EUROGRAPHICS 2012 / P. Cignoni, T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Rationalization of Triangle-Based Point-Folding Structures

Henrik Zimmer Marcel Campen David Bommes Leif Kobbelt

RWTH Aachen University

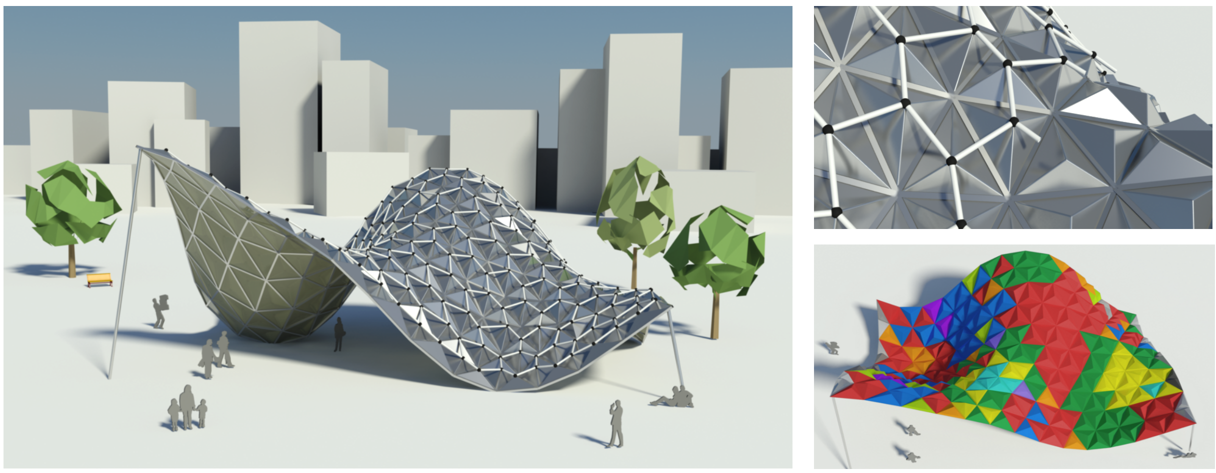


Figure 1: *Point-folding structure built from pyramidal elements according to a triangulated free-form base surface. Although almost all base triangles are non-similar, thanks to rationalization only 21 mold dies are necessary to produce the 270 elements.*

Abstract

In mechanical engineering and architecture, structural elements with low material consumption and high load-bearing capabilities are essential for light-weight and even self-supporting constructions. This paper deals with so called point-folding elements – non-planar, pyramidal panels, usually formed from thin metal sheets, which exploit the increased structural capabilities emerging from folds or creases. Given a triangulated free-form surface, a corresponding point-folding structure is a collection of pyramidal elements basing on the triangles. User-specified or material-induced geometric constraints often imply that each individual folding element has a different shape, leading to immense fabrication costs. We present a rationalization method for such structures which respects the prescribed aesthetic and production constraints and finds a minimal set of molds for the production process, leading to drastically reduced costs. For each base triangle we compute and parametrize the range of feasible folding elements that satisfy the given constraints within the allowed tolerances. Then we pose the rationalization task as a geometric intersection problem, which we solve so as to maximize the re-use of mold dies. Major challenges arise from the high precision requirements and the non-trivial parametrization of the search space. We evaluate our method on a number of practical examples where we achieve rationalization gains of more than 90%.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems

1. Introduction

The *rationalization* of complex free-form designs currently receives increasing attention in Architectural Geometry and Computer Graphics. Before being fabricated, free-form shapes are usually tessellated into sets of (discrete) so-called panels. Here the number and complexity of the panels directly influence the production cost of the design – if all elements require unique individual molds, production costs can quickly become prohibitive. The goal of rationalization is to

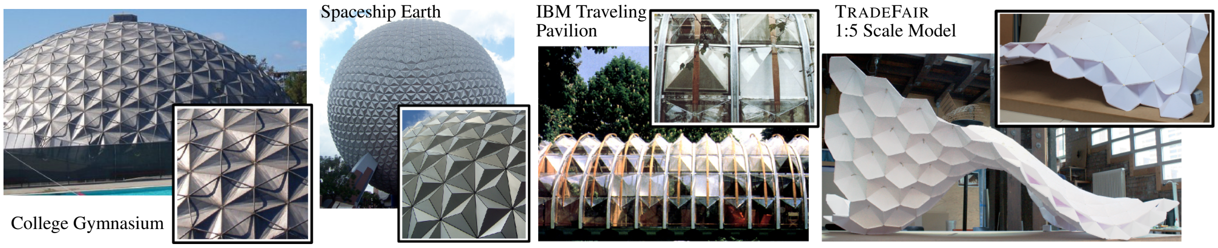


Figure 2: *Examples of existing folding structures. Left to right: Two* Fuller-*inspired domes, one with connected apexes for increased structural capability. The* IBM traveling pavilion *by* Renzo Piano *with quad-based folding elements connected by beams. A 1:5 scale hexagon-based, two-layered folding structure prototype [SSD] based on the* TRADEFAIR *model.*

find a smaller set of representative panels such that one optimized panel or production mold can be re-used as often as possible. This can be achieved by allowing small changes to the original shape and/or by exploiting adherent construction tolerances or structural degrees of freedom. Rationalization is a complex process usually involving highly non-linear optimization methods previously often deemed impracticable, but becomes increasingly interesting as the geometric understanding of such problems as well as computational capacities increase. This led to several highly reputable rationalization approaches which were recently presented in the computer graphics community, e.g. [FLHCO10, SS10, EKS*10].

Our rationalization deals with so-called *folding elements*. In the most general case, a folding element can be thought of as any non-planar panel with creases, and a *folding structure* is an assembly of adjoining folding elements. Just like a simple crease can transform a sheet of paper from a fluttering to a much stiffer state, folding or creasing of, e.g., thin metal sheets can be used to create elements of highly increased inherent stiffness without adding mass [SDG10]. Hence, they are appealing building blocks in the construction of light-weight, self-supporting structures without the need for heavy beams or additional support structures [Tra09].

More specifically, we consider the rationalization of structures with pyramidal elements, so-called *point-folding structures*, with triangular bases. Figure 1 shows an architectural design example. Figure 2 shows existing real-world examples of such structures (some with quadrilateral and hexagonal pyramid elements). Due to the triangular bases, such a point-folding structure can very naturally be described by a triangle mesh (plus pyramid apex positions). The potential benefit of rationalization of such structures was already pointed out by Herkrath and Trautz [HT11].

1.1. Contributions

For our rationalization task, the input base mesh is assumed to be purposely designed and shall not be altered. Besides respecting the designers intent, this opportunely serves the tractability of the problem. Hence, we propose to directly exploit other problem-inherent structural degrees of freedom and tolerances (subject to various hard constraints) to define a novel approach that completely respects the input.

Considering the constrained degrees of freedom, the range of feasible folding elements per input triangle is determined and parameterized. The rationalization task is then posed as a greedy optimization problem on top of a geometric intersection problem which we solve using a carefully designed, easily parallelizable algorithm. Results are guaranteed to respect user-specified constraints, which might be due to productional, constructional, or aesthetic requirements.

What sets our approach aside from many other rationalization methods is the way we deal with given constraints: no soft-constraints or continuous minimization is used – we consider the given constraints and tolerances as absolute and each solution fulfills them exactly rather than approximately, as it is often required in real-world engineering applications.

1.2. Related Work

Light-Weight Structures. Light-weight structure design and research was pioneered by Buckminster Fuller in the 1920s [Bal97]. One of the first commercial results was the Kaiser dome on Hawaii, that was made up from triangular pyramid elements with rod-connected apexes [Gil61]. An easy to deploy folding-based structure in later years was the *IBM Traveling Pavilion* – a half-cylindric structure covered by quad-based pyramidal elements [Buc00].

Rationalization. Note that the above mentioned constructions only used simple geometries such as cylinders and spheres. General (tessellated) free-form surfaces consisting of *all-different* elements would have contradicted Fuller's idea of efficient designs and affordable housing. With the availability of today's computers, rationalization of complex architectural shapes is becoming increasingly feasible and emerged as an active research topic. Recent high-profile work includes the rationalization of surfaces based on triangular panels [SS10], quad panels [FLHCO10], strips of single-curved panels [PSB*08], as well as more general curved panels [EKS*10, EDS*10]. In the latter work the possibility of producing several panels of the same shape but then cutting them to different sizes and forms is exploited – a possibility which we also make use of in order to allow the same pyramid shape to fit onto multiple triangles. The core

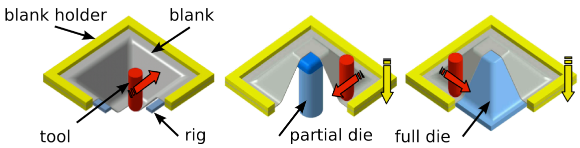
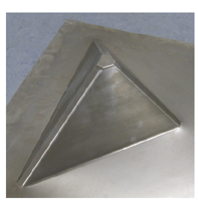


Figure 3: Incremental Sheet Forming *(ISF) can be performed without a die, with a partial die and with a full die. The sheet is formed by a robot-controlled tool incrementally pressing down the blank. (Images from [IMF])*

of our folding element rationalization is conceptually similar to the plane rationalization used by Décoret et al. [DDSD03].

Origami and Foldings. Folding elements and structures can be seen as a utilization and generalization of classical origami – the difference being that folding elements not necessarily adhere to taboos of traditional origami like cutting or gluing/welding. The simulation and synthesis of classical origami folds is an active area of research (e.g. Tachi [Tac09, Tac06] and Kilian et al. [KFC*08]). But also in the areas of architecture [Tra09, TK09] and structural engineering [SDG10, BW10] the stiffening property of folds is actively leveraged as it enables high load-bearing capabilities in light-weight materials such as thin-sheet metal. Recently, self-supporting rigid and kinematic folding structures were realized in full-scale as described by Buri and Weinand [BW10] and Künstler and Trautz [KT11].

Production. Besides the traditional possibility of cutting, folding, and welding sheets to creased elements, we briefly mention another promising production technique for creating polygon-based folding elements which can also directly profit from our rationalization technique. In *Incremental Sheet Forming* (ISF) [JMH*05] metal sheets are shaped by a robot-arm, incrementally pressing down points of a blank sheet with a tool, either without a die, with a partial die, or with a full die (cf. Figure 3) – in the order of increasing



accuracy of the resulting shape, but decreasing production flexibility. A resulting triangular folding element is depicted in the inset figure. Being cost- and time-efficient for low volume production, the costs for ISF increase significantly for large volume production of unique elements, especially when accuracy demands a different die for each individual element. Rationalizing the number of different parts can reduce this overhead drastically. Trautz and Herkrath [TH09] present a first approach to point-folding structures and ISF.

2. Point-Folding Structures and Elements

In the following, after introducing the definitions and notations used throughout the paper, we present the requirements that are typically posed on folding structures and state the actual rationalization problem setting we deal with.

2.1. Definitions and Notation

Let $\mathcal{M}$ be the input triangle mesh with n_F faces T_j and n_V vertices p_i. When considering a triangle $T \in \mathcal{M}$, we enumerate its incident vertices p_0, p_1, p_2. A folding element (or pyramid) P^a of a face T is completely defined by its apex a (the pyramid base being formed by the vertices p_i of T). The *crease vectors* $\mathbf{u}_i^a = p_i - a$, $i \in 0,1,2$, connect the apex to base corners and the three sides of the pyramid have normals $\mathbf{m}_i^a$. See Figure 4 for an illustration.

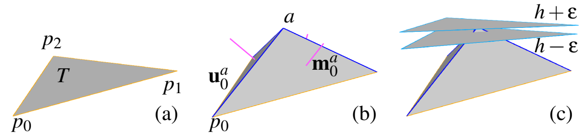


Figure 4: *(a) Triangle T. (b) A folding element (pyramid) P on T, with apex a, side normals $\mathbf{m}_i^a$ (pink) and crease vectors $\mathbf{u}_i^a$ (blue). (c) Simplest variant of a validity range for pyramid height h over T with tolerance ε.*

2.2. Typical Requirements

Extensive discussions with architects and construction engineers [SSD, IMF] revealed a set of requirements that have to be fulfilled and properties that should be controllable in order to come up with desirable and realizable folding structures. These requirements can be due to structural as well as aesthetic considerations. In essence, given a (tessellated) free-form surface, design and rationalization tools for point-folding structures should be able to

- control **height** and **centricity** of each pyramid,
- respect **production constraints** for producible elements,
- prevent **collisions** between neighboring pyramids,
- **preserve** the given free-form shape (and even its often purposely crafted tessellation).

Respecting the fourth point, i.e. considering the given triangle mesh (and thus the pyramid bases) fixed, all pyramids are completely defined by the apex positions. Hence, handling all other conditions reduces to control over the apexes. In our approach we thus define *validity volumes* (VVs) for the apexes such that inlying (= *valid*) positions fulfill all requirements of a given scenario. For a simple example, see Figure 4 (c) – more complex settings are considered later.

2.3. Problem Statement

The input to our rationalization method is a tessellated free-form surface $\mathcal{M}$ with triangular faces and a set of constraints that implicitly defines a volume VV_i of valid pyramid apex positions per triangle T_i. Obviously, two incongruent faces require differently shaped pyramids (even when ignoring the constraints). Now the key to rationalization is the observation that a whole class of triangular pyramids can be obtained from one and the same trihedron (infinite pyramid) by

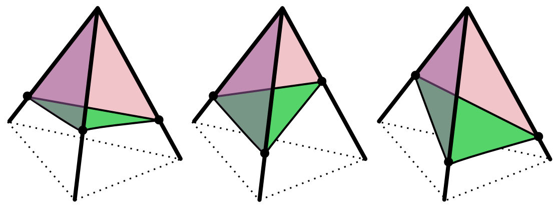


Figure 5: *Various pyramids cut from the same trihedron*

clipping it using differently oriented planes at different distances from the apex (cf. Figure 5). Hence, if two triangles happen to have any two valid pyramids that can be clipped from the same trihedron, one could simply produce a large enough representative pyramid of that class twice and cut it differently to cover both triangles. ISF inherently employs laser-cutting at the end of the process to cut out the pyramid from the sheet, i.e., this way of proceeding might even come at no additional cost depending on the production method.

Exploiting this observation, we can now concisely state the problem we have to solve as:

> *Find a small set of trihedra such that for each triangle a valid pyramid can be cut from one of them.*

The smaller the set we find (compared to the trivial solution of using all unique trihedra) the higher is the *rationalization gain*; e.g., in the example depicted in Figure 1, 21 trihedra (color-coded) sufficed to form pyramids for 270 triangles. In the following sections we formalize this problem and present a method to produce valid solutions.

3. Rationalization of Point-Folding Structures

We first note that the set of all trihedra is of dimension three. It can be parametrized by the three side facets' angles at the apex, i.e., a trihedron is uniquely defined by an angle triplet $\alpha := (\alpha_0, \alpha_1, \alpha_2) \in \mathbb{A}^3$, where $\mathbb{A}^3 := [0°, 180°]^3$.

The angle triplet $\alpha^T(a)$ of the trihedron defined by an apex position a over a given base triangle T can be computed by

$$\alpha_i^T(a) := \arccos\left(\frac{{\mathbf{u}_i^a}^\top \mathbf{u}_{i+1 \bmod 3}^a}{\|\mathbf{u}_i^a\|\|\mathbf{u}_{i+1 \bmod 3}^a\|}\right), \; i \in \{0,1,2\}, \quad (1)$$

where $\mathbf{u}_i^a$ are the crease vectors of the corresponding pyramid (cf. Section 2.1). This essentially allows us to map the validity volumes from $\mathbb{R}^3$ to $\mathbb{A}^3$, and we define the *angular validity volumes* $AVV_i := \alpha(VV_i)$. The search for trihedra that can be cut to pyramids fitting multiple triangles while having valid apex positions can then be posed as an intersection problem on the AVVs. The rationale behind this is the fact that an angle triplet which lies in the intersection of multiple AVVs corresponds to (i.e. is the image under α of) valid apex positions in multiple VVs.

In theory, the truly optimal rationalization solution respecting given constraints now could be obtained as follows:

1. Compute the intersection arrangement of the AVVs of all triangles T_i in $\mathbb{A}^3$ and for each region obtain the corresponding subset of $\{T_i \mid 0 \le i < n_F\}$.
2. Solve the Set Cover problem on the collection of all these subsets and pick an arbitrary representative angle triplet within each region corresponding to a subset of the result.
3. Map the representative angle triplets back to $\mathbb{R}^3$ to obtain a valid apex position within each VV.

The problems with this approach are that 1) computing the intersection arrangement of the AVVs can be considered computationally intractable – even for the simplest VVs the AVV boundaries cannot be described polynomially – and 2) the Set Cover optimization is known to be NP-hard and the number of subsets to be considered could be as large as 2^{n_F}.

To remedy the first problem, we combine adaptive discretization and sampling techniques. This renders approximate determination of the intersection arrangement tractable. The algorithm is presented in detail in Section 5, where we also describe how we ensure that no intersections beyond a certain size are missed and at the same time no false positives violating any constraints are produced.

In order to solve the second problem, we use the greedy algorithm for set covering – a best-possible approximation algorithm for the Set Cover problem with polynomial time complexity [Fei98]. This algorithm chooses subsets in the order of decreasing cardinality until the whole universe (the set of all triangles T of $\mathcal{M}$) is covered.

4. Basic Constraint Handling

Before proceeding with the description of the actual rationalization algorithm, let us first consider the most important type of constraints (height constraints) and how they translate into validity volumes. Incorporation of other, more advanced constraints is then treated afterwards in Section 7.

4.1. Pyramid Height

In the basic setting a prescribed pyramid height plus allowed tolerances is given for each triangle. In architectural applications desired heights h are typically in the range of $[10\%, 30\%]$ of the average edge length. The allowed tolerances depend on the specific application and on the material – hence, unless stated otherwise, we rather aggressively assume allowed height deviations ε to be constrained to $\le 2\%$ of the height, i.e., about 2-6*mm* in the case of 1*m* elements.

For a triangle T_i, these constraints easily translate into a VV_i bounded by two offset planes of T_i, with offset distances $h - \varepsilon$ and $h + \varepsilon$ in normal direction, respectively (cf. Figure 4 (c)). We furthermore clip such a VV by the three planes spanned by the three edge vectors of the triangle with its normal, obtaining a prism (cf. Figure 6 (a)). This rules out pyramid elements with protruding apexes.

Figure 6 shows the kind of AVVs that arise when mapping the prismatic VVs to $\mathbb{A}^3$. We call them *parachutes*. Subfigure (c) shows an arrangement of 11 such parachutes corresponding to different base triangles (with equal height constraints and large tolerances for visualization purposes). Note that mapping (1) depends on the (random) ordering of the indices of vertices p_i of T, i.e., each apex position could actually be assigned three angle triplets. To be independent of this ordering we use the union of three AVVs for each VV in practice. To keep the following explanations and figures simple, however, we shall only consider one of these.

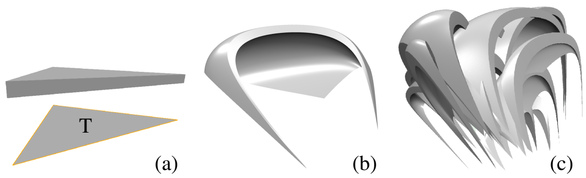


Figure 6: *(a) Validity volume (VV) defined over a triangle T by pyramid height constraints and tolerances. (b) The corresponding angular validity volume (AVV) (or* parachute*) mapped by Eq. (1). (c) An exemplary intersection arrangement of AVVs in $\mathbb{A}^3$ resulting from a mesh with 11 triangles.*

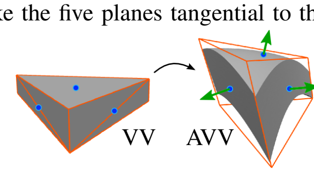
5. Solving the Intersection Problem

We now proceed to (approximately) determine the intersection arrangement of the AVVs in $\mathbb{A}^3$. To achieve efficiency we discretize both, the AVVs and $\mathbb{A}^3$ – the former using a conservative variant of prism refinement (cf. Section 5.1), the latter using an adaptive sampling of the space. However, care has to be taken when doing so, since 1) chances of missing potential good solutions due to the discretization should be really low and 2) chances of producing invalid "solutions" should be zero – contradicting goals in the context of discretization with limited resolution. A further challenge is posed by very high resolution requirements. A simple computation demonstrates that with a typical pyramid height constraint of 10% of the edge length and a tolerance of, e.g., 2%, angle resolutions lower than $\arctan(1/0.098) - \arctan(1/0.102) \approx 0.23^\circ$ might not be enough to even distinguish between the upper and lower boundary of a validity volume in $\mathbb{A}^3$, i.e. in the worst case whole AVVs might be missed when sampling $\mathbb{A}^3$ with resolutions lower than $(2^{10})^3$. The sought intersection volumes of multiple AVVs are typically even much thinner. Hence, we would like to work with resolutions up to about $(2^{15})^3$.

5.1. Discretizing the Parachutes

Even the simple prismatic VVs we introduced so far map to curved AVVs. We represent them using a piecewise linear boundary representation for efficiency. This can be done quite naturally to any desired accuracy by repeatedly performing 1-to-4 splits on the prismatic VVs in $\mathbb{R}^3$, then mapping the generated sub-prism vertices to $\mathbb{A}^3$. Unfortunately, these representations are far from being conservative on coarse levels of the refinement – and we want to exploit also these coarse levels of the inherent multilevel hierarchy of prisms, as described in the following section. In Figure 7 this "lossy" representation is illustrated.

First of all, in order to obtain (deformed) prisms with planar sides in $\mathbb{A}^3$ (to simplify intersection tests) we do not map the prism vertices but take the five planes tangential to the AVV boundary at the images of the five sides' centers as depicted on the right. The corresponding plane normals are computed by restricting (1) to the (triangulated) sides $\triangle_{ABC} \subset \mathbb{R}^3$ of sub-prisms, yielding three restricted coordinate-maps



$$f_i : \triangle_{ABC} \to \mathbb{A},\ (\lambda_0, \lambda_1, \lambda_2) \mapsto \alpha_i(\lambda_0 A + \lambda_1 B + \lambda_2 C)$$

with barycentric coordinates λ_j, from which the directional derivatives can be computed. Computing the cross-product of the directional derivatives of two directions oriented counter-clockwise in $\triangle_{ABC}$ yields the normal of that prism side at the point $(\lambda_0, \lambda_1, \lambda_2)$ mapped to $\mathbb{A}^3$.

We then shift these planes outwards such that the (corresponding part of the) actual AVV volume is contained in the (deformed) prism defined by these planes (as depicted in Figure 7 right). Due to the absence of inflections on the AVV boundaries, the amount of shifting necessary could be determined using gradient ascents along the edges and within the face of each side. For simplicity, in our current implementation we shift to the maximum of several samples.

5.2. Discretizing $\mathbb{A}^3$

We could now naïvely sample $\mathbb{A}^3$ regularly in $(2^{15})^3$ points, perform pairwise inclusion tests for these with the sub-prisms of all parachutes, and (in the manner of a greedy Set Cover algorithm) pick the one sample included in the largest number of parachutes, as it corresponds to the first best representative trihedron. This could be iterated for the remaining parachutes until all base triangles are "covered".

It is immediately clear, that this is computationally utopistic. We hence use an adaptive multilevel discretization of both, the AVVs and the space $\mathbb{A}^3$: we simultaneously perform an octree-based subdivision of $\mathbb{A}^3$ and a 1-4-split-based refinement of the AVVs in an interlocked manner. Initially all level 0 conservative parachute prisms are associated with the octree root cell (encompassing all parachutes). Then, iterating over the subsequent levels, for each associated pair of octree cell and prism, the eight child cells and four child prisms are checked for intersections and associated accordingly (as illustrated in Figure 8 left). By subdividing space adaptively in this way and performing the intersection tests in this doubly-hierarchical manner, spatial regions contained in many parachutes can be located with practicable performance.

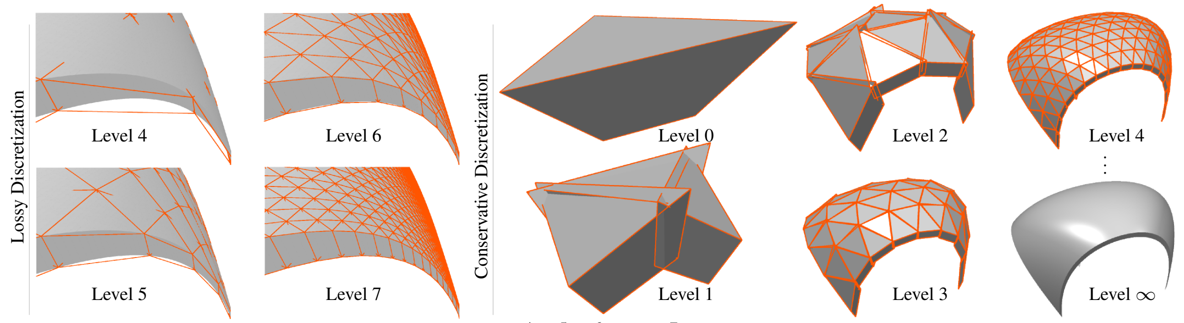


Figure 7: *Left: Lossy discretization of an AVV with 4^4, 4^5, 4^6, and 4^7 prisms. Right: Conservative prisms at various levels.*

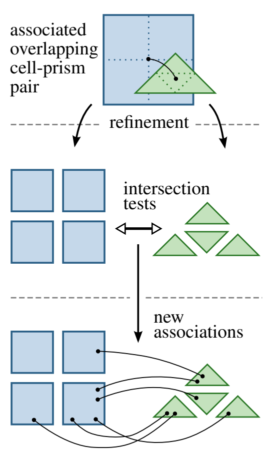
5.3. Memory Efficient Greedy Set Cover

Further measures need to be taken, however, to reduce memory requirements to an acceptable level – the storage needed for (1) the geometry of prisms and cells as well as for (2) all cell-prism associations would by far exceed common main memory sizes at levels beyond 10.

On-Demand Geometry. We address the first issue by not storing the actual geometries of cells and prisms, but rather simple indices encoding their position in the octree resp. 1-4-split hierarchy. The actual geometry is then only created temporarily for the intersection tests. As generating the geometry of a cell from its index is a simple matter compared to constructing a conservative sub-prism (cf. Section 5.1), we let the outer refinement loop run over the prisms and the inner loop over the associated cells. This avoids the need to create the same sub-prism geometry multiple times. Pseudo-code for this routine is given in Figure 8 top right.

Two-Phase Processing. The second issue (size of associations) is addressed in the following way: the refinement is performed in the described way (which we call phase I) until the memory budget is nearly exhausted, then we switch to an extremely memory-friendly depth-first search (called phase II) to be able to proceed through the remaining octree levels (up to usually level 15). Note that the approximation accuracy of the prisms grows quadratically with the refinement level (in contrast to linear increase of octree resolution) and our experiments revealed that prism refinement beyond level 7 never improved the results. Since phase II usually starts after that, we can efficiently rely on fixed prisms in phase II and only traverse the octree hierarchy further on. Also note that the association between cells and prisms only approximately and conservatively signifies an intersection in the corresponding region. To ensure that all constraints are respected, during the depth-first traversal of the remaining levels, the center points of the cells on the final refinement level are mapped back to $\mathbb{R}^3$ (cf. Section 6) and then tested for validity to *exactly* determine how many input base triangles are validly covered by the corresponding trihedron. Pseudo-code for the depth-first search in a cell is provided in Figure 8 – explanation of the employed bounding follows.

Branch-and-Bound. The search efficiency can drastically be improved in a branch-and-bound manner, i.e., if the best point found so far is included in k parachutes, all sub-trees of cells associated with prisms of no more than k parachutes can safely be skipped subsequently (in the pseudo-code this current bound k is kept track of in the global variable `bound`). This is due to the fact, that the number of associated parachutes (`#parachutes(C)`) provides an upper bound on the number of parachutes that might overlap any common point contained in the cell. At the end of the traversal, the point that established the last such bound (i.e. `bestRepresentative`) signifies the trihedron that is valid for the largest number of base triangles. Following the greedy approach to Set Cover, the parachute prisms corresponding to these base triangles are then removed from the octree leafs that resulted from phase I, and phase II is repeated to find the next best trihedra until all base triangles are covered.




```
phase_I():
for each level L from 0 to ...
  for each prism P of level L
    for each child prism p of P
      gp = geometry(p)
      for each associated cell C of P
        for each child cell c of C
          gc = geometry(c)
          if intersects(gp,gc):
            associate(p,c)

depthTraverse(C):
if #parachutes(C)≤bound: return
if finalLevel(C): updateBound(C)
for each child c of C
  intersect&associate(c,prisms(C))
  depthTraverse(c)
  delete c and its associations

updateBound(C):
n = #validlyCovered(center(C))
if n > bound:
  bound = n
  bestRepresentative = center(C)
```

Figure 8: *Left: illustration of one step of the refinement process. Right: Pseudo-code of the important parts of the algorithm:* `phase_I()` *constructs an initial octree. In phase II for each of its leaf cells* `C` *(in descending order w.r.t. the number of associated parachutes)* `depthTraverse(C)` *is invoked. Afterwards the best representative found is output, its associated parachutes removed from the cells and phase II repeated until all base triangles are covered.*

6. Representative Folding Element Construction

Having found the desired representative angle triplets in $\mathbb{A}^3$, we map them back to $\mathbb{R}^3$ to determine the apex positions and construct the pyramids. Our map (1) is non-injective and a global closed-form inverse is not available, but the construction of the pyramid P on a triangle T given an apex angle triplet $(\alpha_0, \alpha_1, \alpha_2)$ is equivalent to the perspective three-point pose (P3P) problem well-known in Computer Vision [FB81]: the position of the camera (here: the apex a) is reconstructed from three sighted points (here: the vertices p_0, p_1, p_2) resp. the angles between the corresponding sight rays by finding the real roots of a quartic polynomial.

7. Advanced Constraint Handling

Having described our general rationalization pipeline, we now take a closer look at how further constraints can be incorporated. So far, production and construction constraints have not been taken care of. Furthermore, explicit control over the centricity of the elements might be desirable. To gain control over the rationalization process in this regard, we only have to modify the VVs to suit our needs, the rest of the pipeline remains unchanged.

7.1. Collision Prevention

In non-convex regions of the base mesh, the simple VVs (cf. Section 4.1) of neighboring triangles might be non-disjoint, potentially leading to solutions with intersecting pyramids that could not be assembled physically. Unless unusually high pyramids shall be placed in extremely curved concave regions or narrow passages, we may safely assume that intersections might only happen between pyramids on faces sharing a common edge or vertex.

By replacing the base-orthogonal clipping plane on an edge by that containing the edge normal (averaged incident face normals), intersections between VVs of directly adjacent triangles can be prevented, but intersections across vertices potentially remain. These are additionally be ruled out by, for each edge of a triangle, instead taking the *innermost* plane of the base-orthogonal clipping plane and the two that contain either incident vertex's normal. Here *innermost* means the plane whose outwards normal has the largest dot product with the base triangle normal.

7.2. Centricity Control

The prismatic VVs can be modified further to gain control over the centricity of the resulting apexes. We here consider the case that apexes shall be restricted to lie no further than some distance r apart from the base-orthogonal line through, e.g., the in-center or the center-of-gravity of a base triangle. Obviously, the corresponding VV is a cylinder.

Since usually collision constraints are to be considered additionally, we are interested in the intersection of this cylinder with a prism obtained as above. As the rationalization algorithm relies on adaptive refinement of the VVs' discretization, we do not want to give up on the convenient, regularly refinable triangular structure. Hence, we keep the prismatic VV with its refinement structure and simply mark those sub-prisms as *inactivate* that lie outside the cylinder. These are then ignored in the process. Prisms partially on the outside are kept active to not miss any solutions – the final validity check (cf. Section 5.3) rules out false positives.

7.3. Production Constraints

Depending on the method employed to manufacture the folding elements, different constraints on the attainable element shapes might be given. Hence, we exemplarily consider the ISF production method to illustrate how such constraints can be incorporated.

Incrementally forming metal sheets leads to some thinning of the material [Ame08, JMH*05], inhibiting large deformation angles. On the other hand, the elastic recovery of the material prevents very small base angles. Safely achievable angles typically are in the range of $[min_{ISF}, max_{ISF}] = [20°, 50°]$. To guarantee that the folding elements resulting from the rationalization can be produced, we must modify the VVs to restrict the solution space correspondingly. Note that one cannot just measure the angles to the supporting plane of a triangle T for a given apex a, as this plane is generally not the actual production blank sheet plane due to production of multiple, differently cut pyramids from one representative master – we must base our considerations on a virtual production plane.

We consider the optimal production plane for a given trihedron to be the one having equal angles to all trihedron sides – it simultaneously minimizes the maximum angle and maximizes the minimum angle, resulting in best-possible production quality. The normal $\mathbf{n}$ of this virtual production plane for an apex position a can be found as the vector $a - x$ for the center x of an insphere of arbitrary radius r of the trihedron. Such a point x is found by solving $\mathbf{m}_i^{a\top} x = r - \mathbf{m}_i^{a\top} a,\ i \in \{0, 1, 2\}$. Figure 9 shows the resulting boundaries of the volume of ISF-manufacturable apex

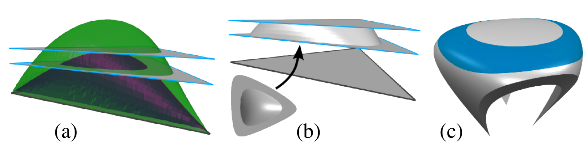


Figure 9: *(a) Visualization of the surfaces implicitly defined by a min_{ISF} (purple) and max_{ISF} (green) constraint. (b) The remaining VV between the additional apex height constraint planes. The inset shows a bottom view – the "carved out" region contained apex positions that would lead to base angles $< min_{ISF}$. (c) AVV of a standard height constraint – the sub-volume that remains when additionally considering the ISF constraints is highlighted in blue.*

	SEASHELL, $n_F = 249$			TRADEFAIR, $n_F = 270$			ALPINEHUT, $n_F = 468$			TRAINSTATION, $n_F = 1038$	
	CP	CP+ISF	CP+CC	CP	CP+ISF	CP+CC	CP	CP+ISF	CP+CC	CP+ISF (level 6)	(level 7)
Unique Elements	13	9	32	10	11	21	22	16	50	11	11
Rationalization Gain	95%	96%	87%	96%	96%	92%	95%	97%	89%	99%	99%
Runtime Phase I (min)	9.1	8.6	8.2	8.9	9.1	8.2	15.0	14.8	13.5	16	83
Runtime Phase II (min)	11.0	5.0	2.3	7.5	8.8	10.0	39.0	51.9	14.8	440	337
Peak Memory (GB)	3.2	2.9	2.5	3.1	3.4	2.1	4.9	5.1	4.1	4.1	11.5

Table 1: *Statistics of the folding element rationalization for our examples. Columns show results for processing with collision prevention constraints only (CP), additional ISF production constraints (ISF), as well as additional centricity control (CC).*

positions over a triangle. Now that we have a point-wise test for ISF compatibility, combining the ISF constraint VVs with other VVs can be performed by deactivating sub-prisms whose vertices lie in invalid regions as described in the last section.

Incorporating further production and construction constraints is easily imaginable. For instance, considering the structural capabilities, the VVs can be restricted to regions that appropriately orient the folding elements into the direction of compressive forces, etc.

8. Results

We evaluate the proposed rationalization method on four architect designed models – SEASHELL ($n_F = 249$), TRADEFAIR ($n_F = 270$), ALPINEHUT ($n_F = 468$), and TRAINSTATION ($n_F = 1038$), as shown in Figure 10. All examples have been processed on a modern standard PC (Intel i7-920 CPU).

Table 1 shows the rationalization results for these models and compares different constraint configurations. Except in the ISF cases the apex height was fixed to $0.2m$ (ca. 20% of the average edge length) and $\varepsilon = 2\%$ was used. For the ISF cases, smoothly varying heights from a range around 20% were automatically set in a way to adapt to the ISF constraints and guarantee non-empty VVs. For comparability, the switch from phase I to phase II of the algorithm has been fixed to after level 7. The specified rationalization gain is defined as the percentage of the number of folding elements of the trivial solution (a unique element per base triangle) made obsolete by rationalization. Figures 11 and 12 illustrate the results.

Achieved rationalization gains range from 87% to 97%

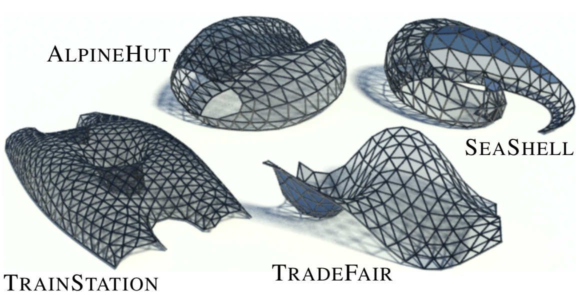
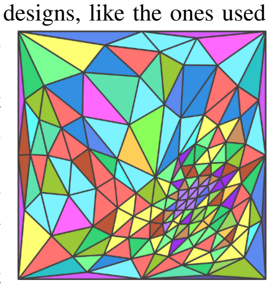


Figure 10: *The four base meshes used in our experiments.*

for the first three models, whose tessellations have been provided by architects. For the TRAINSTATION model, which was quite uniformly meshed based on [BK04], even 99% were achieved. As expected, tighter constraints usually lead to lower gains since AVV intersections tend to be rarer for smaller VVs. The exceptions seen in the table when comparing collision prevention constraints only with additional ISF constraints are due to the variable height constraints applied in the ISF case, hence incidental. The last two columns exemplarily illustrate the effect of the transition from phase I to phase II of the algorithm: switching to phase II earlier (here at level 6 of the octree refinement) leads to lower memory consumption but higher runtime compared to switching later (here at level 7).

Meshes used in architectural designs, like the ones used in our experiments, usually have "nice" elements. Still, it is interesting to see to which extent the rationalization depends on this circumstance: we exemplarily applied our method to the rather irregular mesh depicted on the right. As could be expected, the gain was lower, but still 88% were achieved (using collision constraints).



With our current research implementation, computing a full resolution rationalization on a standard PC is limited to scenarios with less than about 2000 folding elements. Optimization of the employed data structures and redundancy reduction will likely be able to further raise these bounds.

9. Discussion & Future Work

On the production side current research topics include the exploration of novel uses of folding structures and the development of efficient folding element production techniques. Our method makes a big step towards usability in large scale, free-form production scenarios – as the cost of producing the molds, dies, or tools for element production can be drastically reduced. In future work, we would like to explore further aspects in this context, as outlined in the following.

Weighting. Our rationalization method is completely "discrete" – there are no soft-constraints or "more or less preferred" solutions; always *some* valid solution (possibly out

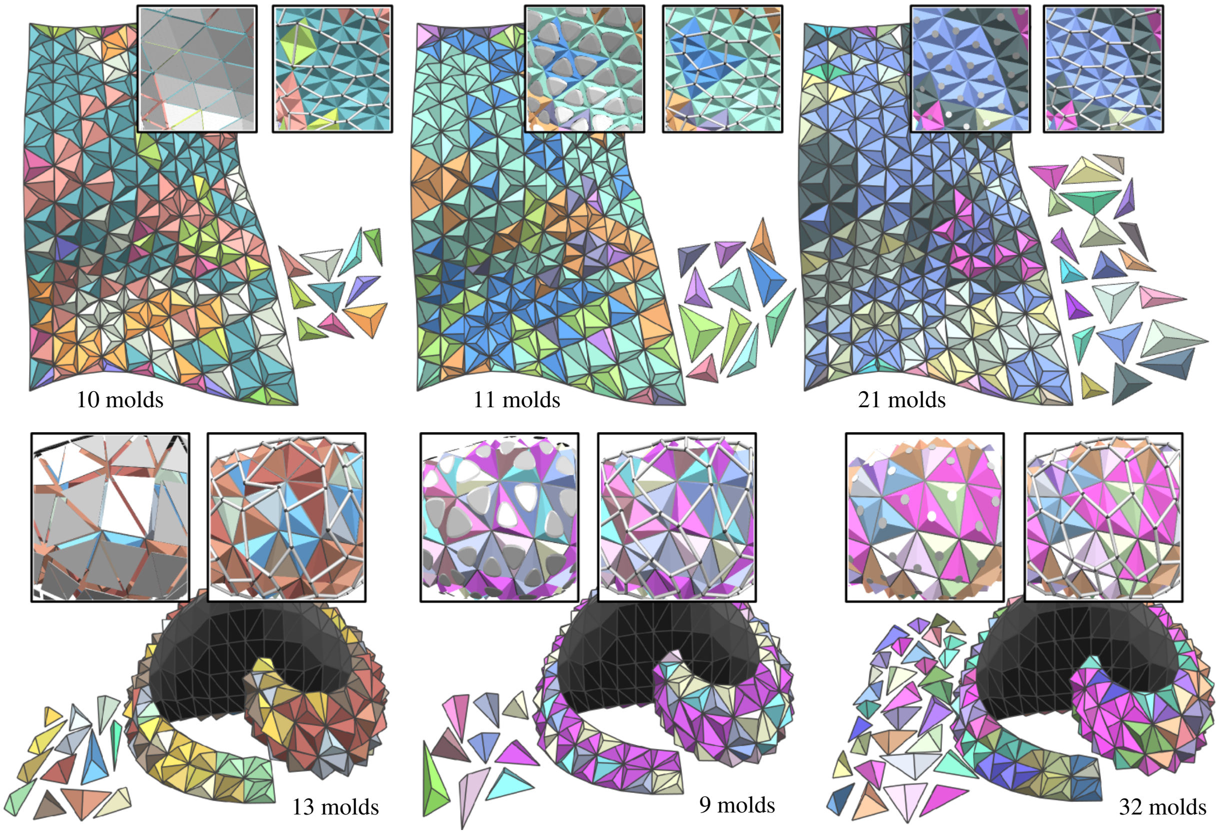


Figure 11: *Results of* TRADEFAIR *(top) and* SEASHELL *(bottom) with collision constraints (left), collision and ISF constraints (middle), and collision and centricity constraints (right). The insets show the corresponding validity volumes and the dual structure in gray. Understandably, the dual structure shows increased regularity under centricity constraints. Canonical representatives of the used element classes (randomly color-coded) are depicted alongside.*

of several similarly good alternatives) is found. When soft constraints are desired, e.g. for mixing aesthetical preferences with hard construction constraints, one could switch to using VVs augmented by weighting fields. While defining such weighted VVs is easy, some work would have to be done to steer the adaptive discretization accordingly.

Tessellation. The TRAINSTATION example demonstrates that uniform meshes (e.g. generated by [AMD02, BK04, YLL*09, NPPZ11]) facilitate high rationalization gains. We considered the case of purposely designed input meshes that shall not be altered. An interesting direction for future work is the exploration of an combined rationalization and modification of the base mesh (e.g. akin to [LZKW10]).

Dual Surface. A deeper analysis of the properties and aesthetics of the implicitly generated dual surface spanned by the apexes would be useful. This is particularly important for double-layered folding structures where primal and dual surfaces are represented by panels (cf. Figure 2 right for an example). For instance, instead of prescribing constant heights, one could think of defining a smoothed offset band over the base surface and constrain the apexes accordingly. This has already rudimentarily been explored in our experiments for adjusting the heights to meet ISF production constraints.

Polygonal Bases. In principle, the greedy rationalization part of our algorithm could also be extended to folding structures based on, e.g., quadrilateral or hexagonal meshes. However, as noted in [FB81], challenges could be posed regarding the inverse mapping due to non-planarity or non-convexity of the base polygons.

10. Conclusion

We have analyzed the rationalization possibilities inherent in the design of point-folding structures and formalized the rationalization problem accordingly. Transforming this task into an optimization setting based on geometric intersection problems led to an algorithm that ultimately allows for the rationalization of point-folding structures based on triangulated free-form surface designs, where gains around 90% are realistic and achievable without altering the geometry of the given triangle mesh. Significant reductions of fabrication costs might thus be attained in practice. We have shown that

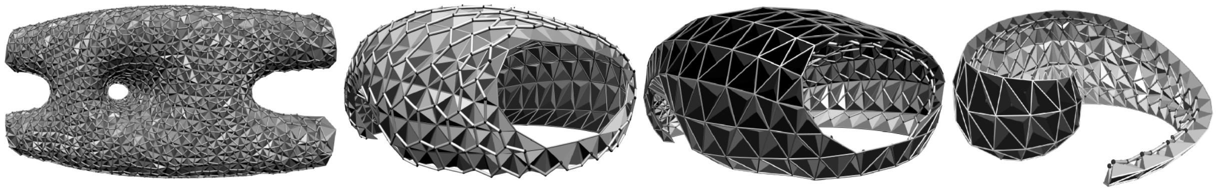


Figure 12: *Left:* TRAINSTATION *covered with elements cut from only 11 different trihedra (ISF constraints have been used – centricity was not enforced). Middle:* ALPINEHUT *covered with 47 unique elements from the outside or 50 unique elements from the inside, respectively (centricity constraints employed). Right:* SEASHELL *with 42 unique elements on the inside.*

user-specified and material- as well as production-induced geometric constraints can be incorporated into the process, leading to true practical applicability of the approach.

Acknowledgements. We thank the San Diego Reader Staff Photographer for the Palomar College Gymnasium photo, Martin Lisnovsky (arquitecturamashistoria.blogspot.com) for providing the photo of the IBM Pavilion [Buc00], and Evan Wohrman for the Spaceship Earth photos. The TRAINSTATION design was kindly provided by Evolute GmbH, Austria (www.evolute.at). We thank the Fold-In team at RWTH Aachen for the inspiring discussions, especially Martin Trautz, Ralf Herkrath and Karl-Heinz Brakhage.

References

[AMD02] ALLIEZ P., MEYER M., DESBRUN M.: Interactive geometry remeshing. *ACM TOG 21* (July 2002), 347–354. 9

[Ame08] AMES J.: *Systematische Untersuchung der Beeinflussung des Werkstoffflusses bei der Inkrementellen Blechumformung mit CNC-Werkzeugmaschinen*. PhD thesis, RWTH Aachen University, Aachen, NRW, Germany, 2008. 7

[Bal97] BALDWIN J.: *BuckyWorks: Buckminster Fuller's ideas for today*. Wiley, 1997. 2

[BK04] BOTSCH M., KOBBELT L.: A remeshing approach to multiresolution modeling. In *Proc. SGP '04* (2004), pp. 185–192. 8, 9

[Buc00] BUCHANAN P.: *Renzo Piano Building Workshop: Complete Works*. No. v. 3. Phaidon Press, 2000. 2, 10

[BW10] BURI H. U., WEINAND Y.: Origami – geometry of folded plate structures. In *Structures & Architecture* (2010). 3

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. SIGGRAPH '03, ACM, pp. 689–696. 3

[EDS*10] EIGENSATZ M., DEUSS M., SCHIFTNER A., KILIAN M., MITRA N. J., POTTMANN H., PAULY M.: Case studies in cost-optimized paneling of architectural freeform surfaces. In *Advances in Arch. Geom.* 2010, pp. 49–72. 2

[EKS*10] EIGENSATZ M., KILIAN M., SCHIFTNER A., MITRA N. J., POTTMANN H., PAULY M.: Paneling architectural freeform surfaces. SIGGRAPH '10, ACM, pp. 45:1–45:10. 2

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM 24* (June 1981), 381–395. 7, 9

[Fei98] FEIGE U.: A threshold of ln n for approximating set cover. *J. ACM 45* (July 1998), 634–652. 4

[FLHCO10] FU C.-W., LAI C.-F., HE Y., COHEN-OR D.: K-set tilable surfaces. SIGGRAPH '10, ACM, pp. 44:1–44:6. 2

[Gil61] GILMORE C. P.: Bucky fuller's wonderful dome. *Popular Science 179* (December 1961), 75–77, 184. 2

[HT11] HERKRATH R., TRAUTZ M.: Starre Faltungen als Leichtbauprinzip im Bauwesen.*Bautechnik 88*, 2 (2011), 80–85. 2

[IMF] IMF:. Institute of Metal Forming, RWTH Aachen University. 3

[JMH*05] JESWIET J., MICARI F., HIRT G., BRAMLEY A., DUFLOU J., ALLWOOD J.: Asymmetric single point incremental forming of sheet metal. *CIRP Annals - Manufacturing Technology 54*, 2 (2005), 88 – 114. 3, 7

[KFC*08] KILIAN M., FLÖRY S., CHEN Z., MITRA N. J., SHEFFER A., POTTMANN H.: Curved folding. SIGGRAPH '08. 3

[KT11] KÜNSTLER A., TRAUTZ M.: Wandelbare Faltungen aus biegesteifen Faltelementen. *Bautechnik 88*, 2 (2011), 86–93. 3

[LZKW10] LI Y., ZHANG E., KOBAYASHI Y., WONKA P.: Editing operations for irregular vertices in triangle meshes. SIGGRAPH ASIA '10, pp. 153:1–153:12. 9

[NPPZ11] NIESER M., PALACIOS J., POLTHIER K., ZHANG E.: Hexagonal global parameterization of arbitrary surfaces. *IEEE TVCG* (2011), 1–14. 9

[PSB*08] POTTMANN H., SCHIFTNER A., BO P., SCHMIEDHOFER H., WANG W., BALDASSINI N., WALLNER J.: Freeform surfaces from single curved panels. SIGGRAPH '08, ACM, pp. 76:1–76:10. 2

[SDG10] SCHENK M., D. GUEST S.: Origami folding: A structural engineering approach. In *Proc. 5OSME* (2010), pp. 293–305. 2, 3

[SS10] SINGH M., SCHAEFER S.: Triangle surfaces with discrete equivalence classes. SIGGRAPH '10, pp. 46:1–46:7. 2

[SSD] SSD:. Chair of Structures an Structural Design, RWTH Aachen University. 2, 3

[Tac06] TACHI T.: Simulation of rigid origami. In *Proc. 4OSME* (2006). 3

[Tac09] TACHI T.: One-DOF cylindrical deployable structures with rigid quadrilateral panels. In *Proc. IASS* (2009), pp. 2295–2305. 3

[TH09] TRAUTZ M., HERKRATH R.: The application of folded plate principles on spatial structures with regular , irregular and free-form geometries. In *Proc. IASS* (2009), pp. 1019–1031. 3

[TK09] TRAUTZ M., KÜNSTLER A.: Deployable folded plate structures – folding patterns based on 4-fold-mechanism using stiff plates. In *Proc. IASS* (2009), pp. 2306–2317. 3

[Tra09] TRAUTZ M.: Das Prinzip des Faltens. *Detail – Zeitschrift für Architektur und Baudetail 12* (2009), 1368–1276. 2, 3

[YLL*09] YAN D.-M., LÉVY B., LIU Y., SUN F., WANG W.: Isotropic remeshing with fast and exact computation of restricted voronoi diagram. In *Proc. SGP* (2009), pp. 1445–1454. 9