Procedural Modeling of Structurally-Sound Masonry Buildings

To appear in the ACM SIGGRAPH conference proceedings

Procedural Modeling of Structurally-Sound Masonry Buildings

Emily Whiting John Ochsendorf Frédo Durand

Massachusetts Institute of Technology

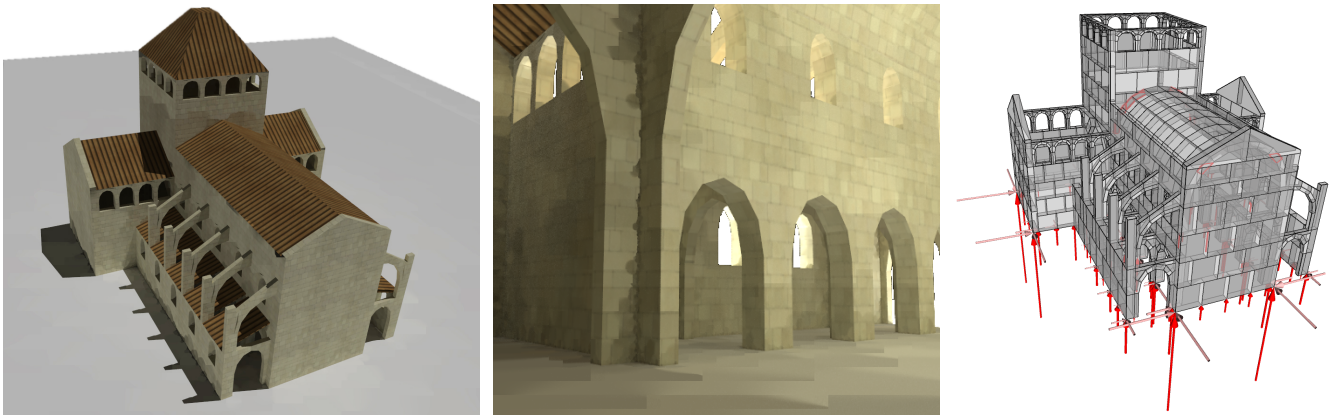


Figure 1: *Our method generates models of masonry buildings that are structurally stable. In this building based on Cluny Abbey in France, parameters controlling the flying buttresses, columns, and window sizes have been automatically optimized to support a stone vaulted ceiling. The right image shows reaction forces at the ground plane. We solve for forces at all block interfaces, and apply a compression-only constraint for masonry materials.*

Abstract

We introduce structural feasibility into procedural modeling of buildings. This allows for more realistic structural models that can be interacted with in physical simulations. While existing structural analysis tools focus heavily on providing an analysis of the stress state, our proposed method automatically tunes a set of designated free parameters to obtain forms that are structurally sound.

Keywords: procedural modeling, statics, structural stability, architecture, optimization, physics

1 Introduction

Content creation for virtual environments has become a bottleneck in computer graphics and interactive applications. Geometric models are required to have high visual realism and also be suitable for use in physical simulations. Structurally stable models enhance realism in virtual environments by allowing characters to interact with the built surroundings, whereas models which are not consistent with mechanics might collapse under their own weight.

Procedural modeling has emerged as a powerful technique for generating architectural geometry. However, existing techniques focus on visual realism and do not account for the structural validity of the results. Users may not have intuition about the mechanics that govern structural stability, or knowledge of traditional proportions used in building design. Determining the precise dimensions of a structure that guarantee stability can be a tedious task. We present a method to automatically "snap" to feasible dimensions, while leaving control in the designer's hands for deciding which aspects of the model are variable.

Our contribution is to introduce physical constraints into procedural modeling methods. We solve an inverse statics problem: given a set of physical constraints and a building topology, we determine an appropriate shape. The user provides a set of production rules that describes the desired architectural style, along with a small set of free parameters. The relationship between rule parameters and internal forces in the structure is nonlinear. Using gradient-based nonlinear optimization, our method searches over the parameter space for a stable configuration.

We focus on masonry structures, which encompass historic cathedrals, stone bridges, brick walls, unreinforced concrete dams, and other common structures. Masonry constructions behave as undeformable rigid blocks with interaction forces limited to compression and friction [Heyman 1995]. In order to impose structural feasibility, a forward analysis tool is required to assess the soundness of a structure. However, current engineering tools based on finite element methods and elasticity theory [Zienkiewicz 1971] are not appropriate in this context because they focus on material failure and stress, and because the high stiffness of stone results in poorly conditioned numerical systems. In contrast, the critical factor in masonry structures is the geometric configuration and whether it is in static equilibrium. In particular, Block et al. [2006] demonstrated that linear elastic theory was unable to differentiate between a feasible masonry arch and an infeasible arch. For this reason, we revisit an approach introduced by Livesley [1978], and we present a new forward structural analysis method based on optimization under linear constraints.

We model the stress state by dividing the structure into rigid elements and computing force resultants on inter-element boundaries. We formulate the stability problem as a quadratic program, where

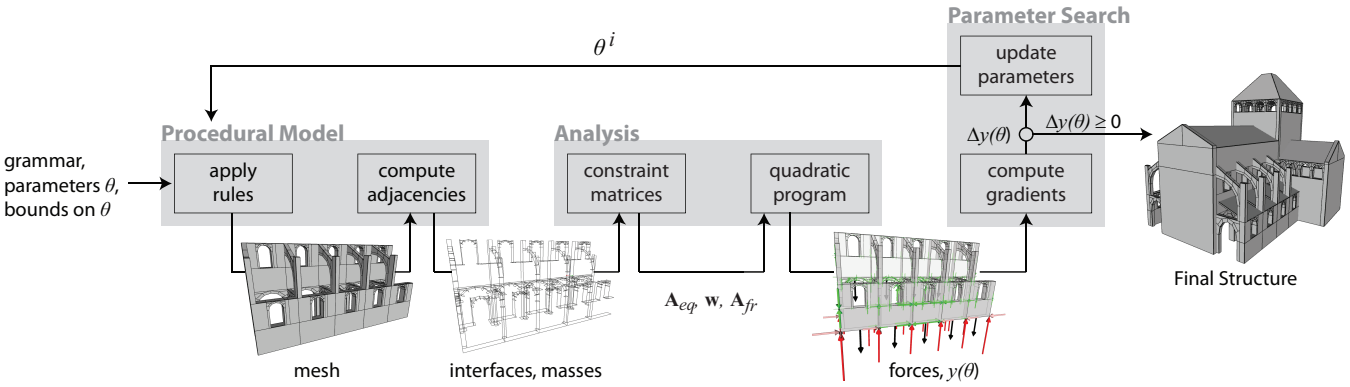


Figure 2: *Pipeline. User input is a set of production rules, the selection of free parameters, and associated bounds.*

each element is subjected to static equilibrium constraints. We build on this forward analysis to search for parameters of our procedural model that yield a stable building. In particular, we extend the analysis to return a measure of infeasibility when the quadratic program fails, by minimizing violation of failure criteria at the joints. This allows us to define an energy function and use non-linear optimization to find the appropriate parameters.

Contributions We introduce the idea of generating structurally feasible procedural models of buildings through automatic parameter selection.

We present a measure of infeasibility that determines how close a model is to being structurally sound. It is enabled by a quadratic programming formulation and agrees closely with available experimental data.

We use the measure of infeasibility as an energy function, and apply gradient-based optimization to select rule parameters that satisfy structural stability constraints.

We show examples of procedural models of buildings with both internal and external structure that are consistent with mechanics.

1.1 Related Work

Procedural Modeling Our work builds on the approach proposed by Müller et al. [2006]. Their system uses grammars to produce variations of building designs, generated through random or user-selected parameter adjustment. In contrast, our method selects rule parameters by determining values that will make the model stand in a stable configuration. Previous work in procedural modeling has focused on visual realism and detail in the building façade and does not model the internal structural elements [Lipp et al. 2008; Müller et al. 2007; Müller et al. 2006; Wonka et al. 2003; Parish and Müller 2001].

Inverse Statics Static analysis is an important tool for posing characters, e.g. positioning the center of mass over the ground supports [Shi et al. 2007]. In plant modeling, static analysis has been used to balance the weight of branches for creating realistic tree structures [Hart et al. 2003]. Statics has also been applied to automatic truss design that optimizes for minimal material consumption [Smith et al. 2002]. We solve a similar problem of determining model geometry based on physical constraints. However, the set of possible production rules for buildings has greater complexity than branching systems used in plant modeling. Further, we work with three-dimensional template shapes, compared to the 1-D elements in bridge structures.

Design by Optimization Optimization has been used in a number of design scenarios. Harada et al. [1995] optimize constrained layout designs with physically based user interaction. Welch and Witkin [1992] solve a constrained variational optimization for interactive modeling of free-form surfaces. In architectural applications, optimization has been used for modeling free-form surfaces that meet fabrication criteria [Pottmann et al. 2008; Pottmann et al. 2007; Liu et al. 2006]. However, these examples do not consider structural feasibility constraints.

Structural Engineering Some CAD modeling systems, such as CATIA, provide visual feedback from finite element analysis that indicates the current state of stress. However, they do not guide the user on how to modify designs for improved stability, and manual model adjustment is still required. This approach can be ineffective for designers lacking intuition in mechanics. Our method modifies the structure automatically to a feasible solution.

Block et al. [2006] investigated interactive analysis of masonry structures, but were limited to two-dimensional slices of buildings. Gilbert et al. [2006] focus on friction behavior for rigid block structures and also consider two-dimensional problems. Livesley [1978] described the use of linear programming for 2D masonry analysis in 1978, with further work on a small class of 3D structures [Livesley 1992]. In practice, the RING software applies limit state analysis to 2D masonry bridges [Gilbert 2001]. A comprehensive review of analysis techniques for historic masonry structures is given by Lourenco [2002]. We extend Livesley's approach to handle infeasible cases and provide a measure of infeasibility that can be used for optimization.

1.2 Overview

Our approach allows users to generate architectural models using procedural modeling, and then automatically tweak a set of designated design variables to make the model structurally sound. We propose an iterative algorithm that loops over three main steps. First, we construct a model given a grammar and a set of fixed parameters. Then, we estimate the stability of the obtained structure. Finally, if the model is not stable, we modify the parameter values to reduce the instability and start a new iteration.

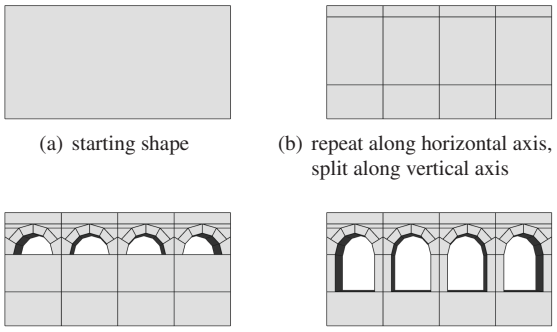
The pipeline is shown in Figure 2. The input is a set of grammatical rules, a selection of free parameters, $\boldsymbol{\theta}$, and their associated upper

and lower bounds. The information computed at each step of the optimization loop is as follows:

- Standard procedural modeling rules (e.g. repeat, split, transform) are used to generate the geometry of buildings. The output is a mass model representing the blocks of the structure, including the interfaces between all adjacent blocks.
- The analysis stage computes interaction forces at each interface using quadratic programming, and outputs a measure of distance to a feasible structure, $y(\boldsymbol{\theta}^i)$.
- At each iteration of the parameter search a new set of values, $\boldsymbol{\theta}^i$, is chosen for the free parameters.
- The optimization terminates when a feasible structure is found ($y(\boldsymbol{\theta}^*) = 0$), or a local minimum.

2 Procedural Modeling

To create the geometry of our structures, we use procedural modeling methods as described by Müller et al. [2006]. Beginning with a coarse volumetric model, production rules iteratively refine the geometry with internal structure and façade details. The procedural system carries semantic information including architectural labels (e.g. arch, wall, column) and rule parameters (e.g. column diameter).



(a) starting shape

(b) repeat along horizontal axis, split along vertical axis

(c) split middle section into capped arch and wall

(d) split wall symmetrically into two columns and scale

Figure 3: *Procedural generation of a wall with four windows.*

A difference in our approach to procedural geometry is our use of mass modeling. Müller et al. [2006] consider the building volume as a single solid object with no interior. In contrast, we model solid objects as interior columns, walls, and other support structure.

Free Parameters The key feature of our approach is that we automatically choose rule parameters according to physical constraints. The user designates a set of free parameters which will be optimized to reach a stable structure. Typical examples may be the wall thickness or the width of a window element. We can also place bounds on the parameters – these limits are set by the user to define the family of design variations.

Library of Primitives In generating procedural models, we use a set of basic shapes that emphasize internal structure. In addition to the primitives proposed by Müller et al. [2006], we add a set of template objects that are typical to the intended style of architecture. For masonry structures, these include flying buttresses, domes, arches, groin vaults, and capped arches, in both gothic and romanesque styles (pointed arch versus circular profile). All of these shapes can be manipulated with a set of mesh parameters: {tessellation, radial thickness} in addition to the scope parameters.

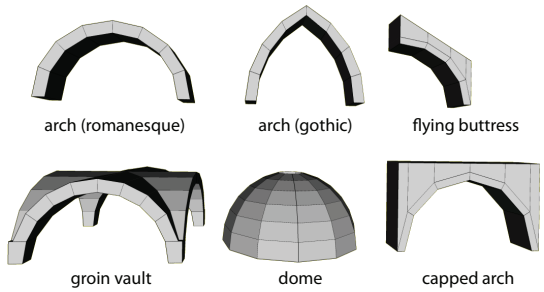


Figure 4: *Template shapes specific to masonry architecture. Each has parameters to alter tessellation and block thickness.*

We chose interface orientations to mimic typical masonry construction. For example, the blocks in walls and columns are laid in horizontal courses, while the blocks in arches and flying buttresses are cut radially. A poorly-chosen interface orientation can make a structure unstable. The effect of interface orientation on the solution space is an open area for future research.

Nonstructural Shapes Procedural methods afford the ability to tag shapes for differing material properties and appearance. We extend this capability to tag shapes as "nonstructural." For example, the roof of a cathedral is often constructed from wood frames which have little mass compared to the density of stone. We exclude these shapes from the analysis stage. This allows for decorative elements without adding unnecessary complexity to the constraint equations.

Adjacencies Once the geometry of the building model has been generated, we compute contact surfaces between adjacent blocks as shown in Figure 2. These interfaces are used later in the analysis step. We assume neighboring blocks have coplanar faces and do not interpenetrate. We apply simple spatial acceleration for computing adjacencies based on bounding boxes.

3 Analysis

The analysis stage solves a forward statics problem: given the geometry of a structure, we compute the interaction forces between blocks and determine whether it is in static equilibrium. This section first reviews the feasibility conditions for a structurally sound model. Next, for structures that do not satisfy these conditions, we introduce a *measure of infeasibility* that determines how far a structure is from a stable configuration. This will be used later in the parameter search.

3.1 Background: Static Analysis

To be physically feasible, the forces in a structure must satisfy static equilibrium, friction constraints, and additional constraints dependent on the material.

Contact Forces We model structures as an assemblage of rigid blocks, and analyze the force distributions at the interfaces between adjacent elements. Figure 5 illustrates the contact surface discretization. A three-dimensional force $\mathbf{f}^i$ is positioned at each vertex of the interface, modeling a linear force distribution across the

surface. Although three contact points could model the force distribution, we chose this representation for simpler constraint specification.

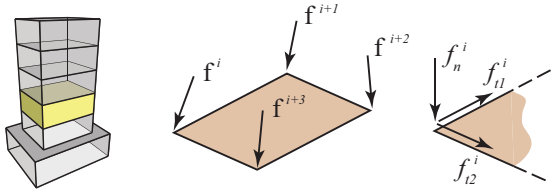


Figure 5: *Model of contact forces at interfaces between blocks.*

We represent $\mathbf{f}^i$ in the local coordinate system of the interface: one axial component f_n^i perpendicular to the face, and two orthogonal in-plane friction components, $f_{t_1}^i$ and $f_{t_2}^i$. The direction of in-plane friction forces is determined independently at each block interface, with t_1 aligned to an (arbitrary) edge of the block face. Friction forces on shared faces have opposite orientation.

Static Equilibrium Static equilibrium conditions require that net force and net torque for each block equal zero, accounting for self weight of the structure and external applied loads. Combining equilibrium constraints for each block gives a linear system of equations [Livesley 1978]:

$$\mathbf{A}_{eq} \cdot \mathbf{f} + \mathbf{w} = \mathbf{0} \tag{1}$$

where $\mathbf{w}$ is a vector containing the weights of each block, $\mathbf{f}$ is the vector of interface forces, and $\mathbf{A}_{eq}$ is the matrix of coefficients for the equilibrium equations (see Appendix). The system has 6 rows per building block, 3 for the 3 components of the net force, and 3 for the net torque. In general, structures have a sparse $\mathbf{A}_{eq}$ matrix due to the small number of interactions between blocks: each interface force in $\mathbf{f}$ affects only the two blocks adjacent to that interface, and a column of $\mathbf{A}_{eq}$ only has two non-zero coefficients.

Compression Constraint The compressive stresses in traditional structures are typically low relative to the strength of masonry and the material can be treated as rigid. Second, according to limit analysis of masonry as summarized by Heyman [1995], the material can be assumed to have zero tensile strength. Although mortar is used to fill interstices, it is relatively weak and is not assumed to add strength to the construction. This condition is expressed as a non-negativity constraint on the axial forces:

$$f_n^i \geq 0, \quad \forall\, i \in \text{ interface vertices} \tag{2}$$

Friction Constraints A friction constraint is applied at all vertices of the block interfaces. For each triplet of forces $\{f_n^i, f_{t_1}^i, f_{t_2}^i\}$ the two in-plane forces are constrained within the friction cone of the normal force f_n. To linearize, we approximate as a friction pyramid:

$$|f_{t_1}^i|, |f_{t_2}^i| \leq \alpha f_n^i, \quad \forall\, i \in \text{ interface vertices} \tag{3}$$

where α is the coefficient of static friction with a typical value of 0.7. As long as the per vertex friction forces satisfy the friction cone constraint, the resultant friction force over the interface is guaranteed to satisfy the constraint. The approximation is made conservative by using a reduced friction coefficient $(1/\sqrt{2})$ such that the cone circumscribes the pyramid. Alternatively, one could use an octagonal pyramid that more closely approximates the cone; however this would double the number of constraints which increases computation time.

Combining friction constraints over the entire assemblage of blocks in the structure gives a sparse linear system of inequalities:

$$\mathbf{A}_{fr} \cdot \mathbf{f} \leq \mathbf{0}$$

The friction constraint may not ensure feasibility in all cases, see the limitations section for details.

In summary, for a structure to stand in equilibrium, a force solution $\mathbf{f}$ must exist that satisfies the described linear constraints:

$$\begin{array}{rcll} \mathbf{A}_{eq} \cdot \mathbf{f} & = & -\mathbf{w} & \texttt{\textbackslash\textbackslash equilibrium} \\ \mathbf{A}_{fr} \cdot \mathbf{f} & \leq & \mathbf{0} & \texttt{\textbackslash\textbackslash friction} \\ f_n^i & \geq & 0, \quad \forall\, i \in \text{ interface vertices} & \\ & & & \texttt{\textbackslash\textbackslash compression-only} \end{array} \tag{4}$$

3.2 Measure of Infeasibility

We introduce a new formulation to analyze a model's geometry and measure its closeness to a feasible structure. The core problem we solve is that the constraints in (4) provide only a yes/no answer on stability. The unknowns, $\mathbf{f}$, can be solved using linear programming [Livesley 1978], provided that a feasible solution exists. However, if the structure is infeasible, no solution exists and no information is given on how far the structure is from a stable configuration.

We introduce a method to measure a structure's infeasibility by translating (4) into a penalty form. Our penalty formulation softens the compression constraint, which allows tension forces to act as "glue" at block interfaces to hold the structure together (e.g. Figure 6). We penalize the tension forces, and use their magnitude to measure the distance to a feasible solution. The first step is to express axial forces in terms of compression and tension using a variable transformation. The axial force at each vertex is written as the difference of two nonnegative variables [Bertsimas and Tsitsiklis 1997]:

$$\begin{array}{c} f_n^i = f_n^{i+} - f_n^{i-} \\ f_n^{i+}, f_n^{i-} \geq 0 \end{array} \tag{5}$$

where f_n^{i+}, f_n^{i-} are the positive and negative parts of f_n^i. The force f_n^i can take on any real value by choosing appropriate values for f_n^{i+} and f_n^{i-}. f_n^{i-} represent tension forces, and f_n^{i+} compression. Our penalty formulation of (4) is then a quadratic program:

$$\begin{array}{rll} y(\boldsymbol{\theta}) = & min_{\mathbf{f}} & \sum_{i=0}^{n} (f_n^{i-})^2 \\ & s.t. & \mathbf{A}_{eq} \cdot \mathbf{f} = -\mathbf{w} \\ & & \mathbf{A}_{fr} \cdot \mathbf{f} \leq \mathbf{0} \\ & & f_n^{i+}, f_n^{i-} \geq 0, \quad \forall i \end{array} \tag{6}$$

where the objective function is the squared norm of the tensile forces, and $y(\boldsymbol{\theta})$ is the measure of distance to a feasible structure. We choose a quadratic objective for smoothness of the resulting energy landscape when we vary the parameters of the procedural model in the structure optimization.

From a structural mechanics viewpoint, the constraints in (6) describe a statically indeterminate structure. Static equilibrium conditions do not specify a unique set of forces, rather, there are many possible solutions. We make the system well-posed by searching for the solution that is closest to satisfying material compression constraints.

Figure 8 illustrates the infeasibility measure for a two parameter system consisting of a semi-circular arch with free thickness, supported on two columns with free width. The ridge along the column

width axis illustrates that for arbitrarily large columns (effectively acting as ground), the arch has local feasibility limits on thickness.

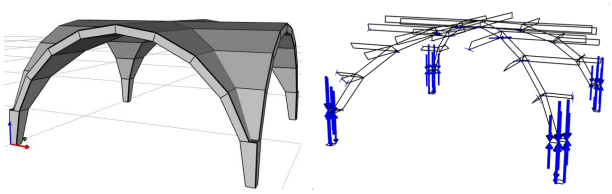


Figure 6: *Result of the quadratic program for an infeasibly thin groin vault. The minimum tension solution places tension forces (blue arrows) around the base of the vault to counteract the outward push from accumulated weight of the blocks.*

3.3 Validation

We compared our results from the quadratic program to known feasibility limits for semi-circular masonry arches. As shown by Milankovitch [1907], the minimum feasible thickness of an arch is 0.1075 of the average (centerline) radius. Our results were consistent giving a minimum thickness/radius ratio of 0.10746. We used a 100-block tessellation to approximate a continuous arch. A second validation test measured the maximum angle of ground tilt before an arch becomes infeasible. For an arch with 0.20 thickness/radius ratio, the critical tilt angle is 15.84 degrees [Ochsendorf 2002]. Our results match this value exactly for a 100-block arch. Arches at varying thicknesses were tested with similar accuracy. Note, however, that their results were obtained using 2D analysis while our method handles fully three-dimensional structures.

3.4 Robustness

The constraints in (6) describe the minimum requirements for a structure to support its own weight. In order to give the structure robustness to external perturbations, we incorporate a *geometric factor of safety* using the concept of the *kern*. The kern is the central portion of the interface where, if the resultant axial force lies within this region, the entire interface will act in compression. For rectangular sections the kern is the middle third [Heyman 1995]. When the resultant force lies outside of the kern, the compressive force is concentrated over a smaller effective interface. The structure forms a hinge when the resultant force reaches the boundary of the interface. We incorporate the kern limit by shrinking the boundaries of the contact polygon (Figure 7), and applying the interface forces at the modified vertex positions, which provides a margin of safety.

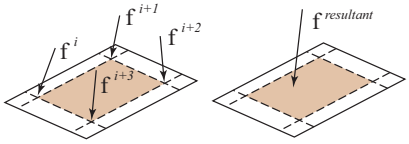


Figure 7: *The resultant force (right) has the equivalent net force and torque contribution to all vertex forces. For compression-only solutions the resultant must lie inside the interface boundaries. The geometric factor of safety shrinks the effective interface (orange) to tighten the compression constraint.*

Another criterion for robustness is to incorporate live loads, which are typically more critical for e.g. bridges, but less important for buildings such as cathedrals where the self-weight governs stability. The live load criterion is straightforward to use in our approach by modifying the external forces at any block of the building, which simply translates to adding the load to **w** in eq. (1).

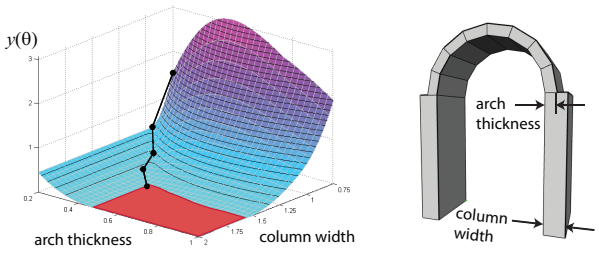


Figure 8: *Energy landscape for a two-parameter structure: circular arch supported on columns. The feasible region is the zero plane highlighted in red.*

4 Parameter Search

The parameter search determines parameter adjustments that reduce the instability of the building. We apply an iterative optimization technique, with progress measured by the energy $y(\boldsymbol{\theta})$ from the previous analysis step.

There can be a nonlinear relationship between the free rule parameters $\boldsymbol{\theta}$, and the measure of infeasibility. In order to search over the parameter space we use gradient-based nonlinear optimization in conventional form:

$$\begin{aligned} &argmin_{\boldsymbol{\theta}} \quad y(\boldsymbol{\theta}) \\ &s.t. \quad lb \leq \boldsymbol{\theta} \leq ub \end{aligned} \tag{7}$$

where $y(\boldsymbol{\theta})$ is the infeasibility metric from expression (6). Figure 8 shows the energy landscape for a two-parameter structure and the corresponding optimization path. The energy function is C^1 continuous due to the quadratic objective function, but may have a discontinuity of the second derivative when penalty forces, f_n^{i-}, become inactive.

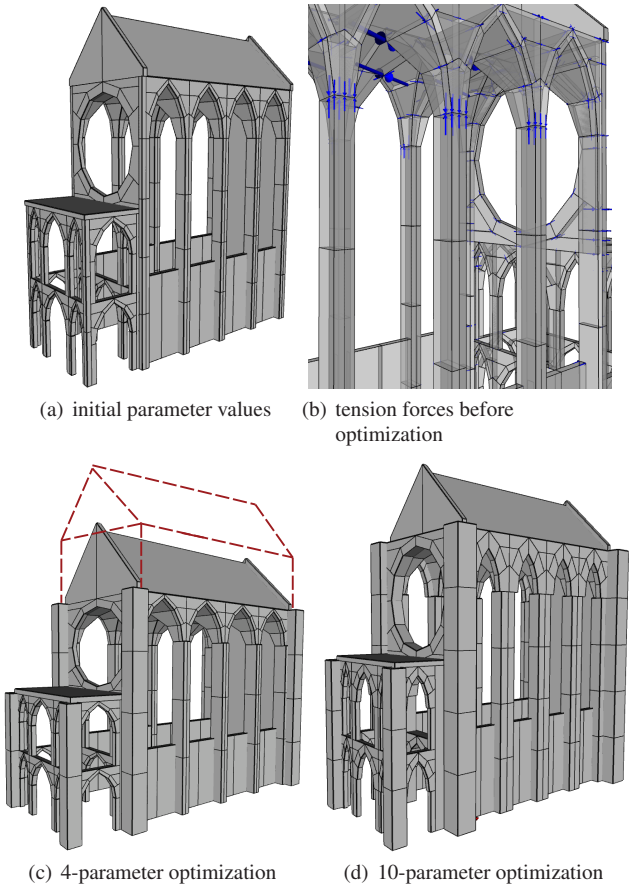
At each iteration of the optimization, the geometry of the procedural model is updated according to parameter values θ^i, we then use forward analysis to measure progress towards a feasible structure as described in section 3.

In contrast to forward statics where forces are determined based on fixed geometry, this step determines new geometry that satisfies constraints on the forces.

5 Results

Implementation We use the BPMPD interior point solver for quadratic programming [Mészáros 1996]. For nonlinear optimization, we use Matlab's active-set algorithm, based on a sequential quadratic programming method [Gill et al. 1981]. Gradients are estimated using forward finite differences.

Modeling stable buildings Figure 1 depicts a building model inspired by Cluny Abbey in France. The user has set up a grammar describing the placement of structural elements, and overall look of the building. The user then selects a set of free parameters, including the wall and buttress thickness, and the width of the windows. Our approach automatically finds the parameter values that satisfy structural stability constraints.



(a) initial parameter values (b) tension forces before optimization

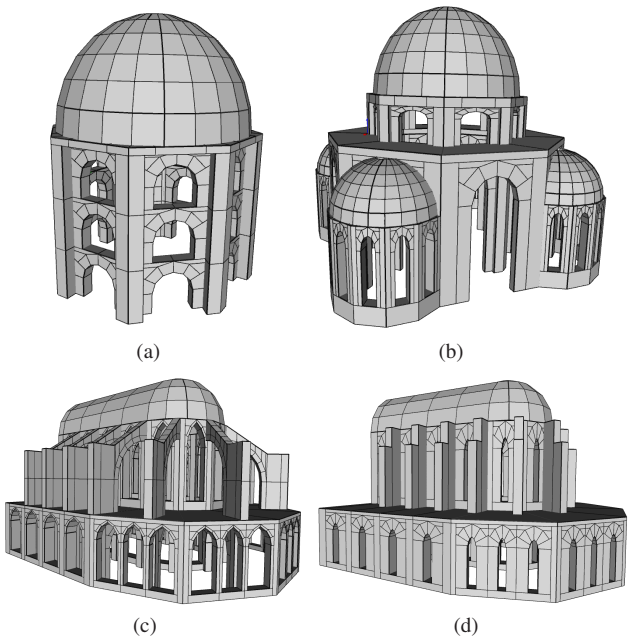
(c) 4-parameter optimization (d) 10-parameter optimization

Figure 9: *In this model inspired by the Sainte Chapelle in Paris, France, four parameters were optimized in (c) for column depth, corner thickness of the main hall and entranceway, and overall height of the building. By freeing additional parameters (d), a feasible model is possible without decreasing the height.*

The user controls which parameters are free variables and can affect the structural tradeoffs that yield a sound shape. The optimization in Figure 9(c) automatically adjusts 4 parameters (column dimensions) and reduces the overall height of the building to reach stability. In comparison, the 10-parameter optimization (Figure 9(d)) finds a feasible solution at the original building height in exchange for smaller windows and thicker walls. Figure 10 shows a variety of structural models achieved by making modifications to the grammar. The optimized model in Figure 10(d) has small windows to support the domed ceiling. Extending the grammar to include flying buttresses (Figure 10(c)) transfers the load away from the walls, allowing for larger windows for increased natural light in the interior.

The tower in Figure 11 is an example structure where it may be difficult to judge stability by intuition alone. A feasible structure was generated with a 32-parameter optimization that adjusted the horizontal position of each level individually.

In Figure 12 the shape of the arches is optimized. We use a cubic Bezier curve: the first and last control points are fixed at the base, while the two inner control points (red dots) are variable. The z-coordinate of the arch is scaled to maintain a constant height and to maintain contact between adjacent shapes. The free parameters control the horizontal position of the two inner control points. In the original configuration (Figure 12, left) the arches are too thin



(a) (b) (c) (d)

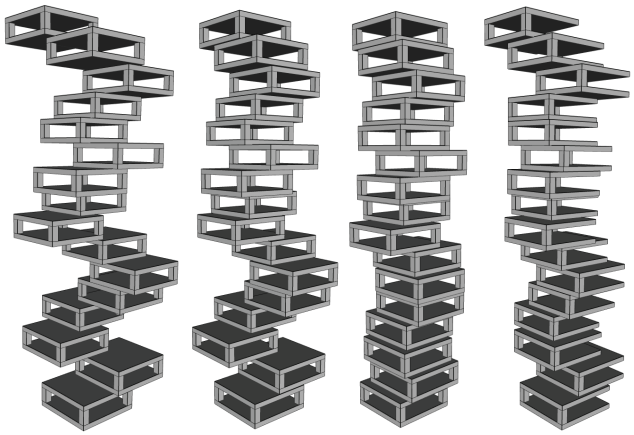
Figure 10: *Image (a) shows a mosque generated with procedural rules, and optimized for feasibility. Image (b) was generated by extending the grammar. In (c) radial flying buttresses are optimized, while (d) shows that smaller windows are required when the buttresses are removed from the grammar.*

to stand. The parameter search generates arches resembling catenaries (Figure 12, right) which provides feasibility without increasing block thicknesses. Note that the two lower arches are slightly skewed to account for outward forces transferred from the top arch.

Performance Table 1 shows performance and convergence results for a few representative examples. The most expensive step in the pipeline is the quadratic program which evaluates the energy function. The total cost is $n_{it} n_{\boldsymbol{\theta}}\, complexity$(BPMPD solver), where n_{it} is the number of iterations in the parameter search, and $n_{\boldsymbol{\theta}}$ is the number of free parameters. The linearity in the number of parameters is due to our use of finite differences for gradient computations.

Table 1: *Performance*

model	blocks	parameters	iterations	time/iter.
Cluny (Fig.1)	986	4	10	45.7 s
		5	5	57.3 s
		7	4	70.0 s
		9	9	106.6 s
arch (Fig.8)	10	2	6	0.1 s
Sainte Chapelle (Fig.9)	486	3	4	12.5 s
		5	9	26.5 s
		7	6	29.3 s
		10	8	40.1 s
tower (Fig.11)	96	32	6	12.5 s
barrel vault (Fig.13)	140	1	8	0.6 s



(a) infeasible stacking (b) 32-parameter optimization (c) geometric factor of safety (d) modified grammar

Figure 11: *From the original tower input (a), 32 parameters were optimized to create a feasible stacking arrangement (b). The position of each level was controlled by two separate parameters for translations in x and y. Variations can be found using different initial positions. In (c) the interfaces were shrunk by 0.5 along each edge as a geometric factor of safety. In (d), modifying the grammar provides further variation.*

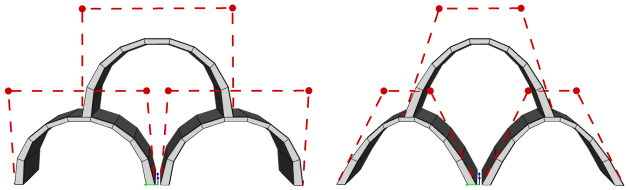


Figure 12: *The arch profiles are defined using Bezier curves. A 6-parameter optimization controls the horizontal position of the two inner control points (red dots) for each arch. (Left) In the initial configuration the circular shape is infeasible. (Right) The feasible result. The bottom arches are slightly asymmetrical to account for horizontal forces transferred from the top arch.*

Editing Parameters Interactive editing of parameters is another valuable usage scenario when speed permits. The user may manually edit a model while our system automatically updates the free parameters to maintain structural feasibility. For example, in Figure 13, as the user increases the span of the vaulted ceiling, the angle of the buttresses is modified to account for increasing horizontal forces. In our prototype, the result updates under five seconds for this model. Changes in a design can alter the loads on many other parts of the structure. Traditionally, a change in one element requires tweaks to many other dependent elements of the model. We simplify the task of exploring design variations by automatically identifying and updating dependent design parameters.

Dynamic simulations Physically feasible models make it possible to run dynamic simulations in interactive environments. Under no external forces, feasible models will stand in static equilibrium. Users may then apply effects such as earthquakes and collisions and the model will exhibit realistic dynamic behavior. As a proof of concept for dynamics applications, we generated simulations using the Bullet open-source rigid-body dynamics library [Coumans 2008]. Figure 14(a) shows a structure reacting to perturbations of the ground plane. The perturbation was generated by applying a lateral impulse to the centroid of the ground plane, causing a change in ground velocity of 4 m/s over a time step of 1/60 s. The model is approximately 10m wide. In Bullet, the restitution value (bounciness) was set to the default value of 0.0, and the friction coefficient was set to 0.895.

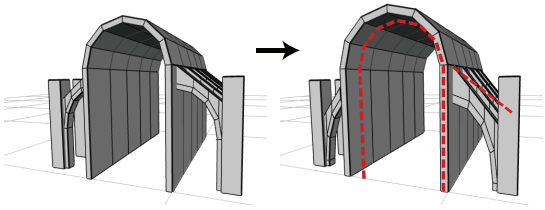
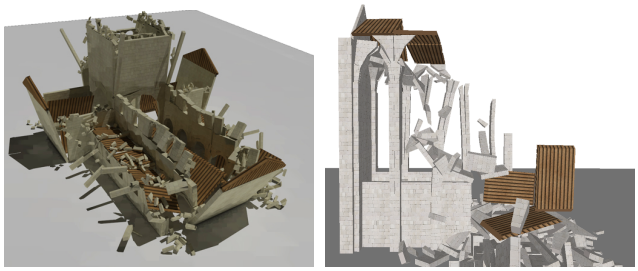


Figure 13: *Interactive editing of parameters. As the user increases the span of the barrel vault roof, our system automatically selects the angle of the flying buttresses required to maintain stability. Red lines highlight the original structure.*



(a) (b)

Figure 14: *Structurally sound models can be manipulated in physically simulated environments. (a) The Cluny model collapses after a ground shake is applied; (b) the Sainte Chapelle model collapses after a central column is broken (see supplementary video).*

Friction Cone Approximation To test the effect of the friction pyramid parameterization on feasibility, we performed parameter searches on the Sainte Chapelle model (Figure 9) with the friction pyramid rotated 45 degrees. In a 3-parameter search we found that the corner columns of the main hall were 10.5% thinner than with original friction pyramid. In a 4-parameter search, the thickness of the arches in the windows was 4.3% smaller with the 45 degree rotated pyramid. Alternatively, the columns underneath the windows were only 1% thinner.

Block Size The number of blocks should match that used in the final simulation. Using fewer (i.e. larger) blocks as an approximation may over-estimate the stability of the final structure in some cases, e.g. for arches, vaults and flying buttresses, where hinging failure mechanisms may occur. The tradeoff between accuracy and computation speed is an area for future investigation.

We tested the Sainte Chapelle model (Figure 9) by choosing parameters that were "just stable" (small variations make the structure unstable), then varied the number of blocks. The chapel remained stable when we increased the block count from 486 to 876 by subdividing the columns, arched windows, and circular window. The chapel became unstable when we further subdivided the groin vaulted ceiling.

Local Minima MATLAB's active-set algorithm does not guarantee convergence to the global minimum. If local minima exist, it is

possible a local minimum will be the result of the parameter search. We have not encountered this problem in the provided examples – all models converged to the global minimum (zero tension) solution without any aids, e.g. multiple starting points. We tested the tower model in Figure 11 by running the 32-parameter search from randomly generated starting points and all converged to a zero-tension solution.

Limitations A feasible structure does not always exist within the given set of rules and free parameters. Under these circumstances, we return the structure within minimum infeasibility, and the user is required to manually add new structural elements. For experienced users, visualization of the tension forces provides guidance for altering the design specifications.

We do not consider the "sawtooth" friction case described by Gilbert et al. [2006]; we assume idealized interfaces where only tangential displacement would occur. Our method identifies structures where a feasible equilibrium solution exists. However, the resulting structure may still be unstable if alternative equilibrium states exist where friction constraints are violated.

Our approach applies to masonry buildings which are rigid block compression-only structures. We can trivially handle structures where pairs of blocks interact in tension-only by flipping the sign of the compression constraint.

6 Discussion

We have introduced structural soundness as a key objective in procedural modeling of buildings. To achieve this, we have addressed the limitations of current engineering analysis tools based on elastic theory and have instead relied on a quadratic programming formulation of the equilibrium equations. Comparisons with available data validates the accuracy of the technique. We have introduced a penalty form that allows us to measure the degree of infeasibility of a structure, which can be used in a search procedure to yield a stable building. A variety of stable structures can be created and the user can decide which parameters are fixed in order to control the structural tradeoffs.

Feasible models are valuable for virtual environments to allow users to interact physically with built surroundings, and simulate realistic dynamics such as collapse under collision or earthquakes. These interactions are not possible unless a structure is capable of standing under self-weight. Our method makes it easy for users to create feasible buildings, letting the optimization take care of the complex equilibrium conditions. We believe that inverse statics techniques such as the one we developed have tremendous potential beyond interactive virtual environments. We are excited about applications in historical education and architectural design. Procedural grammars can encapsulate families of buildings, such as the Romanesque churches of a particular region, creating a useful interface for analyzing existing historic architecture. Furthermore, our approach can be applied to designing buildings with other materials, as shapes which act predominantly in axial stress rather than bending are less prone to deformation.

Acknowledgements

Phillippe Siclait implemented the dynamics simulations in Bullet. Sylvain Paris and Yeuhi Abe provided helpful discussions. Jovan Popović suggested the use of procedural modeling. Thanks to reviewers of the MIT pre-deadline. This work was supported by funding from NSERC Canada and grants from the Singapore-MIT Gambit Game Lab.

References

BERTSIMAS, D., AND TSITSIKLIS, J. N. 1997. *Introduction to Linear Optimization*. Athena Scientific.

BLOCK, P., CIBLAC, T., AND OCHSENDORF, J. 2006. Real-time limit analysis of vaulted masonry buildings. *Computers & Structures 84*, 29–30, 1841–1852.

COUMANS, E., 2008. Bullet: Collision detection and rigid body dynamics library. Available at http://bulletphysics.com.

GILBERT, M., CASAPULLA, C., AND AHMED, H. 2006. Limit analysis of masonry block structures with non-associative frictional joints using linear programming. *Computers and Structures 84*, 873–887.

GILBERT, M. 2001. RING: a 2D rigid-block analysis program for masonry arch bridges. In *ARCH01: Third International Arch Bridges Conference*, 459–464.

GILL, P. E., MURRAY, W., AND WRIGHT, M. 1981. *Practical Optimization*. Academic Press, London.

HARADA, M., WITKIN, A., AND BARAFF, D. 1995. Interactive physically-based manipulation of discrete/continuous models. In *Proceedings of SIGGRAPH 95*, ACM Press / ACM SIGGRAPH, R. Cook, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 199–208.

HART, J. C., BAKER, B., AND MICHAELRAJ, J. 2003. Structural simulation of tree growth and response. *The Visual Computer 19*, 2-3, 151–163.

HEYMAN, J. 1995. *The Stone Skeleton: Structural Engineering of Masonry Architecture*. Cambridge University Press.

LIPP, M., WONKA, P., AND WIMMER, M. 2008. Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics 27*, 3, 102.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.-L., AND WANG, W. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graphics 25*, 3, 681–689. Proc. SIGGRAPH.

LIVESLEY, R. K. 1978. Limit analysis of structures formed from rigid blocks. *International Journal for Numerical Methods in Engineering 12*, 1853–1871.

LIVESLEY, R. K. 1992. A computational model for the limit analysis of three-dimensional masonry structures. *Meccanica 27*, 3, 161–172.

LOURENCO, P. 2002. Computations on historic masonry structures. *Progress in Structural Engineering and Materials 4*, 3, 301–319.

MÉSZÁROS, C. 1996. Fast cholesky factorization for interior point methods of linear programming. *Computers & Mathematics with Applications 31*, 4-5, 49–54. Selected Topics in Numerical Methods.

MILANKOVITCH, M. 1907. Theorie der druckkurven. *Zeitschrift fr Mathematik und Physik 55*, 1–27.

MÜLLER, P., WONKA, P., HAEGLER, S., ULMER, A., AND GOOL, L. V. 2006. Procedural modeling of buildings. *ACM Transactions on Graphics 25*, 3, 614–623.

MÜLLER, P., ZENG, G., WONKA, P., AND GOOL, L. V. 2007. Image-based procedural modeling of facades. *ACM Transactions on Graphics 26*, 3, 85.

OCHSENDORF, J. 2002. *Collapse of Masonry Structures*. PhD thesis, University of Cambridge.

PARISH, Y. I. H., AND MÜLLER, P. 2001. Procedural modeling of cities. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, E. Fiume, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 301–308.

POTTMANN, H., LIU, Y., WALLNER, J., BOBENKO, A., AND WANG, W. 2007. Geometry of multi-layer freeform structures for architecture. *ACM Transactions on Graphics 26*, 3, 65.

POTTMANN, H., SCHIFTNER, A., BO, P., SCHMIEDHOFER, H., WANG, W., BALDASSINI, N., AND WALLNER, J. 2008. Freeform surfaces from single curved panels. *ACM Transactions on Graphics 27*, 3, 76.

SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., AND GUO, B. 2007. Mesh puppetry: cascading optimization of mesh deformation with inverse kinematics. *ACM Transactions on Graphics 26*, 3, 81.

SMITH, J., HODGINS, J. K., OPPENHEIM, I., AND WITKIN, A. 2002. Creating models of truss structures with optimization. In *Proceedings of SIGGRAPH 2002*, ACM Press / ACM SIGGRAPH, J. Hughes, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM, 295–301.

WELCH, W., AND WITKIN, A. 1992. Variational surface modeling. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, ACM, 157–166.

WONKA, P., WIMMER, M., SILLION, F., AND RIBARSKY, W. 2003. Instant architecture. *ACM Transactions on Graphics 22*, 3 (July), 669–677.

ZIENKIEWICZ, O. C. 1971. *The Finite Element Method in Engineering Science*. McGraw-Hill, London.

Appendix

We detail the matrix equation for static equilibrium (1). We use an example construction for an arch consisting of n blocks. Note that the $\mathbf{f}$ vector has $n+1$ contact surfaces because there are $n-1$ shared interfaces between blocks in the arch, and two interfaces in contact with the ground plane.

$$\mathbf{A}_{eq} \cdot \mathbf{f} + \mathbf{w} = \mathbf{0}$$

$$\begin{bmatrix} A_{0,0} & A_{0,1} & & & \\ & A_{1,1} & A_{1,2} & & \\ & & \ddots & & \\ & & A_{n-1,n-1} & A_{n-1,n} \end{bmatrix} \begin{bmatrix} \mathbf{r}_0 \\ \vdots \\ \mathbf{r}_n \end{bmatrix} + \begin{bmatrix} \mathbf{w}_0 \\ \vdots \\ \mathbf{w}_{n-1} \end{bmatrix} = \mathbf{0}$$

$\mathbf{w}_j$: 6×1 vector containing the 3D weight and net torque for block j. Typically the only non-zero element is the z-component of weight. For any external loads acting on block j, the force and torque contributions are added here.

$\mathbf{r}_k$: Contains the unknown force vectors $\mathbf{f}^i$, for vertices i on interface k. $height(\mathbf{r}_k)$ is $3v_k$, where v_k is the number of vertices on interface k and each vertex contributes a 3D force. Note that after decomposing the axial forces into positive and negative parts (eq. 5), the dimension of $\mathbf{f}^i$ increases to 4×1 which changes the $height(\mathbf{r}_k)$ to $4v_k$.

$\mathbf{A}_{j,k}$: Submatrices $\mathbf{A}_{j,k}$ contain coefficients for net force and net torque contributions from interface k acting on block j. Each $\mathbf{A}_{j,k}$ has dimension $6 \times height(\mathbf{r}_k)$. Rows 1-3 are coefficients for net force contributions in x, y, z and rows 4-6 are coefficients for net torque contributions about the x, y, z axes.

$$\mathbf{A}_{j,k}\mathbf{r}_k = \begin{bmatrix} \mathbf{a}_{k_x} & \mathbf{a}_{k_x} & \cdots \\ \mathbf{a}_{k_y} & \mathbf{a}_{k_y} & \cdots \\ \mathbf{a}_{k_z} & \mathbf{a}_{k_z} & \cdots \\ \mathbf{b}_{i,j,k_x} & \mathbf{b}_{i+1,j,k_x} & \cdots \\ \mathbf{b}_{i,j,k_y} & \mathbf{b}_{i+1,j,k_y} & \cdots \\ \mathbf{b}_{i,j,k_z} & \mathbf{b}_{i+1,j,k_z} & \cdots \end{bmatrix} \begin{bmatrix} \mathbf{f}^i \\ \mathbf{f}^{i+1} \\ \vdots \end{bmatrix}$$

where $\mathbf{f}^i = [f_n^i \ f_{t1}^i \ f_{t2}^i]^T$, $\quad \mathbf{a}_{k_x} = [\hat{\mathbf{n}}_{k_x} \ \hat{\mathbf{t}}_{k_{1x}} \ \hat{\mathbf{t}}_{k_{2x}}]$

and $\mathbf{b}_{i,j,k_x} = [(\hat{\mathbf{n}}_k \times \hat{\mathbf{v}}_{i,j})_x \ (\hat{\mathbf{t}}_{k_1} \times \hat{\mathbf{v}}_{i,j})_x \ (\hat{\mathbf{t}}_{k_2} \times \hat{\mathbf{v}}_{i,j})_x]$.

$\hat{\mathbf{n}}_k$, $\hat{\mathbf{t}}_{k_1}$ and $\hat{\mathbf{t}}_{k_2}$ are the normal vector and friction basis vectors for interface k (see Figure 15). The subscript $_x$ refers to the x-component of the vector.

The number of submatrices $\mathbf{A}_{j,k}$ in row j of $\mathbf{A}_{eq}$ is equal to the number of neighbors incident on block j. There are two submatrices in each column k, since $\mathbf{r}_k$ represents the interaction between surfaces of two adjacent blocks.

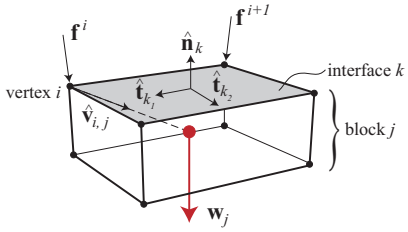


Figure 15: *Indexing for equations of static equilibrium. Vector $\hat{\mathbf{n}}_k$ is the unit normal for interface k, and $\hat{\mathbf{t}}_{k_1}$ and $\hat{\mathbf{t}}_{k_2}$ are the directions of in-plane friction forces. Unit vector $\hat{\mathbf{v}}_{i,j}$ is the relative position of vertex i w.r.t. the centroid of block j. $\mathbf{w}_j$ is the 3D weight vector for block j.*

Size Complexity The sizes of the constraint matrices for static equilibrium and friction are as follows:

$\mathbf{f}$: $length = \sum_k v_k (\#forces\,per\,vertex)$, over all interfaces k in the structure. v_k is the number of vertices on interface k.

$\mathbf{A}_{eq}$: $size = 6(\#blocks) \times length(\mathbf{f})$. The number of non-zero elements in $\mathbf{A}_{eq}$ is 12 per column, since there are 6 equilibrium equations and 2 interacting blocks per interface. The number of non-zero elements in each row j is $\sum_k v_k (\#forces\,per\,vertex)$, over all interfaces k on block j.

$\mathbf{A}_{fr}$: square with dimension = $length(\mathbf{f})$. The number of non-zero elements in $\mathbf{A}_{fr}$ is $\sum_k 8v_k$ over all interfaces k in the structure, assuming a 4-sided friction pyramid.