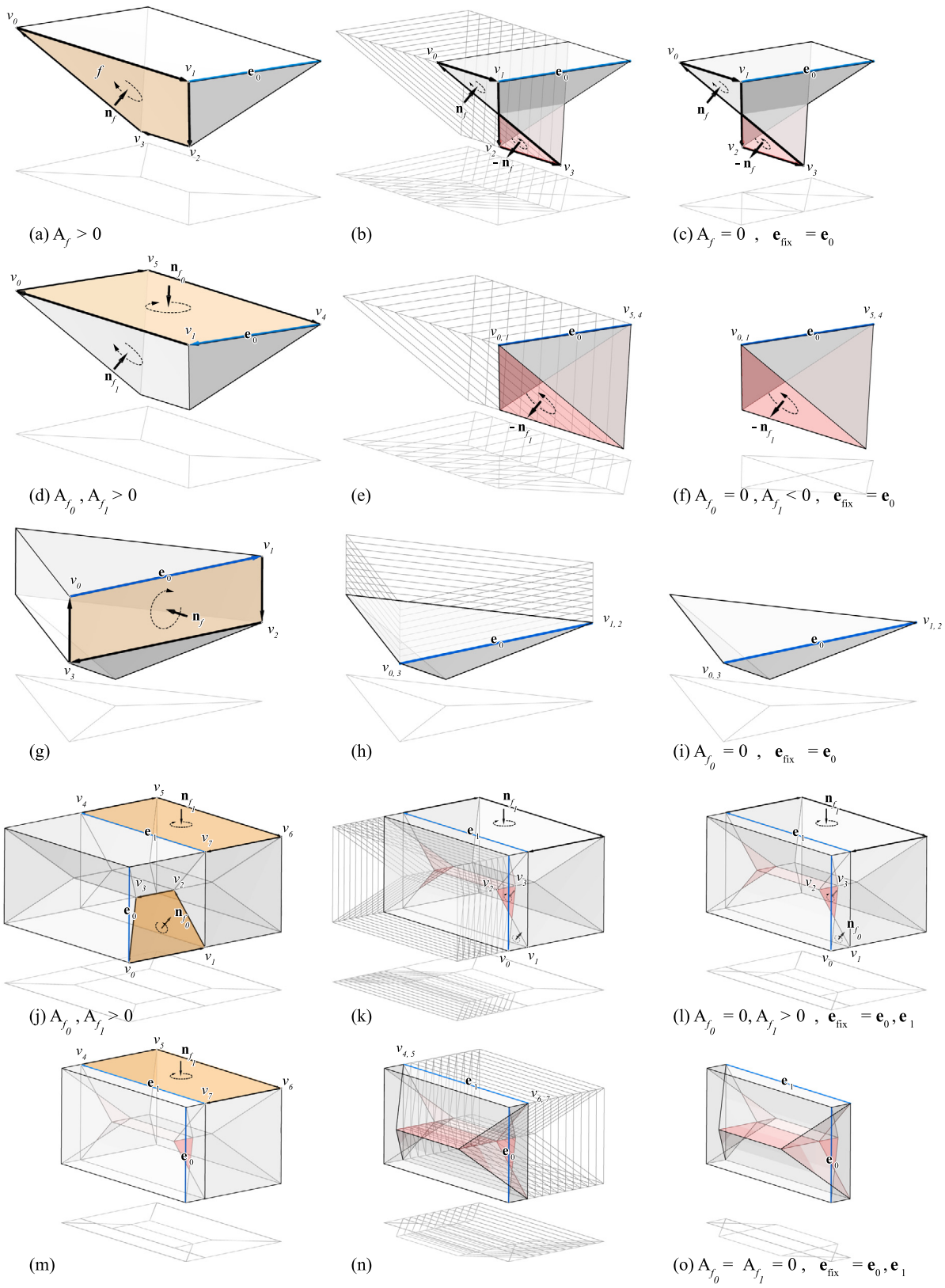

Fig. 11. (a), (d), (g), (j), (m) Multiple polyhedral geometries with selected faces (orange) and user-assigned fixed edges; (b), (e), (h), (k), (n) the face area computation and visualization in multiple steps (for visualization purposes only); and (c), (f), (i), (l), and (o) the resulting polyhedral geometries with zero areas for the selected faces. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the dual diagram before we apply any changes to the force and its face areas. The MPI method of this section is as same as the method described in Section 2.5.3 with $\mathbf{b}_p$ being the zero vector. As a result, the solutions to Eq. (33) can be described as

$$\mathbf{q}^{\dagger} = \left(\mathbf{Id} - (\mathbf{E}^{\dagger})^{+}\mathbf{E}^{\dagger}\right)\xi.$$

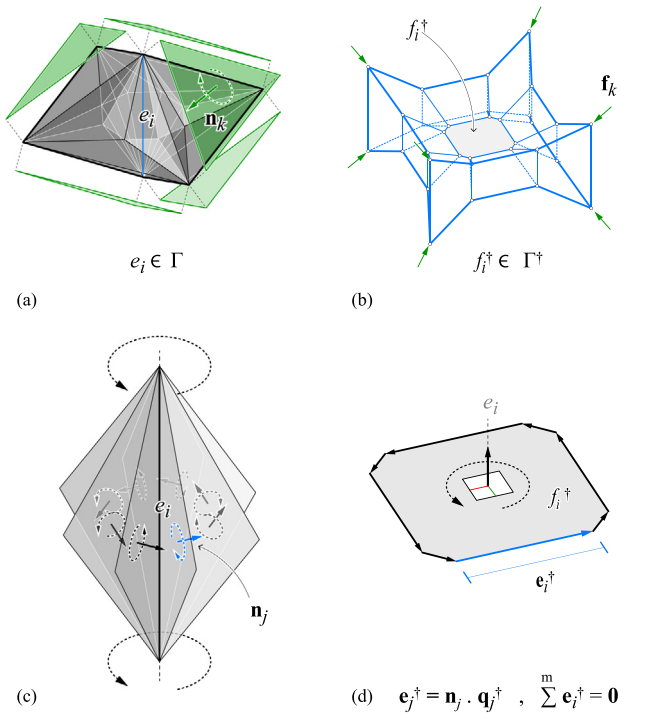

Fig. 12. (a) An input force polyhedron as primal and its corresponding (b) funicular polyhedron as the dual; (c) going around each edge of the primal with its attached faces (c) provides the direction of the edge vectors of the corresponding face (e) in the reciprocal diagram where the sum of the edge vectors must be zero.

Here $(\mathbf{E}^\dagger)^+$ denotes the MPI of the equilibrium matrix $\mathbf{E}^\dagger$. For the parameter ξ, we choose the vector whose entries are all 1 to obtain a dual diagram with well-distributed edge lengths.

The direction of the internal forces

The initial direction of the internal force as compression or tension is stored and altered after the computation of the force with prescribed areas. The tensile force members are updated in the form if the normal of a face in the force diagram flips after the computation. As shown, the geometry of a face can become self-intersecting in some cases. On such occasions, if the area of the region with the initial normal direction is bigger than the area with the flipped normal, then the direction of the initial internal force does not change; otherwise, the direction of the internal force will flip. If the face is a zero area face, the member will carry no force and can disappear in the form diagram (Fig. 14).

3. Implementation

In this section, we explain the computational setup for the methods described in Section 2. The input for this framework is a polyhedral system with planar faces and is considered as the force diagram for the methods of 3D graphic statics. we compute the dual geometry of the updated primal diagram according to Hablicsek et al. [43].

The user can initially select certain edges in the system and assign a target length per selected edge. Similarly, s/he can select multiple faces and assign an area per face. Our method is a sequential computational setup that updates the geometry of the polyhedral system for each user-selected face at each step. For instance, let us assume that the user selects three faces with a target area per face. We start from a face, compute the new geometry of the face with the target area, update the geometry of the polyhedral system based on the new geometry of the face, and move on to the next face and continue the computation until there is no face left.

Accordingly, in each step of the computation, we construct the area matrix and the equilibrium matrix for the user-selected face. We then add the user-defined edge constraints as non-homogeneous linear equations and compute the Constrained Geometric Degrees of Freedom, *CDGoF* using RREF method explained in Section 2.4. In the next step, we compute the geometry of the face, and then we update the geometry of the polyhedral system using MPI method described in Section 2.5.

After the multi-step computation process is completed, we update the direction and the magnitude of the forces in the members of the dual that was initially constructed.

The above description can be summarized into three main sections as shown in the flowchart of Fig. 13. These sections are as follows:

- computing the new geometry for a face with constrained edges and areas;
- updating the new geometry of the polyhedral system based on the newly-computed face geometry and the fixed edges; and,
- updating the internal forces in the members of the dual based on the new force magnitudes.

Sections 3.1, 3.2, and 3.3 provide additional details of the algorithms used in this process. These algorithms include: computing the area matrix for a face; face reconstruction with constrained edges and target area; and updating the geometry of the polyhedron with constrained areas of faces and edge lengths.

3.1. Constructing the area matrix

Algorithm 1 receives a face f_i with an ordered list of vertices $[v_0, \ldots, v_{k-1}]$ as an input and outputs a symmetric matrix $\mathbf{M}_{f_i}$ which is used in the quadratic form, Eq. (18), to compute the area of the face.

First, we choose an (arbitrary) normal vector for the face, by taking the unit cross product of the first two consecutive edge vectors. Then, we construct the matrix $\mathbf{M}_{f_i}$ row by row as follows: starting from each vertex v_n, we create an ordered list of directed edges $\mathbf{e}_i$, and compute the scalars $\eta_{i,j}$ as explained in Eq. (9).

Once the whole matrix $\mathbf{M}_{f_i}$ is constructed, the algorithm outputs a symmetric matrix (see Eq. (19)) to be used in the quadratic form for computing the area of the face f_i.

3.2. Updating the geometry of a face with constrained edges and a target area

Algorithm 2 updates the geometry of a face with constrained edges and a target area. The input of this algorithm is a user-selected face f_i with a target area A_{f_i} and (user-selected) edges of $\bar{e}_{fix}$ with prescribed edge lengths.

First, we compute the linear constraint equations, Eq. (22), where the equilibrium equations describe the geometry of the face and the constraint equations come from the (user-selected) edge constraints.

Once, the constraint equation system, Eq. (22), is created, we compute the Constrained Geometric Degrees of Freedom of f_i, $CDGoF_{f_i}$, of the face using RREF method. If $CDGoF_{f_i} = -\infty$, the algorithm stops, i.e., the constrained equation system, Eq. (22), cannot be solved. In this case, the user may modify the input by selecting less constrained edges or by selecting different edges.

If $CDGoF_{f_i}$ is at least zero, the algorithm classifies the edges of f_i into *ci*, *nci*, *fix* and *nfd* edges. Next, the we construct the area

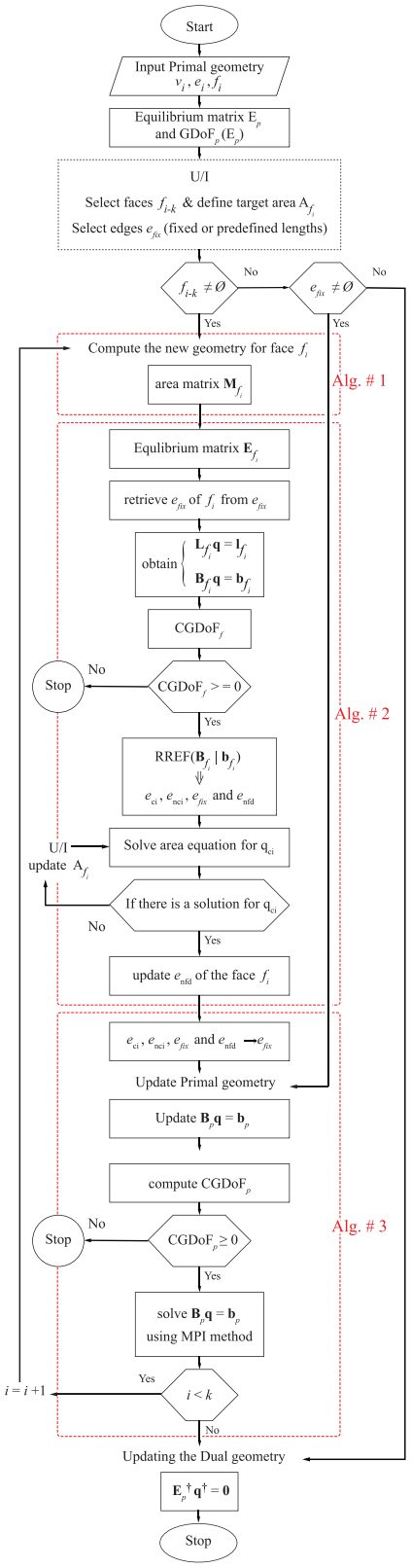

Fig. 13. The flowchart expanding multiple steps in computing the primal geometry/force diagram with preassigned edge lengths and face areas, and the updated dual geometry as the form diagram.

matrix $\mathbf{M}_{f_i}$ using Algorithm 1. Using the area equation, Eq. (18), and Eq. (27), we compute the edge length q_{ci} of the edge e_{ci}. If

Algorithm 1: Computing the area matrix $\mathbf{M}_{f_i}$

Input: $f_i : [v_0, v_n, ..., v_{k-1}]$ the ordered list of vertices around f_i

Output: $\mathbf{M}_{f_i}$ the area matrix of the face f_i.

Function $M_{row}(\overline{v}_{row}, \mathbf{n}_f)$**:**

- $\overline{v}_{row} : [v_j, v_{j+1}, ..., v_{j-1}]$ # ordered list of vertices starting from v_j
- **for** $v_n \in \overline{v}_{row}$ **do**
  - $\overline{e} \longleftarrow e_i[v_n, v_{n+1}]$ # ordered edges
- **for** $e_i \in \overline{e}$ **do**
  - $\mathbf{e}_i \longleftarrow \langle x_{n+1} - x_n, y_{n+1} - y_n, z_{n+1} - z_n \rangle$ # direction vector from v_n to v_{n+1}
  - $|\mathbf{e}_i| \longleftarrow l_i$ # length of the vector $\mathbf{e}_i$
- **for** $e_i \in E$ **do**
  - $\mathbf{n}_{0,i} = \mathbf{e}_0 \times \mathbf{e}_i$ # cross product of $\mathbf{e}_0$ and $\mathbf{e}_i$
  - **if** $x_{\mathbf{n}_f} \neq 0$ # *the x coordinate of* $\mathbf{n}_f$ **then**
    - $\eta_i = x_{\mathbf{n}_{0,i}} / (x_{\mathbf{n}_f} * |\mathbf{e}_i| * |\mathbf{e}_0|)$
  - **else**
    - **if** $y_{\mathbf{n}_f} \neq 0$ # *the y coordinate of* $\mathbf{n}_f$ **then**
      - $\eta_i = y_{\mathbf{n}_{0,i}} / (y_{\mathbf{n}_f} * |\mathbf{e}_i| * |\mathbf{e}_0|)$
    - **if** $z_{\mathbf{n}_f} \neq 0$ # *the z coordinate of* $\mathbf{n}_f$ **then**
      - $\eta_i = z_{\mathbf{n}_{0,i}} / (z_{\mathbf{n}_f} * |\mathbf{e}_i| * |\mathbf{e}_0|)$
  - $M_{e_i} = (k - 1 * i - 1) * \eta_i$ # matrix coefficient for edge e_i where k is the length of V_{row}
  - $\mathbf{M}_{row} \longleftarrow M_{e_i}$
- **return** $\mathbf{M}_{row}$

begin

- $\mathbf{e}_0 \longleftarrow$ # vector from v_0 to v_1
- $\mathbf{n}_{f_i} \longleftarrow \mathbf{e}_0 \times \mathbf{e}_1$ # the cross product of the first two edges of f_i
- **for** $v_i \in \{v_0, ..., v_{k-1}\}$ **do**
  - $\overline{v}_{v_i} \longleftarrow [v_i, v_{i+1}, ..., v_{i-1}]$ # ordered list of vertices starting at v_i
  - $\mathbf{M}_{v_i} = \mathbf{M}_{row}(\overline{v}_{v_i}, \mathbf{n}_{f_i})$
- $\mathbf{M}_{f_i} \longleftarrow \mathbf{M}_{v_i}$ # matrix whose rows are the $\mathbf{M}_{v_i}$
- $\mathbf{M}_{f_i} = \frac{1}{2}\left(\mathbf{M}_{f_i} + \mathbf{M}_{f_i}^T\right)$ # output is a symmetric matrix

Eq. (27) has no solution, the algorithm may ask the user to modify the target area. Eq. (27) often has two solutions, the user may choose from those particular solutions (see Fig. 8).

Once a solution is chosen for q_{ci}, the algorithm updates the lengths of the e_{nfd} (see Section 2.4.1) and outputs the updated lengths of the edges of f_i.

3.3. Updating the geometry of the polyhedron

The last algorithm updates the geometry of a polyhedron after the new geometry of each face is computed. In fact, this is a multi-step process, in each step, we update the geometry of the polyhedron after updating the geometry of a face. Note that, in each step, the list of fixed edges is updated after computing the geometry of each face, in other words, we solve for the geometry of the polyhedron with more and more constraints.

Algorithm 3 describes the computation of the new geometry of a polyhedron with a given list of constrained edges, $\overline{e}_{fix}$. It computes the linear constraint equation system, Eq. (29). This is a non-homogeneous linear equation system which can be solved using MPI method. The parameter ν in the MPI method can be

Algorithm 2: Updating the geometry of the face (UF)

Input: $\begin{cases} f_i : [v_0, ..., v_{k-1}] & \text{face with ordered list of vertices} \\ A_{f_i} & \text{target area for the face} \\ \bar{e}_{fix} : [e_m, ..., e_q] & \text{list of constrained edges} \end{cases}$

Output: $Q : [q_0, ..., q_{k-1}]$ list of edge lengths of f_i with area A_{f_i}.

begin

- $\bar{e} \longleftarrow [e_0, ..., e_{k-1}]$ # ordered list of edges
- **for** $e_i \in \bar{e}$ **do**
  - $\mathbf{e}_i \longleftarrow \langle x_{n+1} - x_n, y_{n+1} - y_n, z_{n+1} - z_n \rangle$ # direction vector from v_n to v_{n+1}
  - $|\mathbf{e}_i| \longleftarrow l_i$ # length of the vector $\mathbf{e}_i$
  - $\mathbf{u}_i = \mathbf{e_i}/|\mathbf{e_i}|$ # unit direction vector of e_i
  - $\mathbf{E}_x \longleftarrow \mathbf{x}_{\mathbf{u}_i}$ # row vector of the x-coordinates of $\mathbf{u}_i$
  - $\mathbf{E}_y \longleftarrow \mathbf{y}_{\mathbf{u}_i}$ # row vector of the y-coordinates of $\mathbf{u}_i$
- $\mathbf{E}_{f_i} \longleftarrow \mathbf{E}_x, \mathbf{E}_y$ # the $[2 \times e]$ equilibrium matrix of the face f_i
- **for** $e_i \in \bar{e}_{fix}$ **do**
  - $\mathbf{l}_i, q_i$ # the row vector of the constraint equation for a fixed edge Eq. (20)
- $\mathbf{B}_{f_i}, \mathbf{b}_{f_i}$ # create the constraint equation system Eq. (22)
- $RREF((\mathbf{B}_{f_i}|\mathbf{b}_f))$ # compute the RREF of the constraint equation system
- $CGDoF_{f_i}$ # compute the CGDoF of the system
- **if** $CGDoF_{f_i} = -\infty$ **then**
  - no solution $\Longrightarrow$ end program or ask user to modify the input.
- **else**
  - $\mathbf{q}_{nci}$ # identify the nci edges
  - $\mathbf{D}, \mathbf{d}, \mathbf{g}$ # coefficients of Eq. 2.4
- $\mathbf{D}', \mathbf{d}' \longleftarrow \mathbf{D}, \mathbf{d}$ # coefficients of Eq. (25)
- $\mathbf{M}_{f_i}$: # output of Algorithm 1
- a, b, c # coefficients of Eq. (27) using $\mathbf{M}_{f_i}$
- $q_{ci} \longleftarrow a, b, c$ # compute the solution(s) of Eq. (27)
- $\mathbf{q}_{nfd} \longleftarrow q_{ci}, \mathbf{q}_{nci}, \mathbf{q}_{fix}$ # update the edge lengths
- $Q = [q_0, ..., q_{k-1}]$ # the list of updated edge lengths

chosen by the user or can be the vector of the initial values of the edge lengths of the polyhedron.

Algorithm 3: Updating the geometry of the polyhedron

Input: $\bar{e}_{fix} : [e_m, e_n, ..., e_q]$: list of fixed edges

Output: $Q_p : [q_1, ..., q_e]$ the updated list of edge lengths of the polyhedron

begin

- **for** $e_i \in \bar{e}_{fix}$ **do**
  - $\mathbf{l}_i, q_i$ # the row vector of the constraint equation for a fixed edge (Eq. (20))
- $\mathbf{B}_p, \mathbf{b}_p$ # create the constraint equation system (Eq. (29))
- $\mathbf{B}_p^+ \longleftarrow$MPI($\mathbf{B}_p$)
- Q_p # update the edge lengths of the polyhedron using MPI method with a fixed parameter ν (Eq. (29))

4. Applications/results

Figs. 11, 14, 15, and 16 illustrate the potential of using this approach in polyhedral transformation with target areas of faces and constrained (selected) edge lengths, and their corresponding structural forms. In the examples shown in Fig. 11, the intention is to highlight certain properties of the constrained polyhedral computation. In all the examples, the chosen face is highlighted by an orange shade and the constrained edges are highlighted by blue color.

For instance, in Fig. 11a, the target area for the chosen face is zero and an edge has been chosen that does not belong to the selected face. Fig. 11b shows an animated drawing for clarification purposes. As shown in Fig. 11c, the selected face turns into a self-intersecting face. In the next example, in Fig. 11d, the top face is set to be zero with the same constrained edge. As a result of this computation, the face collapses to a line due to its rectangular geometry. Also, the normal of the side faces, $\mathbf{n}_{f_0}$, flips which means that the magnitude of the internal force in the corresponding form will change (Fig. 11h, i). In the Example of Fig. 11g, one of the vertical side faces with a constrained top edge is chosen and the target area is also set to zero. As illustrated in Figs. 11h and 11i, the rest of the vertical faces will disappear after the polyhedral computation. This method can certainly be applied to more than one face in the polyhedral system. In Fig. 11j, one external and one internal face are chosen with a zero area target and as illustrated in Figs. 11k–o, the computation process proceeds sequentially by first, computing the geometry of the face f_0 in Fig. 11l and then recomputing the polyhedral geometry to solve for the new face f_1. In this process, multiple other faces will also turn into a self-intersecting face as shown in Fig. 11l. In the final geometry, the face f_1 will collapse to an edge e_1 (Fig. 11o).

In most of these examples, the target areas were intentionally set to be zero to highlight its resulting reciprocal structural form. Fig. 14a shows the same polyhedron of Fig. 11j. In this figure, certain faces in the force diagram and their corresponding edges in the form diagram are highlighted to show the effect of changing the areas on the magnitude of the internal forces. Starting from the force polyhedron of Fig. 14a (top) and its compression-only form (bottom), faces f_1 and f_0 are emphasized in Fig. 14b (top) and their corresponding compression-only edges in the form. The result of the zero area computation results in face f_0 to turn into a self-intersecting face together with other similar faces attached to the edges of the face f_1. Besides, face f_1 is flipped as well as the direction of its normal vector. Note that as a result of this transformation, the internal force in the corresponding member of the face f_0 is decreased to zero. This is a fascinating effect in the equilibrium of polyhedral frames, as it describes the internal equilibrium of forces in a polyhedral system where the edges or the members can be removed from the system without disturbing the internal and external equilibrium. The zero-force edges have been previously observed in some polyhedral reciprocal diagrams by McRobie [7], but there was no method to compute them, particularly in self-intersecting faces as described in the methodology section. Also, the change in the direction of the normal of the face f_1 results in a reversal of the internal force in the edge e_1 of the form diagram.

Fig. 14d emphasizes the faces f_2, f_3, and f_4 and faces that will be affected as a result of the second step of the computation. As shown in Fig. 14e, the normal of the faces f_3 and f_4 will invert the direction of the internal forces in the form diagram. This transformation also removes the applied load in the system as the area of the face f_2 is zero. In another example, the zero area faces are used to completely remove the external horizontal forces in the system. Fig. 15 shows a force diagram and its corresponding form in another transformation. In this example, one of the vertical faces is chosen and the target area is set to zero. Note that as animated in the drawings of Fig. 15b, all the side faces collapse into a line and the top and bottom faces of the polyhedron become coplanar. This transformation results in the disappearance of all the horizontal applied loads in the system as shown in Fig. 15f. The most interesting geometric outcome of this process is the transformation of the internal face f_1. Face f_1 changes its direction and so does its corresponding edge at the

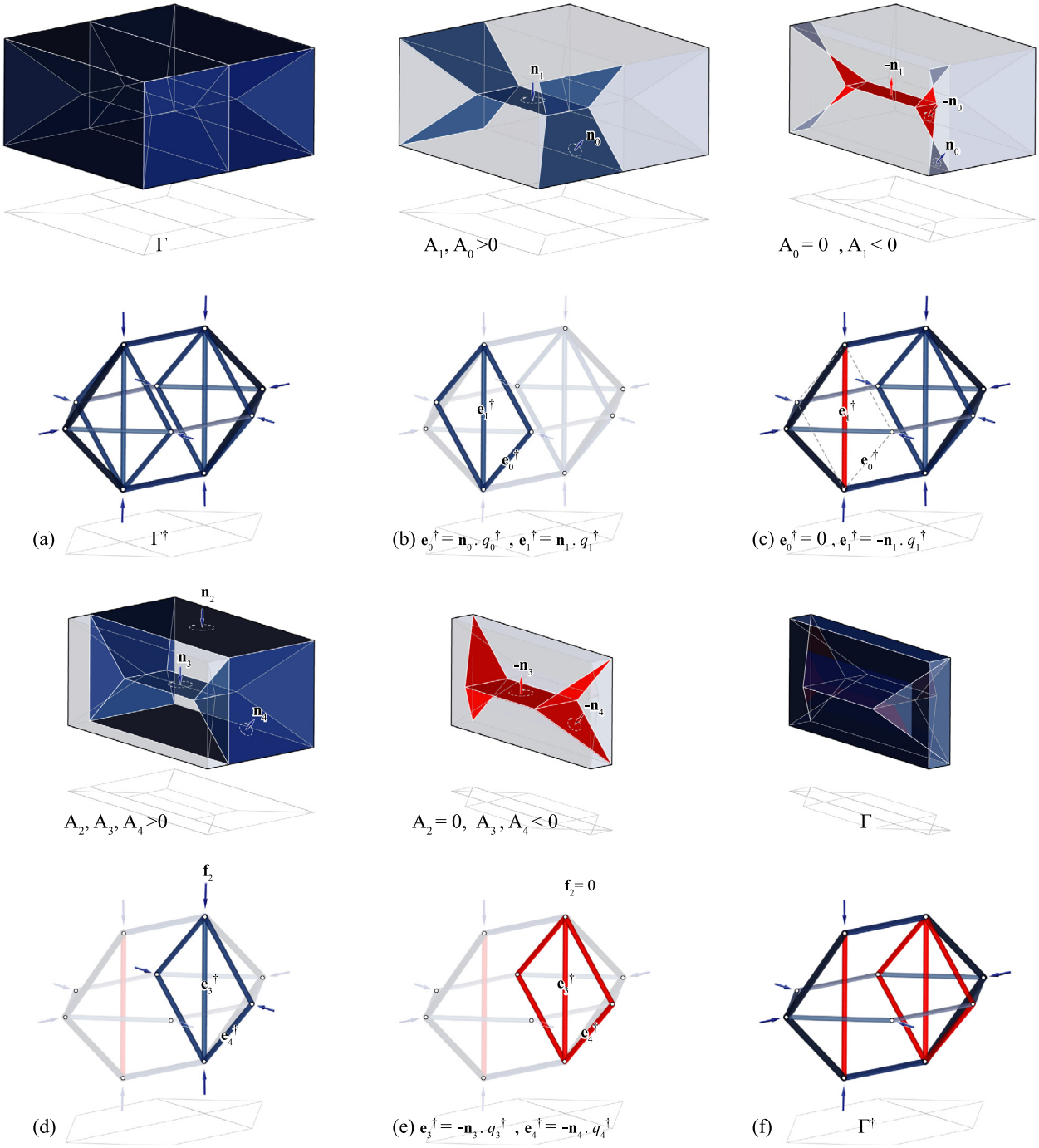

Fig. 14. (a) Initial force polyhedron of Fig. 11j (top) and its reciprocal form (bottom) as a compression-only system; (b) highlighted internal faces of the polyhedron before transformation (top) and the corresponding members in the form (bottom); (c) zero area faces and their updated normal directions and the resulting zero-force members in the form; (d), (e) and (f) highlighted faces in the second step of the transformation and the updated form with new internal force distribution.

boundary of the form diagram. The resulting structure shows a funnel shape compression-only structure with tensile members on the top.

Fig. 16 illustrates another example of using this approach in structural form-finding of a funicular spatial structural form with both tensile and compressive forces in equilibrium. Fig. 16a shows the force diagram consisting of groups of polyhedral cells with planar faces. These faces establish groups of convex polyhedral cells as a force diagram Γ that corresponds to a compression-only form $\Gamma^{\dagger}$ [13]. Using the algorithm explained in this paper, a designer chooses f_1 and f_2 of the global (external) force polyhedron and assigns zero for their target areas (Fig. 16b). Fig. 16c shows the first step of the computation where the area of the face f_1 becomes zero and consequently its corresponding force in the form diagram $\Gamma^{\dagger}$ disappears.

Fig. 16d shows the second step of the computation that makes the area of the face f_2 zero. Together with faces f_1 and f_2 all surrounding faces will also become zero and consequently all the horizontal forces in the system will disappear (Fig. 16d). Note that in all steps of the computation the rest of the faces of the polyhedral system will be updated to match the coordinates of the newly created zero-area faces. As a result the faces corresponding to the top and bottom members flip that reverses the

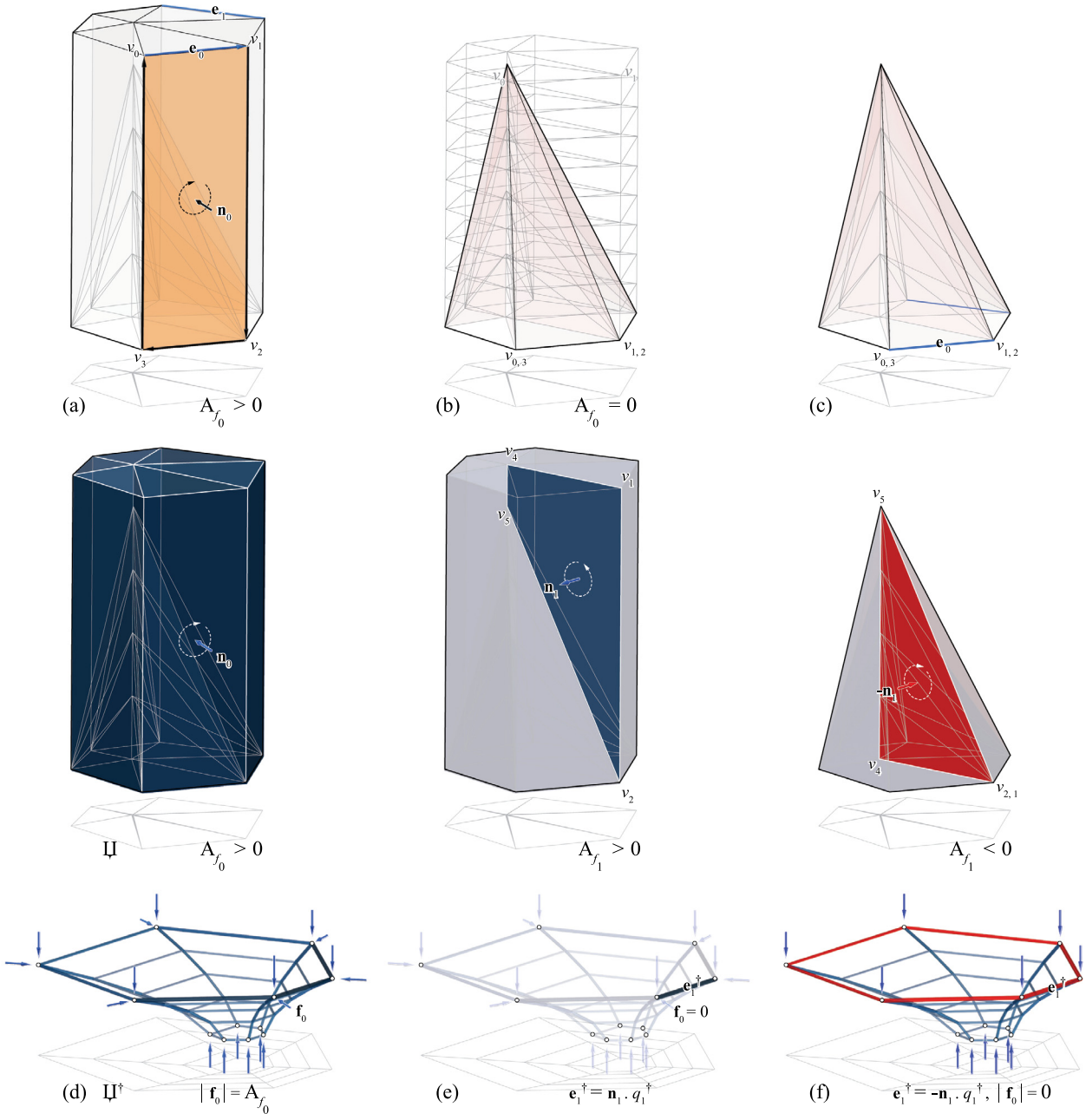

Fig. 15. (a) A force polyhedron and a selected face with zero area target, and two fixed edges on the top; (b) the transformation animation (for visualization purposes only); (c) the resulting force polyhedron where the selected face and all other side faces collapse to a line; (d), (e), and (f) the force and form diagrams and their transformation after changing the areas.

direction of the internal force into tension. The resulting structural form is a funicular form with both tensile and compressive forces with no horizontal reactions at the support (Fig. 16f) (see Figs. 17 and 18).

4.1. Edge lengths of the force diagram vs. form diagram's static indeterminacy

Graphic statics allows for an easy understanding and explanation of the concept of indeterminacy in the form diagram. The Geometric Degrees of Freedom (GDoF) of the force diagram are equal to the degrees of indeterminacy (states of self-stress) of the form [43,49]. As discussed extensively in this paper, the GDoF of a single polygon with e number of edges is equal to $e - 2$. If the GDoF of a force polygon is $e - 2 > 0$, then its corresponding form is statically indeterminate. For a force diagram consisting of multiple faces, the GDoF can be found algebraically by solving all the equilibrium equations of the closed polygons of the faces (see [44]). The number of independent edges in the mentioned equation system is equal to the GDoF of the force diagram, which is equivalent to the states of self-stress of the form. Interestingly, the Geometric Degrees of Freedom (GDoF) of the force polyhedron's external faces correspond to the external degrees of static indeterminacy (states of self-stress) of the form. This is the number of independent edge lengths that can be chosen to construct the external force polyhedron faces with various areas. The external static indeterminacy of the form also means that there are multiple solutions for the geometry and magnitude of the faces belonging to the reaction forces if the areas of the applied loads stay intact. The GDoF of the internal faces of the

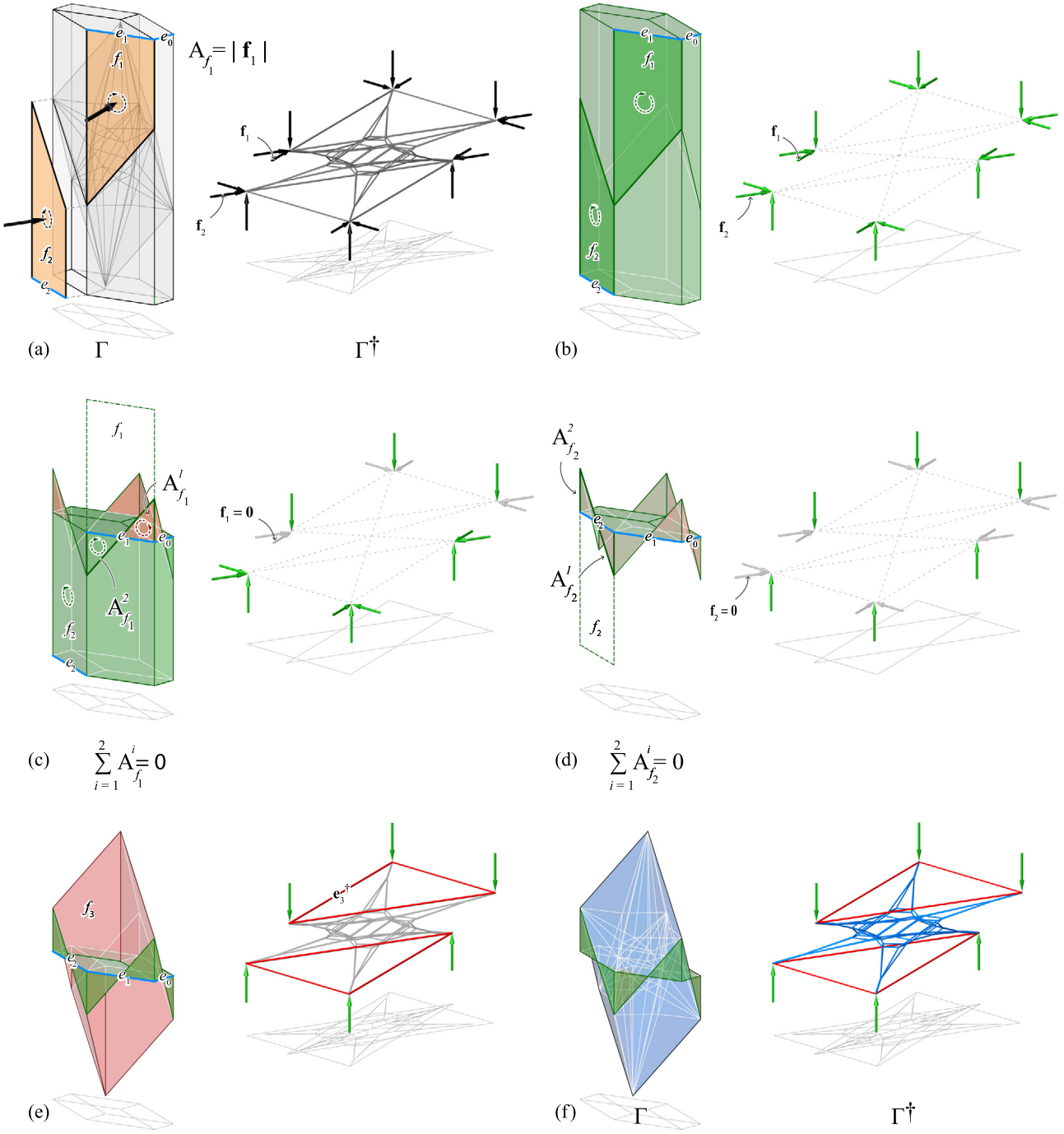

Fig. 16. (a) The force Γ and form $\Gamma^{\dagger}$ diagrams of a compression-only funicular polyhedral structural form; (b) changing the areas of the chosen faces of the force polyhedron to zero; (c) the first step of the computation; (d) the second step of the computation; (e) the tensile members; and (f) the entire updated polyhedron and its corresponding structural form.

force diagram also correspond to the internal degrees of static indeterminacy that can be found numerically using the methods suggested by [50].

5. Computational sensitivity and limitations

The proposed method of this paper is a sequential process, i.e., the algorithm runs as many times as the number of chosen faces with an assigned target area. For instance, if three faces with specific target areas are selected, the algorithm will run three times to change the areas. It starts with the first face area, then goes to the second, and finally to the third. The computation speed has not been a matter of concern since the quadratic equations in this approach are turned into a linear equation system and can be solved using matrix multiplication. The computational speed of the main examples of this paper (Figs. 14, 15, 16) is provided in Table 2. The computational implementation can certainly be improved in the future to include user experience features.

It is worth mentioning that after finding the target area and the new geometry of the face in each step, the edges of the face will be fixed to prevent the area change in the following step. This approach might cause a problem in some cases since a new face's target area may not be achievable given the constraints of the fixed edges of the previously calculated faces. The reason

Fig. 17. The built structure designed using the methods of this paper as part of the *Spatial Efficiency* Exhibition at the Center for Architecture and Design in Philadelphia in January 2020.

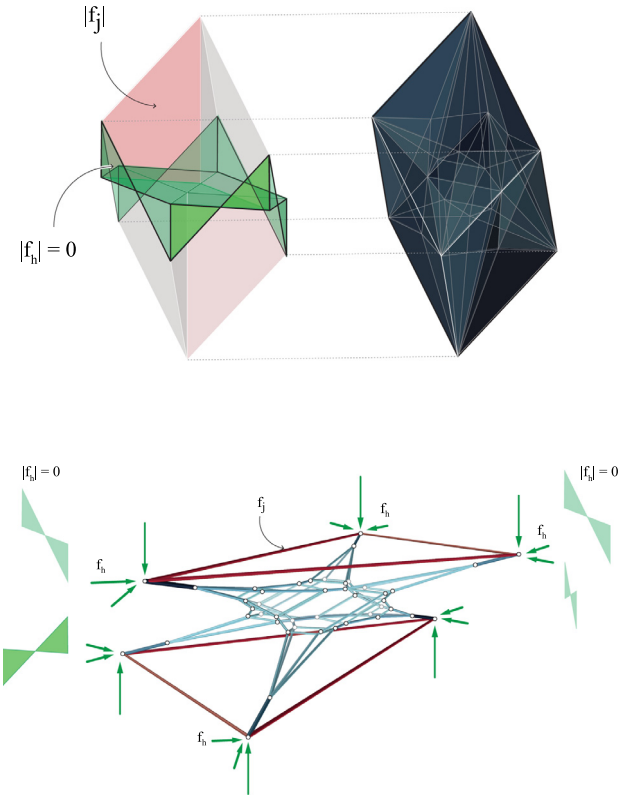

Fig. 18. The force and the form diagram of the built structure with the eliminated horizontal forces in the top and bottom of the structure.

Table 2
Computation time for the examples of the paper for further comparisons.

Input geometry	Number of faces	Running time [s]
Fig. 14	32	0.24
Fig. 15	72	0.13
Fig. 16	232	0.94

behind such problems is twofold. Firstly, if the number of fixed edges in a single face exceeds its GDoF, then the new geometry of the face with the target area cannot be calculated. Secondly, the calculation limitation might be due to some natural geometric restrictions of specific polygons. For instance, the area of a triangle cannot be switched to its opposite value by changing the length of its edges. Similarly, a rectangular face area cannot have a non-zero value if one of its edge lengths is zero.

6. Conclusion and discussion

This paper provides an algebraic formulation alongside algorithms and numerical methods, to geometrically control the areas of the faces of general polyhedrons of the reciprocal diagrams of 3D/polyhedral graphic statics. The presented methods bridge the gap between the previously developed algebraic methods for the construction of the reciprocal polyhedral diagrams and controlling the magnitude of internal and external forces by changing the areas of the faces. This method for the first time allows the user to manipulate both convex and complex faces and explore the compression and tension combined features in structural form-finding using 3DGS. Controlling the areas of complex faces has never been addressed in the literature prior to this research, as the previous approaches mainly dealt with convex polyhedrons. Thus, this research opens a new horizon to the understanding of the equilibrium of both tension and compression forces beyond the existing compression-only polyhedral funicular forms.

The paper explains the process of turning geometric constraints such as edge lengths and target face areas of the reciprocal polyhedral diagrams into algebraic formulations compatible with the previously developed method by Hablicsek et al. [43]. This research describes a quadratic formulation to compute the geometry of a face with a target area and provides a linear formulation to consider the edge lengths as constraints. Solving an equation system including both linear and quadratic equations is a highly complex task. The key idea in our proposed method is to reduce the number of unknowns in the (quadratic) equation system of a face using the Reduced Row Echelon (RREF) method. Computing the updated geometry of the polyhedral diagrams is achieved by Moore–Penrose Inverse (MPI) method. In this approach, multiple faces and edges can be selected as constraints, and the new geometry of the polyhedral system is computed in a sequential process.

The paper also describes the Constrained Geometric Degrees of Freedom (CGDoF) of the linearly constrained polyhedral systems and opens up an exploration of a wide variety of interesting geometries satisfying the initial equilibrium equations with selected edge lengths. The algorithms and numerical methods provide an interactive tool for the user to study and manipulate large-scale general polyhedral diagrams by assigning face areas and edge lengths.

Future work

The existing algorithm deals with each face at each step and adds the newly computed edge lengths to the initial constraints of the polyhedral system. As a consequence there is no control over the number of constraints generated in each step and the eventual number of constraints to compute the entire polyhedral group. Therefore, in certain cases, depending on the chosen edge by the user, or the geometric degrees of freedom of the entire system, the polyhedral computation becomes over-constrained. This property proposes an interesting problem for future research.

Another interesting future direction is the study of the different solutions of the same initial, constrained problem. As mentioned in Section 2.4, for a single face there can be two, significantly different polygons satisfying the linear and quadratic constraint equations. As a consequence, for a polyhedral system with n assigned face areas, there can be 2^n significantly different updated polyhedral systems that can also be explored in future research.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was funded by U.S. National Science Foundation (NSF) CAREER Award (NSF CAREER-1944691 CMMI-ECI).

References

[1] Rankine M. Principle of the equilibrium of polyhedral frames. Phil Mag 1864;27(180):92.
[2] Akbarzadeh M. 3D graphic statics using polyhedral reciprocal diagrams (PhD thesis), Zürich, Switzerland: ETH Zürich; 2016.
[3] Beghini A, Beghini LL, Schultz JA, Carrion J, Baker WF. Rankine's theorem for the design of cable structures. Struct Multidiscip Optim 2013.
[4] Maxwell JC. On reciprocal figures and diagrams of forces. Phil Mag J Ser 1864;4(27):250–61.
[5] Lee J. Computational design framework for 3D graphic statics (PhD thesis), Zurich: ETH Zurich, Department of Architecture; 2018.
[6] Lee J, Van Mele T, Block P. Disjointed force polyhedra. Comput Aided Des 2018;99:11–28.
[7] McRobie A. Maxwell and rankine reciprocal diagrams via Minkowski sums for 2D and 3D trusses under load. Int J Space Struct 2016;31:115–34.
[8] McRobie A. The geometry of structural equilibrium. R Soc Open Sci 2017;4(160759).
[9] McRobie A, Konstantatou M, Athanasopoulos G, Hannigan L. Graphic kinematics, visual virtual work and elastographics. R Soc Open Sci 2017;4(170202).
[10] Konstantatou M, D'Acunto P, McRobie A. Polarities in structural analysis and design: n-dimensional graphic statics and structural transformations. Int J Solids Struct 2018;152:272–93.
[11] Konstantatou M. Geometry-based structural analysis and design via discrete stress functions (PhD thesis), University of Cambridge; 2020.
[12] Stokes GG. Mathematical and physical papers. Cambridge: Cambridge University Press; 1905.
[13] Akbarzadeh M, Van Mele T, Block P. On the equilibrium of funicular polyhedral frames and convex polyhedral force diagrams. Comput Aided Des 2015;63:118–28.
[14] Ohlbrock PO, D'Acunto P, Jasienski JP, Fivet C. Vector-based 3D graphic statics (part III): Designing with combinatorial equilibrium modelling. In: Kawaguchi K, Ohsaki M, Takeuchi T, editors. Proceedings of the IASS annual symposium 2016 "Spatial structures in the 21st century". IASS; 2016.
[15] Cremona L. Graphical statics: Two treatises on the graphical calculus and reciprocal figures in graphical statics. Oxford: Clarendon Press; 1890.
[16] Wolff J. The classic: on the inner architecture of bones and its importance for bone growth. Clin Orthop Relat Res 2010;468(4):1056–65.
[17] Block P, Ochsendorf J. Thrust network analysis: a new methodology for three-dimensional equilibrium. J Int Assoc Shell Spat Struct 2007;48(3):167–73.
[18] Whiteley W, Ash PF, Bolker E, Crapo H. Convex polyhedra, Dirichlet tessellations, and spider webs. In: Senechal M, editor. Shaping space: Exploring polyhedra in nature, art, and the geometrical imagination. Springer New York; 2013.
[19] Block P. Thrust network analysis: Exploring three-dimensional equilibrium (PhD thesis), Cambridge, MA, USA: MIT; 2009.
[20] Rippmann M. Funicular shell design: Geometric approaches to form finding and fabrication of discrete funicular structures (PhD thesis), ETH Zurich; 2016.
[21] Akbarzadeh M, Bolhassani M, Nejur A, Yost JR, Byrnes C, Schneider J, Knaack U, Borg Costanzi C. The design of an ultra-transparent funicular glass structure. In: Structures congress 2019. Orlando, Florida; 2019.
[22] Bolhassani M, Akbarzadeh M, Mahnia M, Taherian R. On structural behavior of a funicular concrete polyhedral frame designed by 3D graphic statics. Structures 2018;14:56–68.
[23] Zalewski W, Allen E. Shaping structures: statics. New York: Wiley; 1998.
[24] Fivet C, Zastavni D. Constraint-based graphic statics: New paradigms of computer-aided structural equilibrium design. J Int Assoc Shell Spat Struct 2013;54(4):271–80.
[25] Fivet C, Zastavni D, Ochsendorf JA. The papers of maurice koechlin (1856-1946). In: 5th international congress on construction history. 2015.
[26] Heyman J. The stone skeleton: structural engineering of masonry architecture. Cambridge University Press; 1997.
[27] Bolhassani M, Ghomi AT, Nejur A, Furkan MO, Bartoli I, Akbarzadeh M. Structural behavior of a cast-in-place funicular polyhedral concrete: Applied 3D graphic statics. In: Proceedings of IASS annual symposia, Vol. 2018. International Association for Shell and Spatial Structures (IASS); 2018, p. 1–8.
[28] Heisel F, Lee J, Schlesier K, Rippmann M, Saeidi N, Javadian A, Nugroho AR, Van Mele T, Block P, Hebel DE. Design, cultivation and application of load-bearing mycelium components. Int J Sustain Energy Dev 2018;6(1):296–303.
[29] Gerhardt R, Kurrer K-E, Pichler G. The methods of graphical statics and their relation to the structural form. In: Proceedings of the first international congress on construction history, madrid. 2003; p. 997–1006.
[30] Van Mele T, Rippmann M, Lachauer L, Block P. Geometry-based understanding of structures. J Int Assoc Shell Spat Struct 2012;53(2).
[31] Van Mele T, Block P. Algebraic graph statics. Comput Aided Des 2014;53:104–16.
[32] Piker D. Kangaroo physics. 2013.
[33] Schek HJ. The force density method for form finding and computation of general networks. Comput Methods Appl Mech Engrg 1974;3(1):115–34.
[34] Adriaenssens S, Block P, Veenendaal D, Williams C, editors. Shell structures for architecture: Form finding and optimization. London: Taylor & Francis - Routledge; 2014.
[35] Gibson LJ, Ashby Michael Farries SGS, Robertson CI. The mechanics of two-dimensional cellular materials. Proc R Soc Lond 1982;A38225–42.
[36] Han SC, Lee JW, Kang K. A new type of low density material: Shellular. Adv Mater 2015;27(37):5506–11.
[37] Akbari M, Mirabolghasemi A, Akbarzadeh H, Akbarzadeh M. Geometry-based structural form-finding to design architected cellular solids. In: ACM symposium on computational fabrication, SCF'20. Virtual Event, USA: Association for Computing Machinery, ACM; 2020.
[38] Rippmann M, Lachauer L, Block P. Interactive vault design. Int J Space Struct 2012;27(4):219–30.
[39] Little JJ. An iterative method for reconstructing convex polyhedra from extended Gaussian images. In: Proceedings of the third AAAI conference on artificial intelligence. AAAI'83, AAAI Press; 1983, p. 247–50.
[40] Horn BKP. Extended Gaussian images. Proc IEEE 1984;72(12):1671–86.
[41] Moni S. A closed-form solution for the reconstruction of a convex polyhedron from its extended Gaussian image. In: [1990] proceedings. 10th international conference on pattern recognition, Vol. i. 1990; p. 223–6.
[42] Lee J, Van Mele T, Block P. Area-controlled construction of global force polyhedra. In: Proceedings of the IASS symposium 2017. Hamburg, 2017.
[43] Hablicsek M, Akbarzadeh M, Guo Y. Algebraic 3D graphic statics: Reciprocal constructions. Comput Aided Des 2019;108:30–41.
[44] Akbarzadeh M, Hablicsek M. Geometric degrees of freedom and non-conventional spatial structural forms. In: Impact: Design with all senses, design modelling symposium. Berlin, Germany, 2020.
[45] McRobie A. Rankine reciprocals with zero bars. 2017, Preprint.
[46] O'Rourke J. Convex polyhedra realizing given face areas. 2011, CoRR, abs/1101.0823.
[47] Penrose R. A generalized inverse for matrices. Math Proc Camb Phil Soc 1955;51(3):406–13.
[48] Moore EH. On the reciprocal of the general algebraic matrix. Bull Amer Math Soc 1920;26(9):385–96.
[49] Mitchell T, Baker W, McRobie A, Mazurek A. Mechanisms and states of self-stress of planar trusses using graphic statics, part I: The fundamental theorem of linear algebra and the Airy stress function. Int J Space Struct 2016;31(2–4):85–101.
[50] Pellegrino S, Calladine CR. Matrix analysis of statically and kinematically indeterminate frameworks. Int J Solids Struct 1986;22(4):409–28.