IASS 2020/21 SURREY 7
International Conference on Spatial Structures 2020/21

Proceedings of the IASS Annual Symposium 2020/21 and the 7th International Conference on Spatial Structures
Inspiring the Next Generation
23 – 27 August 2021, Guilford, UK
S.A. Behnejad, G.A.R. Parke and O.A. Samavati (eds.)

Web-based, Interactive Platform for Polyhedral graphic Statics

Hua CHAI[*,a], Masoud AKBARZADEH[a,b]

[*,a] Polyhedral Structures Laboratory, School of Design, University of Pennsylvania, Philadelphia, USA

[b]General Robotic, Automation, Sensing and Perception (GRASP) Lab, School of Engineering and Applied Science, University of Pennsylvania, PA, 19146, USA

Abstract

This paper introduces a web-based, interactive educational platform for the methods of 3D/polyhedral graphic statics (PGS). The research includes developing libraries using JavaScript, Three.js, and WebGL to facilitate the construction of the reciprocal diagrams based on procedural and algebraic methods independent from any other software. The framework's mathematical and computational algorithms allow the construction of global equilibrium for the systems, the changes in the magnitude of the internal forces by a simultaneous change in both form and force diagrams. It also visualizes the reciprocal and topological relationship between the elements of the diagrams. This instant open-source application and the visualization interface provide a more operative platform for students, educators, practitioners, and designers in an interactive environment. It not only allows them to understand the topological relationship between the diagrams but also provides a platform to explore and manipulate spatial structural forms and their equilibrium for design purposes. Users can explore and design innovative, funicular spatial structures by changing the boundary conditions and constraints through real-time manipulating both force distribution and geometric properties of the form. This online educational platform broadens the accessibility of the method without the need for an intermediate software and platform. Multiple examples will contribute to the education of geometric methods of static equilibrium in three dimensions.

Keywords: 3D graphic statics, form finding, web-based software, interactive structural design, Three.js

1. Introduction

Recently, geometry-based structural design methods, known as Graphic Statics, have been extended to 3D dimensions based on polygonal reciprocal diagrams and polyhedral reciprocal diagrams. The latter is based on a historical proposal by Rankine and Maxwell in Philosophical magazine, and is called 3D/polyhedral graphic statics and will be the subject of this research [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]. In polyhedral graphic statics, the equilibrium of the forces in a single node is represented by a closed *polyhedron* or a polyhedral *cell* with planar faces. 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 proven using the divergence theorem [12, 13, 7]. In some cases, a cell can have *complex faces* (self-intersecting), which have multiple enclosed regions. 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. 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.

1.1. 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 [14, 15]. 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. This geometric relationship can teach students what parameters control their design and how they can deliberately modify/optimize them to achieve specific design criteria.

1.2. Accessibility of the methods

Learning the methods of polyhedral graphic statics might be challenging to begin with because of the following reasons. First, it requires a necessary preparation to understand the reciprocal relationship between the polyhedral cells and the geometry of the funicular forms. Secondly, all the existing implementation of polyhedral graphic statics heavily rely on modeling software such as Rhino and its related plugins such as PolyFrame [16] and 3D Graphic Statics [17] or another computational framework (e.g., COMPAS [18]). Thus, it needs certain 3D modeling or programming skills which make it quite inaccessible for all users. Like many other new concepts and methodologies, practicing procedural methods and reviewing a series of examples can overcome this obstacle. However, the environment should be easily accessible.

1.3. Web-based, interactive implementations

The web-based environment is the most user-friendly platform that can be manipulated easily across different knowledge backgrounds. Web technologies have been leveraged extensively to facilitate interoperability in software solutions and application programming interfaces (APIs). The WebGL [19], as one of the libraries in JavaScript [20], has advanced rendering and interaction capabilities, access geometric information over the network, and integrates the flexible data input mode. The innovations in WebGL and JavaScript allow the integration of advanced 3D graphic directly into web pages without additional plug-ins. Users can run three-dimensional scenes smoothly on the browser and support multiple platforms. Although advanced 3D visualization is being used broadly for web applications in various fields and modern graphic hardware is becoming increasingly available, most of the three-dimensional display scenes are based on fixed models, which cannot be simulated and displayed based on interacting selected parameters.

1.4. Related works

One of the most successful implementations of the methods of graphic statics is the eQUILIBIRUM 2.0 [21] developed by Block Research Group (BRG) at ETH Zürich. This web-based platform provides a dynamic learning and teaching environment for structural design and geometric equilibrium of forces. The implementation mainly focuses on 2D graphic statics and uses GeoGebra [22] - a suite of dynamic mathematical software that handles geometry, algebra, calculus, probability statistics, data tables, graphs, calculations. GeoGebra is the core computing program that serves as the fundamental drawing tool to generate the line intersections and line segments to represent the relationship between force and form. This platform uses interactive 2D drawings to help designers and engineers with all skill levels to interact, understand, and utilize the methods in 2d. There is currently no educational platform for students to learn the principles of polyhedral graphic statics. Students and researchers will significantly benefit from an interactive platform with multiple examples elucidating the method and underlying relationship between the elements of the form and force diagrams.

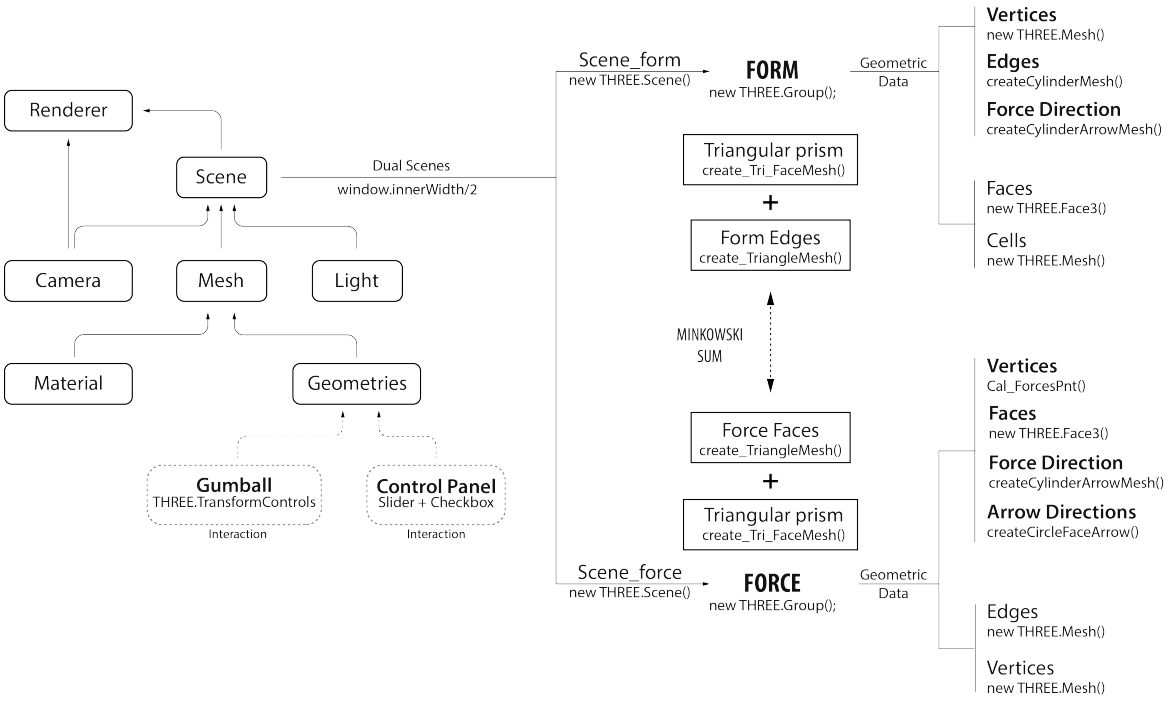

Figure 1: Three.js building architecture and interactive geometric data in both scenes

1.5. Paper contribution

This research develops an educational web-based platform for the methods of polyhedral graphic statics. This implementation utilizes JavaScript and Three.js [23] libraries to take advantage of the dynamic transformations between the form and force diagrams for educational purposes. Its immediate accessibility through the web enables users to visualize and create various structural systems with compression/tension-only and compression and tension combined elements. The possiblity to display changes in a real-time manner allows users to visualize how various parameters have changed the properties of the structural design and the equilibrium of forces.

2. Methodology

This section will introduce the principles of building a web-based platform based on Three.js, the algorithm generating geometric spatial forms, and the mechanisms by which users can interact with the generated structures. The method of interaction is through different UI combinations to change the data type of the form and force diagrams and the displacement of the vertices manipulated by the mouse click or dragging the vertex.

2.1. Overall structure

Figure 1 describes the 3D graphic statics scenes setup created by Three.js, including the structural components displayed in form diagram and polyhedron constructed in force diagram. Three.js as the primary programming tool builds the fundamental framework of the platform to host the geometric objects using the following steps: (1) input Three.js and related libraries packages to define the algorithms to enhance user interaction; (2) set the canvas tag in the HTML page to draw responsive scenes, and divide the current window into two parts intended for the form and force diagrams; this feature will also adapt to the

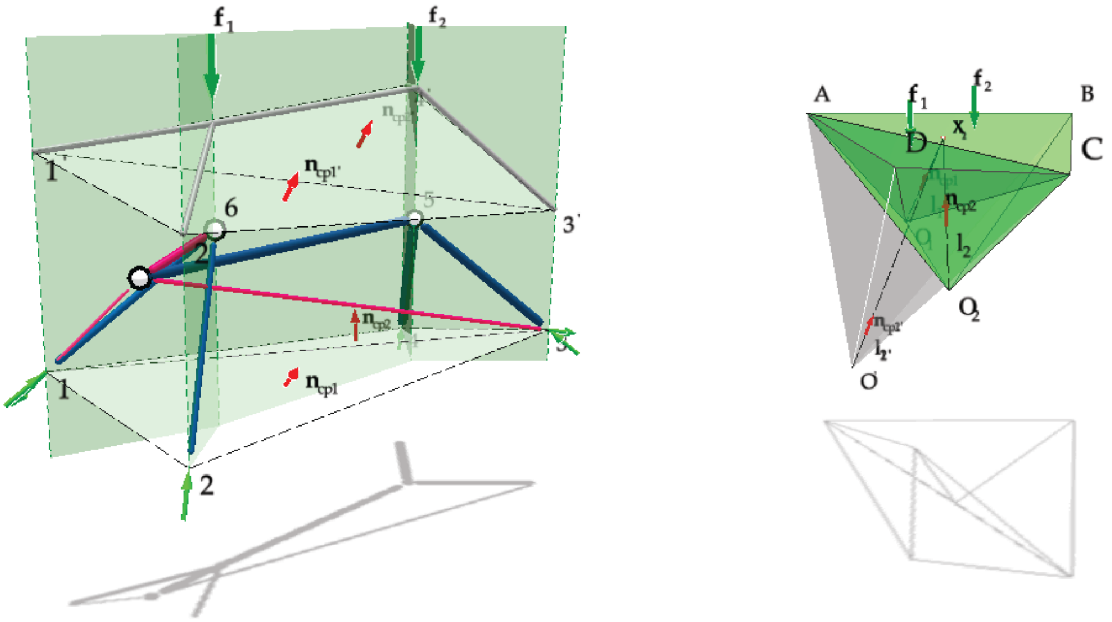

Figure 2: edges in form diagram corresponding to faces in force polyhedron.

window size across various devices; (3) initialize and load the environment factors inside the scene such as: (a) type, color, and intensity of lighting; (b) the closest and farthest viewing range of the camera, the camera tilt angle, aspect ratio, etc.; (c) generate rendering mode, set environment color and visibility of shadow features; (d) setting up relevant auxiliary user interaction functions, such as rotation, translation, scaling, sectioning, and event settings for geometries through mouse clicking event and panel slider manipulation. We will introduce it in the interaction section.

According to Rankine's principle of equilibrium [24] a closed force polyhedron represents the equilibrium of a spatial node of a structure [13]. This reciprocal relationship is at the crux of the methods of polyhedral graphic statics, and it is salient to visualize both form and force diagrams separately and simultaneously.

The surface area in the force diagram corresponds to the thickness of the edges in the form diagram. Visualizing this relationship as the force magnitude in the member of the designed structure can intuitively educate the relationship between the form and the force diagrams. In some examples a user can only move the support locations while in some others there is a control over the location of the applied load as well as the magnitude of the force in the force diagram. Any manipulation by the user results in a change in the force magnitude in the members of the funicular form which will be visualized in the interactive environment. This feature is designed to give an intuitive feedback to designers and structural engineers to realize the force flow in various configurations and have various design options without sophisticated calculations which may save time and effort in very early stages of the design (Figure 2).

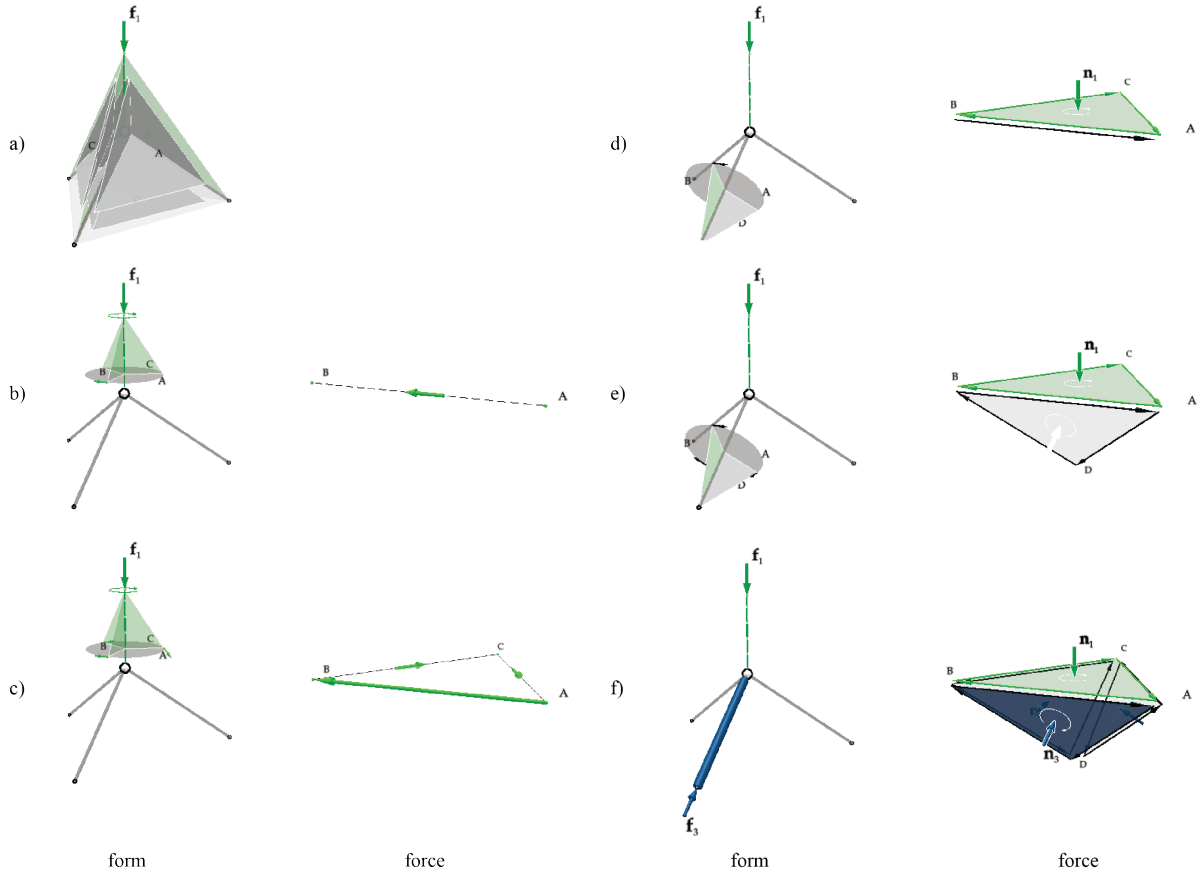

Figure 3: (a) Form diagram showing faces and cells; (b) divide space to A,B,C, and in force diagram draw line from point A to point B perpendicular to the normal of triangle; (c) calculate the intersection point C; (d) draw the rest edges and define the force direction; (e) repeat the previous steps to finish the corresponding triangle and f) define the compression force as $\mathbf{f}_3$.

2.2. Construction process

We will introduce how to build a 3D interactive platform in a web environment, the immediate environment setup with a user-operable visual interface as a container to integrate the content to carry all the back-end data and front-end objects. Algorithms are utilized to calculate spatial geometric linear intersections to generate basic triangles to form the final force polyhedron.

2.2.1. Three-dimensional environment setups

The scene corresponds to the real-world spatial environment and plays the role of a container; all objects, cameras, light sources need to be placed inside the window frame, which is the view-port of the scene. The next step is setting up the cameras. Three.js framework typically defines two main types of cameras - orthogonal and perspective cameras. The purpose is to map the model in three-dimensional space to the two-dimensional plane. The size of the geometry in the orthographic projection remains the same, and in perspective, the projection will appear large near and small far. Each scene has an independent camera, but the viewing angle is controlled simultaneously using the mouse left-click event. For properly presenting the models, the lighting is created to enhance the visual contact with all the relevant interactions. The light source corresponds to all kinds of light in the real world, such as ambient light, parallel light, point light, and hemispheric light. In order to make objects appear more realistic in the scene, it is necessary to simulate the effect of displaying objects in different environments. Lastly, render the

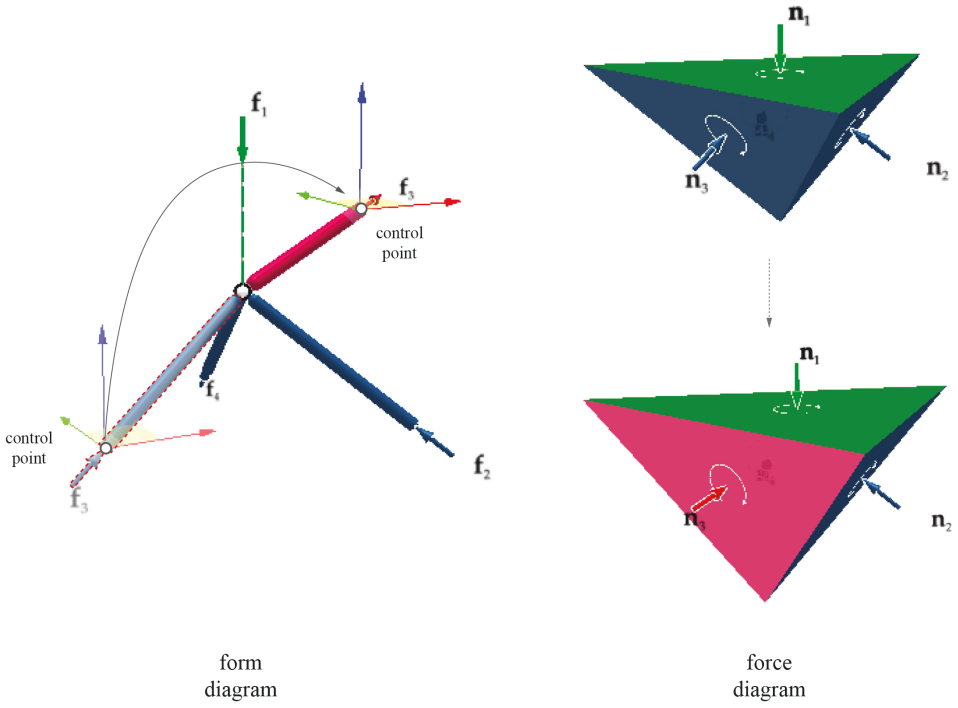

Figure 4: Gumball selection at vertices of form diagram (left) and corresponding force polyhedron showing area-weighted normals (right).

geometric model and material into accurate content by calling the rendering function to render all the elements and visually present them in a window frame.

2.2.2. Generate geometric model

All the examples are calculated on the geometric vector. It plays a significant role in Three.js programming because the display screen is a coordinate system, and the user's interaction with the geometry of the form and force is the result of the object's curvilinear motion in this coordinate system, and what describes these curvilinear motions are vectors. The use of vectors is a proper way to simulate the correspondence transform by data manipulation. The final force polyhedron can be obtained from the trial funicular [2]. The final form diagram can be drawn by spatial vector calculation in a given support locations. Figure 3 shows the steps starting from the apply load, followed by the right-hand rule to generate the corresponding vectors that construct force faces. The direction of the vector is equal to the normal vector of faces in the form diagram and intersects the line where the two vectors meet to create triangle ABC. The algorithm 1 is generated to construct force polyhedron geometry.

Since the length of the vector can be chosen to any value, the distance variation is the size of the force diagram. We will describe this in the next section on interacting with the existing model to obtain different designs.

Algorithm 1: Computing the intersecting $\mathbf{P}_3$

Input: $Line_{1_{dir}}, Point_1, Line_{2_{dir}}, Point_2$, start point and end point with vector directions
Output: $\mathbf{P}_{sect}$ the intersection of two lines.
Function $Lines_{sectPt}$ ($\overline{L}_{1_{dir}}$, $Point_1$,$\overline{L}_{2_{dir}}$, $Point_2$,) **:**

- $\overline{L}_{1_{dir}} : [newTHREE.Vector3()]$ # generate vector directions L_1
- $\overline{L}_{2_{dir}} : [newTHREE.Vector3()]$ # generate vector directions L_2
- **for** $x, y, z \in \overline{L}_{1_{dir}}$ **do**
  - $\overline{L}_{1_{dir}} : normalize$ # get norm vector
- **for** $x, y, z \in \overline{L}_{2_{dir}}$ **do**
  - $\overline{L}_{2_{dir}} : normalize$ # get norm vector
- $\overline{P_1P_2}_{vec} : [newTHREE.Vector3()]$ # generate vector P_1 to P_2
- **for** $x, y, z \in \overline{P_1P_2}_{vec}$ **do**
  - $\overline{P_1P_2}_{vec} : Point_2 - Point_1$ # vector direction
- $P_2P_3{}_{norm} = norm(cross(\overline{P_1P_2}_{vec}, \overline{L}_{1_{dir}}))$ # distance from P_2 to L_1
- $Point_3 = [newTHREE.Vector3()]$
- **for** $x, y, z \in \overline{Point}_3$ **do**
  - $Point_3 = Point_1 - (\overline{P_1P_2}_{vec} * \overline{L}_{1_{dir}}) * \overline{L}_{1_{dir}}$ # projection of $Point_3$
- $cos_\theta = Math.abs(\overline{L}_{1_{dir}} * \overline{L}_{2_{dir}})$
- $k_{pt} = [newTHREE.Vector3()]$
- **if** $cos_\theta < 0$ *# the angle between* $\overline{L}_{1_{dir}}\overline{L}_{2_{dir}}$ **then**
  - $k_{pt} = Point_3$
- **if** $cos_\theta > 0$ *# the angle between* $\overline{L}_{1_{dir}}\overline{L}_{2_{dir}}$ **then**
  - $tan_\theta = Math.sqrt((1 - Math.pow(cos_\theta, 2)))/cos_\theta$
  - $k_{pt_3} = P_2P_3{}_{norm}/tan_\theta$
  - $k_{1\,pt} = newTHREE.Vector3((\overline{Point}_3 + k_{pt_3} norm * \overline{L}_{1_{dir}}))$
  - $k_{2\,pt} = newTHREE.Vector3((\overline{Point}_3 - k_{pt_3} norm * \overline{L}_{1_{dir}}))$
  - $\overline{P_2K_1}_{vec} = newTHREE.Vector3((\overline{k_1}_{pt} - Point_2))$
  - $\overline{P_2K_2}_{vec} = newTHREE.Vector3((\overline{k_2}_{pt} - Point_2))$
  - $D_1 = norm(cross(\overline{P_2K_1}_{vec}, \overline{L}_{2_{dir}}))$
  - $D_2 = norm(cross(\overline{P_2K_2}_{vec}, \overline{L}_{2_{dir}}))$
  - **if** $D_1 < D_2$ *# the y coordinate of* $\mathbf{n}_f$ **then**
    - $k_{pt} = k_{1\,pt}$
  - **else**
    - $k_{pt} = k_{2\,pt}$
- **return** $\mathbf{P}_{sect}$

begin

- $\mathbf{L}_1 \longleftarrow$ # vector of p_1
- $\mathbf{L}_2 \longleftarrow$ # vector of p_2
- $P_2P_3{}_{norm} \longleftarrow \overline{P_1P_2}_{vec} \times \overline{L}_{1_{dir}}$ # the cross product of the first two edges
- $\mathbf{P}_{sect} \longleftarrow \mathbf{K}_{pt}$ # projection at one of the edge

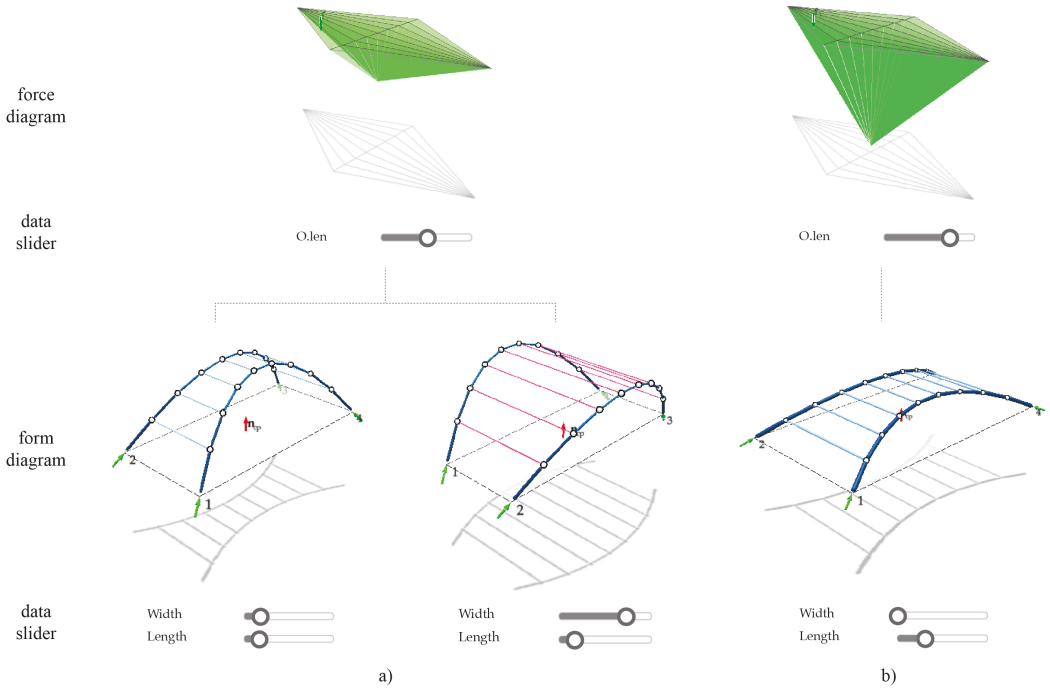

Figure 5: (a) Compression-only and tension/compression combined design from same force diagram; (b) data manipulation in force diagram resulting more flattened bridge.

2.3. User Interaction

Compared to most web-based 3d platforms that can only visualize objects from various camera perspectives, no other properties can be changed. In this research, we not only provide fundamental operations such as visual rotation and scaling, but most importantly, add user manipulation and interactivity, with a back-end algorithm that instantly displays the new structure obtained with each changed parameter.

2.3.1. Interpreted gumball function

The most common interaction on the browser is to detect whether a physical graphic is located below the mouse pointer by clicking on a 3D object—for example, the object picking operation. The selected object changes its color, visibility, other properties, and the object's animation is activated. The process of selecting is precisely the opposite of the process of drawing a physical graphic. Since the mouse click corresponds to the browser's screen coordinates (X, Y) and the Three.js canvas is a 3D scene, the 2D coordinates of the screen click need to be converted to Three.js 3D coordinates (X', Y') to determine the model of the clicked object. The method of coordinate pickup involves complex matrix operations, and the Three.js library provides the THREE. Raycaster object for mouse pickup, which is a traditional object space-oriented select method. The constructor is used as the new Raycaster (origin, direction, near, far): (1) the origin is the starting vector of the Raycast, which is the camera's location; (2) direction is the vector of the light projection (normalized). Utilizing the Raycaster function to determine the intersection detection of the models. The array is sorted by distance, and the closest to the viewpoint is the first one to be selected.

Figure 4 shows the mouse clicking event at the vertices of the form diagram. By selecting a movable point, the edges will move their position. Force direction will change accordingly and the force diagram

will show different colors to indicate tension moment (color in red) at $\mathbf{f}_3$. Related mouse event interaction algorithm 2 is showing here:

Algorithm 2: Creating mouse click event

Input: $ctrl_{pts}, selectobj_{name}$ # define interactive vertices
Output: $trfm_{ctrl}$ # adding gumball to selected object
Function $onMouseDown(event)$**:**
  $rayCaster.setFromCamera(mouse, camera)$
  $varintersects = rayCaster.intersectObjects(Ctrl_{pts})$# option to add more objs as group data
  if $event.button === 2$ *# detach gumball* **then**
    $trfm_{ctrl}.detach()$
  if $event.button === 0 \&\& intersects[0]$ *# add gumball* **then**
    $selectObj = intersects[0].object$ # save current select obj
    $console.log("selectobj.name = " + intersects[0].object.name.charAt(2))$
    $lastPosition = selectObj.position$ # save current select obj position
    $trfm_{ctrl}.attach(selectObj)$
  return $trfm_{ctrl}$

2.3.2. Panel controller

Nowadays, the front-end technologies for web development are usually implemented in JavaScript and CSS. Jquery [25] refers to the integration of these programming packages in the formation of a development framework. It can be leveraged for web front-end development in designing web pages, responding to user behavior, and using the framework to more easily implement features such as web animations and user interactions. One of the essential features to improve user experience is applying Jquery to achieve asynchronous calls. After the user submits a request to the site without waiting for the server to return a response, they can continue to operate the web page or continue to make requests. Then at the appropriate moment, user will get the returned results. With this technique, it is possible to respond requests without loading the whole contents.

In the existing examples, control panels are provided below the form and force diagrams. The displacement of the force diagram at a fixed point position is integrated with the changeable parameters, and the slider and click-box UI are associated with the back-end data. For manipulating the values on the slider, the corresponding geometry of the form or force will change accordingly. In the bridge example shown in Figure 5, by adjusting the horizontal distance of the support point, the structural members of the bridge will switch from compression only to a combination of tension and compression in real-time. Additionally, the curvature of the bridge will also change, with the vertex position farther from the applied load face by adjusting the position of the force polyhedron fixed point, and the bridge tends to be more flattened. If the slider is moved in the other direction, it will produce an arch bridge with significant curvature. The operation described above is an intuitive reflection of how the design can be selected and optimized by controlling the geometry position.

3. Conclusion and future work

This paper explained how to build a web-based 3D interactive platform for polyhedral graphic statics (Figure 6). The mathematical algorithm of spatial geometry is used to demonstrate the reciprocal principle of 3d graphic statics. Users can change the corresponding data to get different structural designs through various interactive methods with interpreted UI designs. Furthermore, this intuitive presentation

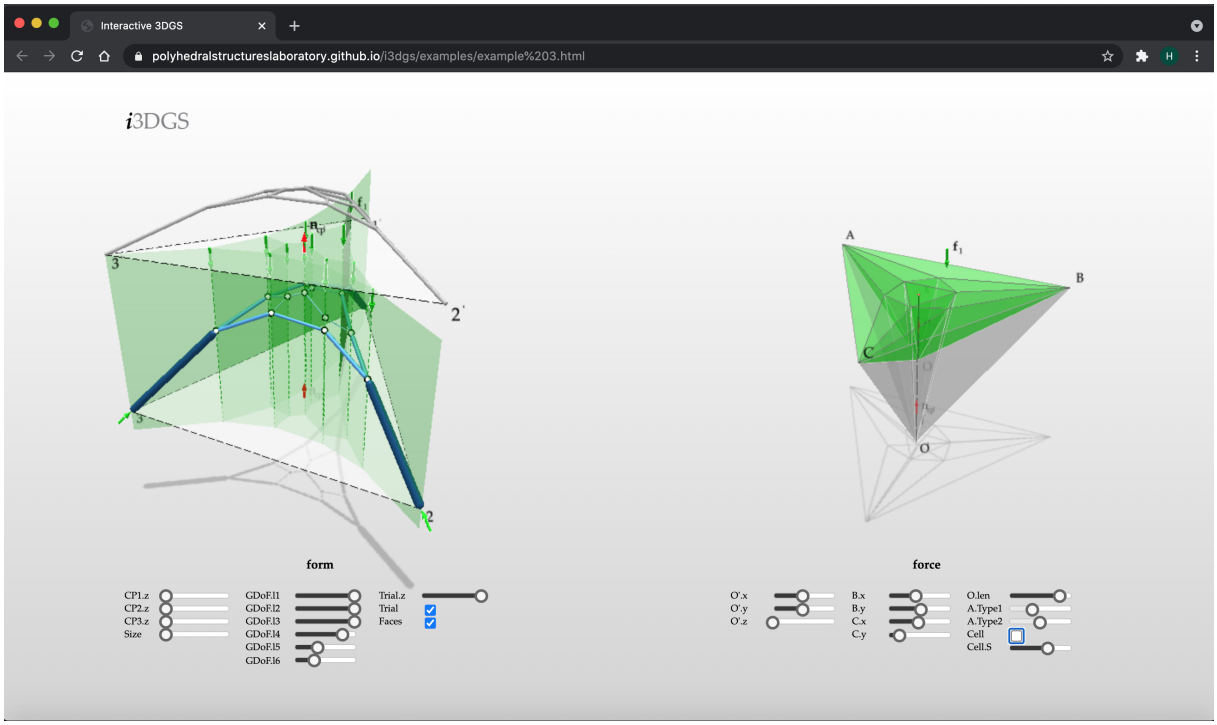

Figure 6: Live mode of the interactive Polyhedral Graphic Statics platform.

provides an accessible, convenient and faster response environment. In the next step, we will provide more possibilities for enhancing the online structural design. It is also essential to improve other platform features to demonstrate its educational nature for users better. For example, adding introduction texts to explain the details - when the user moves the mouse to the geometry, a prompt 2D layer will appear, illustrating more detailed content with highlighted colors. Eventually, the design can be exported directly to other formats and save the geometries.

4. Acknowledgment

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

References

[1] M. Rankine, "Principle of the equilibrium of polyhedral frames," *Philosophical Magazine*, vol. 27, no. 180, p. 92, 1864.

[2] M. Akbarzadeh, *3D graphic statics using reciprocal polyhedral diagrams*. PhD thesis, ETH Zurich, Zurich, Switzerland, 2016.

[3] A. Beghini, L. L. Beghini, J. A. Schultz, J. Carrion, and W. F. Baker, "Rankine's theorem for the design of cable structures," *Structural and Multidisciplinary Optimization*, 2013.

[4] J. Maxwell, "On reciprocal figures and diagrams of forces," *Philosophical Magazine Series 4*, vol. 27, no. 182, pp. 250–261, 1864.

[5] J. Lee, *Computational design framework for 3D Graphic Statics*. PhD thesis, ETH Zurich, Department of Architecture, Zurich, 2018.

[6] J. Lee, T. Van Mele, and P. Block, "Disjointed force polyhedra," *Computer-Aided Design*, vol. 99, pp. 11–28, 2018.

[7] A. McRobie, W. Baker, T. Mitchell, and M. Konstantatou, "Mechanisms and states of self-stress of planar trusses using graphic statics, part ii: Applications and extensions," *International Journal of Space Structures*, vol. 31, no. 2-4, pp. 102–111, 2016.

[8] A. McRobie, "The geometry of structural equilibrium," *Royal Society open science*, vol. 4, no. 3, p. 160759, 2017.

[9] A. McRobie, M. Konstantatou, G. Athanasopoulos, and L. Hannigan, "Graphic kinematics, visual virtual work and elastographics," *Royal Society open science*, vol. 4, no. 5, p. 170202, 2017.

[10] M. Konstantatou, P. D'Acunto, and A. McRobie, "Polarities in structural analysis and design: n-dimensional graphic statics and structural transformations," *International Journal of Solids and Structures*, vol. 152, pp. 272–293, 2018.

[11] M. Konstantatou, *Geometry-based structural analysis and design via discrete stress functions*. PhD thesis, University of Cambridge, 2020.

[12] G. G. Stokes, *Mathematical and physical papers*. Cambridge: Cambridge University Press, 1905.

[13] M. Akbarzadeh, T. Van Mele, and P. Block, "3D Graphic Statics: Geometric Construction of Global Equilibrium," in *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium*, 2015.

[14] R. Gerhardt, K.-E. Kurrer, and G. Pichler, "The methods of graphical statics and their relation to the structural form," in *Proceedings of the First International Congress on Construction History, Madrid*, pp. 997–1006, 2003.

[15] T. Van Mele, L. Lachauer, M. Rippmann, and P. Block, "Geometry-based understanding of structures," *Journal of the international association for shell and spatial structures*, vol. 53, no. 4, pp. 285–295, 2012.

[16] A. Nejur and M. Akbarzadeh, "Polyframe, efficient computation for 3d graphic statics," *Computer-Aided Design*, vol. 134, p. 103003, 2021.

[17] O. Graovac, "3d graphic statics." https://www.food4rhino.com/app/3d-graphic-statics, 2019.

[18] T. Van Mele, T. Méndez Echenagucia, and M. Rippmann, "Compas: A framework for computational research in architecture and structures.," 2017. https://compas.dev/.

[19] M. Foundation, "Webgl - programming language," 2011. https://www.khronos.org/webgl/.

[20] B. Eich, "Javascript - programming language," 1995. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources.

[21] T. Van Mele and P. Block, "equilibrium - an interactive learning platform for structural design." https://block.arch.ethz.ch/eq.

[22] M. Hohenwarter, "Geogbra - open source software," 2002. https://www.geogebra.org/about.

[23] R. Cabello, "Three.js - open source software," 2010. https://threejs.org/.

[24] W. Rankine, "Principle of the Equilibrium of Polyhedral Frames," *Philosophical Magazine Series 4*, vol. 27, no. 180, p. 92, 1864.

[25] J. Resig, B. Aaron, and J. Zaefferer, "Jquery - javascript library," 2006. https://jquery.com/.