

MODELLING OF VIRTUAL COMPRESSED STRUCTURES THROUGH PHYSICAL SIMULATION

P.Brivio¹, G.Femia¹, M.Macchi¹, M.Lo Prete², M.Tarini^{1,3}

¹Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, Varese, Italy - paolo.brivio@uninsubria.it

²DiAP - Dipartimento di Architettura e Pianificazione, Politecnico Di Milano, Milano, Italy

³Visual Computing Group, Istituto Scienza e Tecnologie dell'Informazione, C.N.R., Pisa, Italy

KEY WORDS: (according to ACM CCS): I.3.5 [Computer Graphics]: Physically based modeling I.3.8 [Computer Graphics]: Applications

ABSTRACT

This paper presents a simple specific software tool to aid architectural heuristic design of domes, coverings and other types of complex structures. The tool aims to support the architect during the initial phases of the project, when the structure form has yet to be defined, introducing a structural element very early into the morpho-genesis of the building shape (in contrast to traditional design practices, where the structural properties are taken into full consideration only much later in the design process).

Specifically, the tool takes a 3D surface as input, representing a first approximation of the intended shape of a dome or a similar architectural structure, and starts by re-tessellating it to meet user's need, according to a recipe selected in a small number of possibilities, reflecting different common architectural gridshell styles (e.g. with different orientations, connectivity values, with or without diagonal elements, etc). Alternatively, the application can import the gridshell structure verbatim, directly as defined by the connectivity of an input 3D mesh. In any case, at this point the 3D model represents the structure with a set of beams connecting junctions.

User defined custom constraints (reflecting physical ones) can be imposed over this resulting structure in a quick and intuitive way. Points on the surface can be constrained to never leave a specific position, line or plane.

A physical simulation is then run, with the objective of obtaining a surface that, while preserving to a certain degree the initial shape, and fulfilling the constraints, represents a valid compressed structure, i.e. one composed by sub-elements which are only subject to compression, without residual bending forces.

It is a well established fact, in architecture, that this property allows for elegant structures which can be realized with lighter and cheaper materials (as testified for example by the pioneering work of Spanish Architect Antoni Gaudì, or, more recently, of the Deutsch architect Frei Otto).

During or after the simulation, constraints can be made weaker or lifted, and a few simulation parameters can be reedited, in an interactive way: resulting changes in the final 3D shape of the structure are made visible on the screen. The user can thus easily explore through a set of viable solutions all sharing with the aforesaid structural property.

The final result is exported in standard 3D digital formats for further processing.

In summary the proposed standalone application represents a quick and focussed tool to address a specific task.

1 INTRODUCTION

Curved and self-portant shapes have an important role in architectural design since the studies on catenary systems of the Spanish architect Antoni Gaudì. In order to design buildings such as the chapel of the Colonia Guell and the arches of the Casa Mila, he resorted to catenaries in a preliminary study: the shape of the structures was determined by building a physical model in scale, composed by a net of strings and additional weights, hanging it with hooks to a ceiling, and letting it deform. When inverted (flipped on the vertical direction), the resulting structure acts in pure compression, ideally requiring no additional support to preserve its shape. In this paper, we refer to this characteristic as *compressibility*, and we call such a structure a *compressible structure*¹.

More recently, this approach was extended by the Ger-

man architect and structural engineer Frei Otto, when he devised the procedure he used to build the Multihalle at Mannheim. Firstly, he designed a scaled catenary model, which was then reproduced in place as a static skeleton of beams and junctions. Once completed, every beam has been let move freely around its junctions, so that the whole structure deformed under its own weight. During this phase, any residual internal force has been compensated and the final shape was then in equilibrium. Finally, the beam positions have been frozen and the whole structure has been covered, again ensuring compressibility of the result.

Similar results can be clearly obtained through computer physical simulations.

Earlier in the design process of a new structure, e.g. when the concepts are being considered, architects tend to focus more on factors (aesthetic, function, etc) other than structural ones. It is usually only later in the process that structural properties like compressibility are fully taken into account. Any virtual model must be carefully evaluated

¹Compressible structures are also commonly referred to as "funicular" and "compression-only" (or "tension-only" when referring to the model before vertical flipping).

and validated, before realizing any real-world structure, using specialized tools like those based on FEM (Ekkehard, 2004).

We propose a way to embed structural consideration in the architectural design process as early as the design of the very concept of the structure of interest. To achieve this, we resort to a quick and direct tool capable of visualizing a “compressible version” of a preliminary 3D structure taken as input (and to export it for later uses). Basic editing capabilities allow the user to navigate between alternative plans, all equally compressible, thus navigating through a space of compressible structures. This way, later stages (when structural characteristic must be enforced) will hopefully impose smaller changes in the final output.

During this preliminary process, accuracy is traded for speed and simplicity: at this stage the model is still conceptual, consisting of approximate shapes defined by few elements. Also, some approximation and heuristic are adopted in order to speed up the process.

To compute catenary-based surfaces, we adopt a particle-spring model taking inspiration from the computer graphics physical simulation techniques. Such systems efficiently simulate physical behavior with a sufficient level of accuracy, as testified by the numerous scopes where it has been successfully adopted: hair, human muscle, aquatic plants, and many other simulation (Nealen et al., 2006, Gibson and Mirtich, 1997). We show that these systems can be adapted to simulate closely the behavior of catenary curves and surfaces.

This way, we take advantage of a mature technology (developed for computer-games). Off-the-shelf physical simulator specialized hardware, like AGEIA PhysX accelerator, could be used for the task. With respect to the typical computer-game scenario, our task is simplified, because we are only interested in the final state of the simulation rather than in its step-by-step evolution (this allows for further optimization, see section 2).

We implemented a tool (as a stand-alone software) based on this approach to aid the early stages of designing architectural domes and covering.

1.1 Solution approach

Technically, our goal is to produce a compressible 3D structure which is similar to a given arbitrary one. The idea is to deform the source structure into a new one which resembles it but acts in pure compression.

Our tool allows an user imports (in form of a mesh see Sec. 5.1) the *source structure*, then runs a physical simulation to produce a correspondent one which is compressible.

The shape of the compressible structure produce in output is determined by:

- the shape of the source structure,

- the parameters of the simulation,
- the set of constraints imposed by the user.

Importantly, any surface produced in output by the system is guaranteed to be compressible (in the sense defined in Sec. 1 above). Manipulating any of the factors in the above list through a simple GUI (Sec. 5.2), the user navigates among a set of possible compressible structures. The factors can be changed even before the simulation is fully converged, allowing for a more interactive search among possible solutions.

Note that, depending on several choices, it is not always the case that the resulting compressible structure is close to the source one. For example, the Hausdorff distance between the two cannot be easily bounded a priori. However even surfaces which are far away to the source one can be useful as viable design alternatives to be considered.

1.2 Structure of this paper

First, in the next section, we show the particle-and-spring explicit solver which we will be using. In Sec. 3.3 we show how this system can be used to find the shape of a proper catenary curve embedded in a 2D plane. Then (Sec. 4) we extend the same concept to surfaces embedded in 3D space. Sec. 5 describes the interactive software tool that is used to make this approach adoptable by the final user in the early stage of dome-like structure design. In Sec. 6 we show a small gallery of results which were obtained with our tool and end with a brief discussion.

2 PARTICLE AND SPRINGS SYSTEMS

Particle-and-spring systems are an intuitive way for representing a deformable model physics (Nealen et al., 2006). For more than two decades they have been widely exploited in Computer Graphics. Due to their simplicity and adaptivity to many different contexts, they have been especially exploited for achieving realistic animations of human bodies, human skin, soft organic tissues, clothes and garments, etc.

A particle-and-spring system is composed by a net of particles connected by springs that evolve in time. Springs are linear weightless elements, elastic along their axial direction, which connect particles. Each spring is characterized at least by a *rest length* and a *stiffness coefficient*, which determine an axial reaction force whenever the spring is stressed or stretched, as stated by Hooke’s law. Both factors usually are kept constant during the simulation (however that is not our case, see later). In practice a spring generates a force that tends to restore the spring rest length.

Particles are punctiform concentrations of mass, which can in principle differ from particle to particle. The amount of mass is another constant in typical (again, not ours) particle-and-spring system. Each particle has a positions, a velocity and an acceleration, which evolve in time, which

vary as time elapses, driven by forces applied to it. There are three kind of forces: spring reactions, constraints and external forces. A constraint contrasts particle movement in some direction. It is used for example to prevent penetration with other objects (Terzopoulos and Waters, 1990), or to maintain constant volume inside the model (Lee et al., 1995), and so on. External forces are independent from the system, like gravity.

2.1 Resolution methods

To resolve a particle-and-spring system means to find a state where all forces acting on each mass is zeroed (or is voided by constraints), or, equivalently, when the potential energy is in a (possibly local) minimum.

Particle and spring systems can be solved with explicit or implicit solvers.

Implicit solvers tackle a system of partial differential equations (PDE) that describes the particles motion in function of time is integrated. PDE can be solved by means of *Runge-Kutta* methods, used to integrate the equations of motion.

In explicit solver, an iterative simulation determines the system state at the next time step starting from the current one. At every step, particles are updated in function of the step length. Sometimes, especially when subject to strong forces or long time steps, masses particles tend to oscillate. A damping coefficient can then be added to springs, helping the system be more stable. Larger time-steps make the system converge faster, especially when the current state far from the limit Explicit system can be made more stable adopting a *Leapfrog integrator*. As we are dealing with a second-order system, we calculate positions and displacements alternately, at alternate time points, so that they “leapfrog” over each other. This method is still only second-order accurate, but it is very stable. The simulation runs until it converges to a stable configuration.

2.2 Adopted particle and spring system

In our particle-and-spring model for compressible structures, the only external force is due to the uniform gravitational field, even though of opposite direction (as structures are compressed, not tended). Constraints will be added (see Sec. 5.2) to particles and represents physical structures (like pillars, etc) which force that particle to never leave the starting position, line or plane during the evolution of the system.

We adopt an explicit *Leapfrog integrator*. This a standard choice for many physical system like those used in computer game industry, and many solvers are available, including those running on specialized hardware or GPUs. However, in our case we are only interested in the final zero-energy state, rather than the evolution of the system from the initial position to convergence. For this reason, we can model out inertia from the system, meaning that the the evolution is computed without using velocities: forces

directly displace position, rather than producing accelerations which affect velocities (equivalently, the system can be said to have a “dumping coefficient” of 100%, which nullifies velocities after they have been applied to displace particle positions). This further simplifies the system avoiding useless oscillations around the solution and makes the system converge with fewer iterations.

3 LINEAR CATENARIES

As stated, we are interested in catenary-based surfaces because, once flipped, they have the compressibility property. Here, we consider simple catenaries curves (embedded in 2D plane) before treating catenary surfaces (embedded in 3D space) as their natural extension, see Sec 4.

The word *catenary* is derived from the Latin word *catena* meaning chain, a term introduced by Leibniz. A catenary is a 1D curve defined as the theoretical shape of a hanging flexible chain or cable when supported at its ends and acted upon by a uniform gravitational force (i.e. its own weight) and in equilibrium. Each segment of the chain pulls on the weight of the other segments resulting in a “U” shape that is similar in appearance to a parabola, though it is a different curve.

A catenary can be computed explicitly with a closed formula:

$$y = a \cosh\left(\frac{x}{a}\right) = \frac{a}{2} \left(e^{\frac{x}{a}} + e^{-\frac{x}{a}}\right)$$

which gives the height y of each point x up to a scaling factor a . The latter can be interpreted as the ratio between the horizontal component of the tension on the chain (a constant) and the weight of the chain per unit of length. Using the opposite of values of y , one could for example model a compressible arch.

3.1 Catenaries curves and particle-spring systems

Even though the closed formula (3) can be used for the 1D case, we will be forced to move to a numerical solution instead when we will extend from 1D catenary curves to 2D catenary surfaces: for illustration purposes, we show here how the 1D case could be addressed by means of a 1D particle and spring system (used to compute the shape of the curve).

The curve can be approximated by a piecewise linear curve, i.e. one composed by linear segments, each connected to the next by a punctiform *junction* (see figure 1).

Within this approximation, we want to model a catenary curve as the solution of particle and spring system. We assign a spring to each segment and a particle to each junction. Each mass is assigned an equal weight and initialized in equidistant positions along a line. Each spring is assigned the same stiffness coefficient and rest length. The resulting mass and spring system is run until convergence (see Fig. 1).

This intuitive approach, followed for example by (Axel and John, 2005), suffer from a few caveats, which we discuss in the following section.

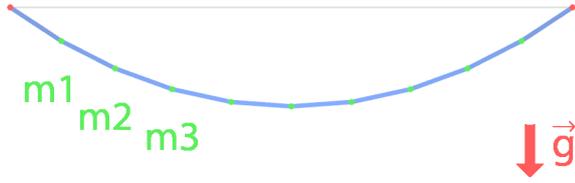


Figure 1: A catenary curve approximated by a particle and spring system. The shape is defined by the equilibrium position reached by the system.

3.2 Approximations and sources of error

Simulating catenaries through particle-spring systems introduces approximations and sources of error. Before extending to the case of surfaces, we analyze these errors and the ways to counter them, as the same will hold for the surface case.

The first error is of course linked to the use of finite elements to simulate a continuous string. The number of elements represent different trade-off between simulation complexity and accuracy. Reasonably fast systems composed by relatively few elements will be acceptable, since the tool is intended for preliminary concept exploration. It should also be noted that, at least in some cases, dividing the structure into finite linear elements could even be a faithful representation of the intended physical object rather than a modelling artifact.

Another problem is that we are assigning a weight to particles only, thus assuming the springs to be weightless. In reality, it is the beams, not the junctions between them, who have the most mass. Even if the weight will be applied in the simulation over a discrete set of points (rather than in a continuum), we can still make sure that the total weight and its distribution reflect those of the real world object. We achieve this by assigning to each particle a mass which is the halved sum of the springs attached to that particle.

The most serious source of error, however, is inherent in our choice to use deformable elements (spring) to simulate rigid elements (the beams). The difference is substantial: using uniform masses in particles, the physical system disposes the curve to shape a parabola rather than a catenary curve. The difference between the two curves (ideal catenary and actual parabola) increases the less stiff springs are. So the discrepancy can be minimized using stiffer springs, i.e. ones producing return forces which are very large (compared the only other force involved: gravitational force). That is the approach used for example in (Axel and John, 2005), where the difference is actually measured by comparing the finite element system with the closed formula (3). However, large stiffness coefficients, on one hand, poses problem with stability resolution systems, while on the other, they can only reduce, rather than fix, the inherent problem. We propose a different solution to address these shortcomings.

3.3 Mass redistribution

Our solution is simple and it is based on the observation that the discrepancy between extensible springs and the rigid beams they represent boils down, in the final state of the system, to the uneven distribution of mass along the curve.

When the spring is subject to forces at its extremes it unavoidably elongates and the two masses are moved apart, regardless of stiffness coefficient (if they did not elongate, they would produce no force). The problem however is not that the in final state linear elements are longer than the source state: these elements are just correctly modeling physical (incompressible) elements that are as long as the simulated spring in the *final* state of the simulation. Rather, elongation of springs results in an artifact smaller concentration of mass in that part of curve, because the mass is only represented at particles. This can be easily countered by dynamically rebalancing the masses assigned to each particle after each iteration of the particle-and-spring system.

Specifically, a parameter w_L of our system describes the weight per linear length of the string (kg/m). During system evolution, the current length l of each spring determines its mass $w_L \cdot l$ for that iteration. The distribution of beam masses among particles described above (Sec.) is repeated before each iteration using the beam mass of that iteration.

This approach forces us to use an explicit system, but luckily we can now afford smaller stiffness constants (because, up to discretization errors, we always get a catenary shape regardless of stiffness) so that is a viable solution.

Note that overall mass of the structure is not a constant of the described system: as soon the simulation dictates that beams elongate, they weight also increases (but convergence can be easily assured). We will discuss later some implications, in Sec. 4.1.

4 CATENARIES EXTENDED TO SURFACES

There is no univocal way to extend the concept of catenary curves to catenary surfaces. Even more importantly for our purposes, there is no known closed formula to describe a catenary surface. However, the approximation we proposed in Sec. 3.1 of catenaries as minimum energy state of a particle-and-spring system can be easily extended to surfaces, because such systems can be embedded in three (or any number of) dimensions. The final result will still produce a compressible surface once flipped upside down.

Our catenary surfaces are found as convergence (minimum energy) states of particle-and-springs system build placing a spring over each edge of a two-manifold polygonal mesh. Regular quad meshes or regular triangle meshes can be used. Again, each in the system represents a beam in the final structure, and each particle a joint between two beams. In this case also, final beams lengths are not known

a priori, as the system computes them evolving the initial state. However, the length of a beam in the source configuration determines directly the rest length of the corresponding string, therefore the ultimate length will tend to be similar to it.

4.1 Redistribution of masses in catenary surfaces

One difference with respect to the one dimensional case is that now we can now impose not only a weight per length unit to each linear beam, but also a weight per area unit to each sub-piece of the surface (representing the covering). The system is parameterized with the parameter w_A , the weight per unit of surface of the covering (kg/m^2). This approximately accounts for the weight of the physical surface elements covering the beam lattice, but also, when appropriate, for other unaccounted elements like effect of precipitations etc. The system is enriched with surface polygonal elements having springs as their sides. During system evolution, the area a for each element is computed determining its mass $w_A \cdot a$. That quantity is subdivided among particles in parts proportional to the solid angles of the corresponding corner (effectively, this assign to each particle the mass corresponding to the covering area of the Voronoi cell around that particle).

Changing the values of w_A and w_L the system will produce different shapes, allowing an user to explore several possibilities (all equally compressible).

Unlike what happens real world, the mass of the structure does change over time as our physical simulation progresses. What is important, however, is that the final minimum energy (zero forces) position will consist of a correct catenary curve of some length and some weight, even if the both quantities are different from the one specified in the initial state; what is kept constant is, rather, the mass concentration per unit of length (for beams) and of area (for coverings).

Since the initial state only serves as a way for the user to control the final result, it is acceptable that the system returns a structure which does not strictly maintains the lengths of its element nor, consequently, its overall mass. The important thing is that the produced structures are compressible. As long as the difference is small, the input shape can be effectively used as a way to drive the system.

4.2 Update of rest lengths

However, it is sometime desirable to enforce a stricter similarity between the input shape and the output compressible shape. The rationale is that this makes for a more controllable way for the user to select which compressible shape to get.

To achieve a better preservation of beam lengths from the input shape to the final result, we adopt an heuristic: first, the system is made to converge just as described above, that is, using initial distances between particles as rest length of the corresponding spring. Let l_i be this rest length for

the spring number i , i.e. its length in the input shape, and let l'_i be the length of the spring at the convergence state.

After convergence we then reassign to each spring a new rest length l''_i by resolving:

$$l'_i : l_i = l''_i : l'_i \quad (1)$$

This defines a new system which is in turn simulated until convergence.

In words, (1) means assuming that, in the new system, each given spring will be eventually elongated by the same percentage of its rest length as in the old system. Under this assumption, we assign a rest length so that the elongated final length of the spring will be the desired one. Since the assumption does not hold in general, the final lengths in the new system will not necessarily match the initial lengths defined in the input state (even if they usually get nearer).

This process can be iterated (wait for the system to converge, then update rest lengths, and repeat) but this is not guaranteed to converge. Instead, in our system the user can manually trigger the process any number of times, obtaining a new surface every time. In other words, this phase is provided as an additional tool for the user to navigate among compressible 3D shapes. Triggering this process means, intuitively, asking the system for a surface which is more similar to the one provided in input (if such a surface exists with is also compressible).

5 AN INTERACTIVE TOOL FOR PHYSICAL BASED MODELLING

The system discussed in Sec. 4 is made available to end user by means of a interactive stand-alone application, which loads, computes, displays, and saves catenary surfaces.

5.1 Import and export

In order to be embedded in the typical pipeline of architectural design, shapes can be imported into the system by loading a 3D two-manifold mesh into the system, which will be used as a starting state for the physical system.

If the connectivity of the mesh correspond to a desiderata of the architect designer in term of tessellation of beam and junctions (their number, approximate length, connectivity etc), then these factors are inherited from the input model (a spring is created for to each mesh edge, and a particle for each mesh vertex, as natural). Conversely, the input mesh can be used as a mean to define a general shape of the dome into the system. In this case, the surface is regularly resampled with a user-defined step (using the U-V parametrization of the input mesh) and a custom connectivity is added selecting it among a set of possibilities (quads, splitted quads, and so on, see Fig. 2). Additionally, a flat surface of a given size can be used as a starting position.

Likewise, two-manyfold polygonal meshes (standard formats) are also used as a mean to export the resulting compressible surface so that it can be embedded in the design pipeline and further processed.

5.2 Graphical user Interface

The Graphical User Interface (GUI) of the system allows the user to set the various system parameters described in this paper, like the values for w_L and w_A (see Sec. 3.3 and 4.1), the default stiffness constant and so on, by means of buttons, sliders, menus and so on.

It also allows the user to impose or remove position constraints associated to particles. A particle can be constrained to lie its specific position, on a vertical plane, or on a vertical line. The GUI provides tools to quickly assign constraints manually to particles or group of particles, as well as a small set of ready-made recipes to automatically determine constraints for every particle in one go (for example, constraining every corner particle in position, and every border particle to an appropriate vertical plane). Current constraints are shown to the user through simple 3D widgets.

The tool also embeds some basic tool to edit the original shape (either imported or resampled), like lowering or rising specific junctions in the source position; more complex edits are left to the CAD application producing the mesh before importation into the system (a more flexible solution would be to provide the functionalities of our tool as a plug-in for such CAD system).

Finally, the user is provided at all times with a rotatable 3D view to show the current shape of the dome as resulting from the simulation. The model is visualized as a lattice of beams or as the covering of the structure. Shadows and a plant view are used as visual aids to make the rendered shape clearer to the viewer. Optionally, the stress computed for each beam can be measured to be reported or used to color-code the elements (e.g. see Fig 3 and 4).

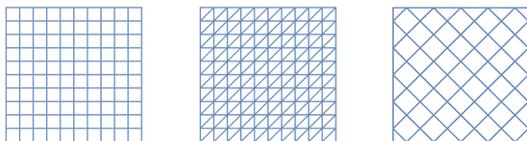


Figure 2: Examples of default conductivities which can be imposed upon resampling an input mesh.

6 RESULTS AND DISCUSSION

6.1 Result examples

We show here a few examples of compressible structures designed with our tool. To display the surfaces we use the same rendering which is used in our application.

6.2 Conclusion

We proposed a system to help architects embed structural considerations earlier during the process of design of domes and similar architectures. This is not intended to exempt from the need of careful validation of structural constraints later in the computer aided design process (as our system does not and cannot take in account parameters like accurate materials properties, weather factors, finite elements),

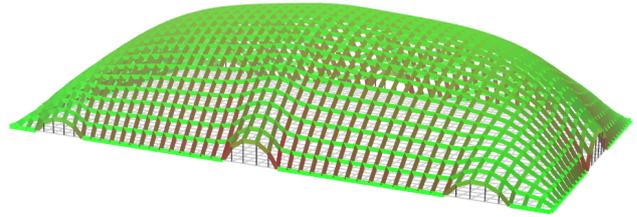


Figure 3: A completely compressible (self-portant) tennis dome realized with few pillars/portant-walls. Other examples on the next page.

but lets architect quickly explore a wide variety of generally sound solution. In this, the system is the computer analogue of physical scaled down systems that can be set up with similar purposes.

The many degrees of freedom that the system provides (including initial shape, parameters of the simulation, constraints) let the user interactively navigate among a space of viable solution.

As a physical system to compute compressible surfaces, we propose a simple variation of particle and spring system. It guarantees that (within discrimination limit) the obtained surface have the desired characteristic of compressibility (and so they represent the shape of compression-only, or self-portant, structures). Technique to parallelize particle-and-spring systems, like GPU based ones, could be easily adopted in our adapted system.

The resulting stand alone tool is being tested and used for its intended purposes.

REFERENCES

- Axel, K. and John, O., 2005. Particle-spring systems for structural form finding. *Journal of the International Association for Shell and Spatial Structures* 46(2), pp. 77–84.
- Ekkehard, R., 2004. Shape finding of concrete shell roofs. *Journal of the International Association for Shell and Spatial Structures* 45(1), pp. 29–39.
- Gibson, S. F. F. and Mirtich, B., 1997. A survey of deformable modeling in computer graphics. Technical report, Mitsubishi Electric Research Laboratories.
- Lee, Y., Terzopoulos, D. and Walters, K., 1995. Realistic modeling for facial animation. In: *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, pp. 55–62.
- Nealen, A., Mueller, M., Keiser, R., Boxerman, E. and Carlson, M., 2006. Physically based deformable models in computer graphics. *Computer Graphics Forum* 25(4), pp. 809–836.
- Terzopoulos, D. and Waters, K., 1990. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation* 1(2), pp. 73–80.

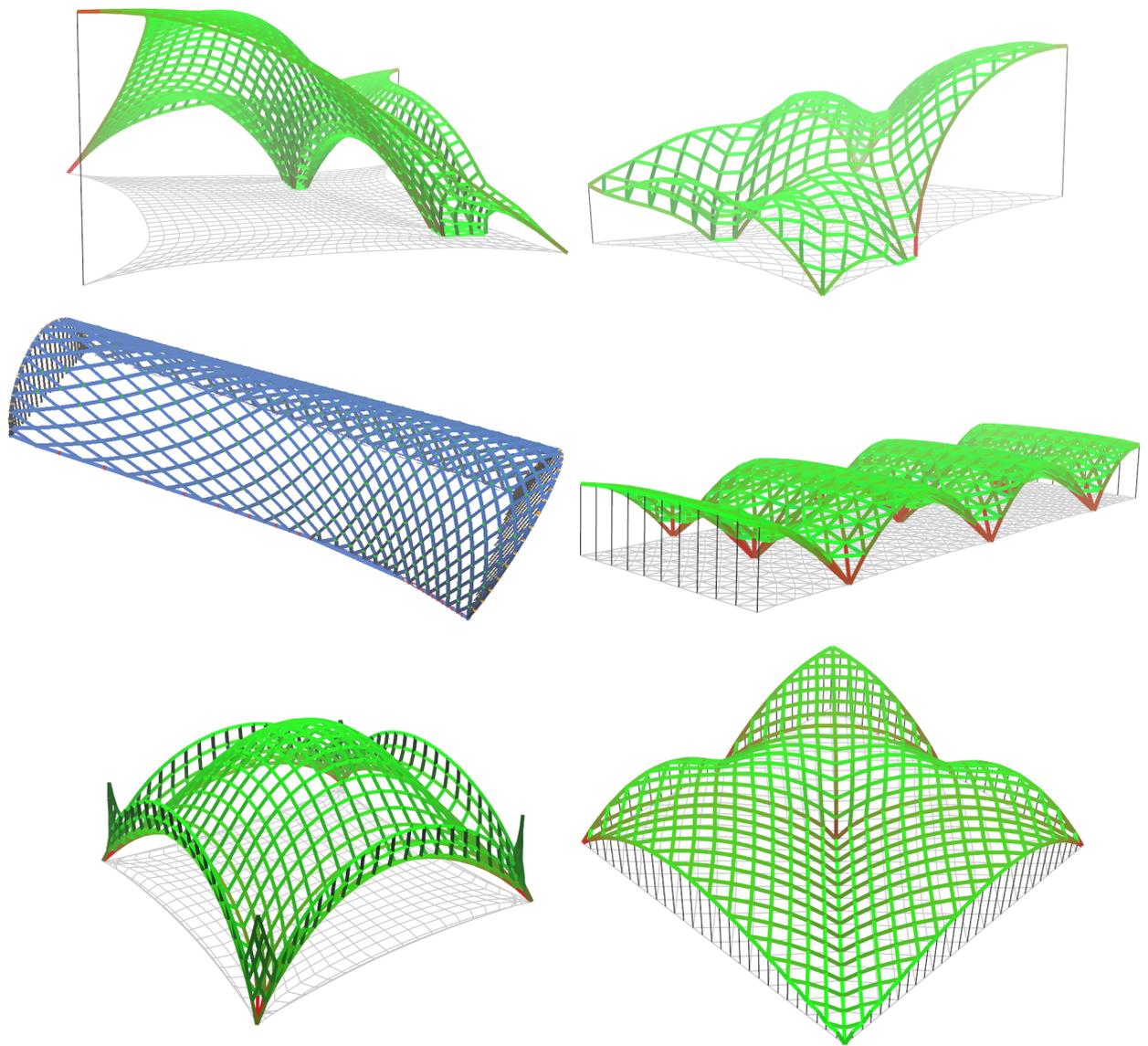


Figure 4: Example of various compressible (self-protant) structures designed with our tool.