Design Of Structurally-Sound Masonry Buildings Using 3D Static Analysis

by

Emily Jing Wei Whiting

B.A.Sc., University of Toronto (2004)
S.M., Massachusetts Institute of Technology (2006)

Submitted to the Department of Architecture
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Architecture: Building Technology

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2012

Author .....
Department of Architecture
November 1, 2011

Certified by .....
John Ochsendorf
Associate Professor of Architecture and Civil & Environmental Engineering
Thesis Supervisor

Certified by .....
Frédo Durand
Associate Professor of Electrical Engineering & Computer Science
Thesis Supervisor

Accepted by .....
Takehiko Nagakura
Associate Professor of Design and Computation
Chair of the Department Committee on Graduate Students

Thesis Supervisor .....

John Ochsendorf

Associate Professor of Architecture and Civil & Environmental Engineering

Thesis Supervisor .....

Frédo Durand

Associate Professor of Electrical Engineering & Computer Science

Thesis Reader .....

Terry Knight

Professor of Design and Computation

Design Of Structurally-Sound Masonry Buildings Using 3D Static Analysis

by

Emily Jing Wei Whiting

Submitted to the Department of Architecture
on November 1, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Architecture: Building Technology

Abstract

In the design of buildings, structural analysis is traditionally performed after the aesthetic design has been determined and has little influence on the overall form. This thesis presents methods to integrate architectural design and structural analysis. While existing tools focus on providing an analysis of the stress state, the proposed methods focus on geometry and equilibrium to obtain forms that are more structurally sound. The feasibility of masonry structures is modeled using a novel penalty formulation, assuming a rigid-block behavior of masonry. Two methods were developed that apply this model of feasibility to structural optimization.

In the first approach, structural feasibility is introduced into procedural modeling of buildings. A set of designated free parameters are automatically tuned to achieve structural feasibility constraints. It is demonstrated how this allows for more realistic structural models that can be interacted with in physical simulations.

In the second approach, a closed form derivation of structural gradients is presented that measures the change in stability of a building with respect to geometry modifications. The method computes the gradient of structural feasibility constraints, parameterized by vertex modifications. The gradients are visualized as interaction tools, giving user-guidance for effectively modifying a structural design. User-controlled constraints, formulated as penalty functions, are incorporated so that the user can explore variations of structurally feasible designs.

Thesis Supervisor: John Ochsendorf
Title: Associate Professor of Architecture and Civil & Environmental Engineering

Thesis Supervisor: Frédo Durand
Title: Associate Professor of Electrical Engineering & Computer Science

Acknowledgments

I would like to thank my PhD Co-Advisors Frédo Durand and John Ochsendorf for many things: In this unusual blend of structural engineering and computer graphics they recognized and encouraged the opportunity for original research. Without their unique insights on all problems forceful, graphical, and optimizable I would not have a thesis. And their unending optimism was never undervalued.

Sincere thanks to Terry Knight for acting on my Dissertation committee, and especially for being one of my first sources of encouragement to attend MIT as a graduate student.

I gratefully acknowledge my sources of funding: the Singapore-MIT Gambit Game Lab, the NSERC Canada PGS-D Scholarship, and the MIT Presidential Fellowship.

I would like to acknowledge the fantastic work of my collaborators: Robert Wang developed and implemented the planar quad mesh constraints and user constraints in chapter 5. Phillippe Siclait was a highly valued UROP who ran the dynamics experiments in chapter 4 and contributed PBRT renderings. Jason Boggess implemented the selection UI in chapter 5.

An enormous thank you goes to my colleagues in the MIT Computer Graphics Group: Sylvain for helpful discussions, editing, and encouragement, Jovan for suggesting procedural modeling, Yeuhi for helpful discussions, Jiawen for his treasure hunting skills, Bryt as an endless resource, and Eugene for his fast forward ingenuity. Thank you to the entire group for their camaraderie over the years: Abe, Alec, Bennett, Daniel, Forrester, Ilya, Jaakko, Jrk, Marco, Paul, Sara, Soonmin, Tilke, Tom, Vlad, and Zhunping.

Thanks to my colleagues in the Building Technology Group: Philippe, Matt, Jenn, Rory and Kathleen, for their inspiration in all things masonry.

And finally, thank you to my parents for their patience and encouragement.

Contents

List of Tables

Chapter 1

Introduction

1.1 Motivation

Design informed by stability is the notion behind such structural landmarks as Kings College Chapel (Figure 1-1) or the work of Catalan architect Antoni Gaudi (1852-1926). These buildings are feats of engineering in the spans and heights they were able to achieve, and the elegant vaulted and arched forms were designed through an understanding of structural properties. The connection between shape and structure is illustrated in Figure 1-1(b): an unstable semi-circular arch is shown alongside its collapse mechanism. The second catenary-shaped arch, while having the same span and thickness, is structurally stable due to a better choice of shape. In this case, stability is a function of the geometry rather than failure of the material. This thesis is an investigation of the relationship between form and feasibility, with the goal of using computational techniques to find structurally stable forms.

While computer graphics and computer-aided-design (CAD) have dramatically broadened the range of shapes available for architectural design, structural considerations have often been ignored. Structural analysis of a building is usually performed after the aesthetic design has been determined and has little influence on the overall form. An architect designs the shape, which is passed to structural engineers to make the building stable, typically using support structures of steel and reinforced concrete. Existing structural analysis software, such as finite element analysis, is a

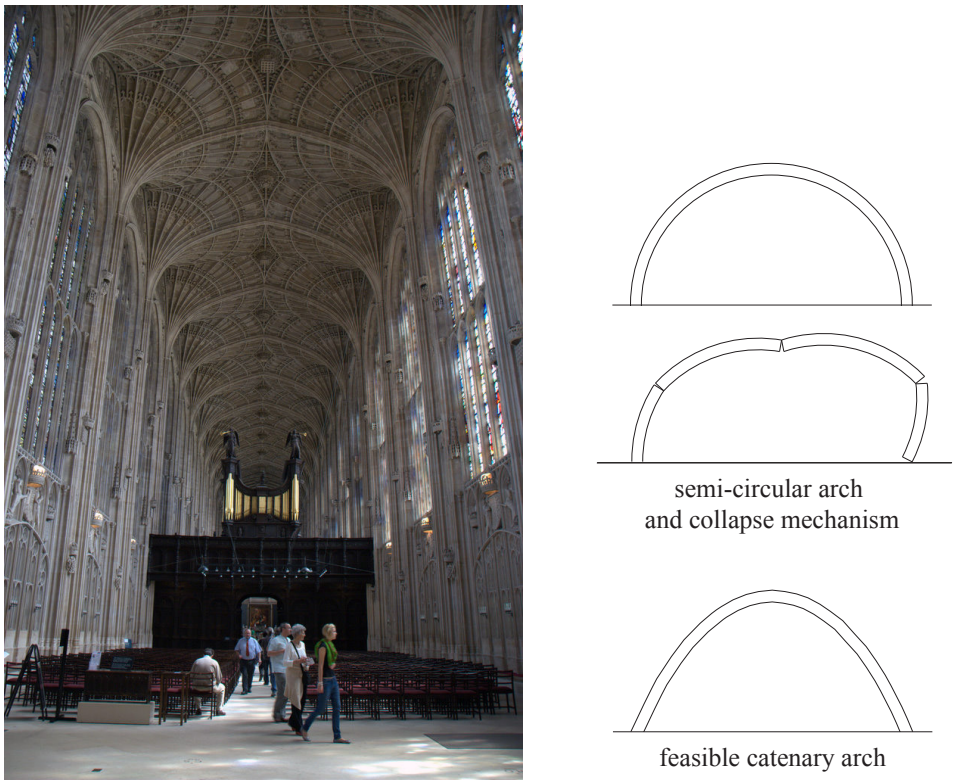


Figure 1-1: (a) King's College Chapel, Cambridge University, UK. (b) Comparison of arch structures where shape determines the feasibility rather than material strength. The semicircular arch is infeasible with a collapse mechanism as shown [web.mit.edu/masonry]. The modified catenary arch is feasible.

powerful method for analyzing a given structure, but does not directly suggest ways for the designer to improve the geometry in order to reduce the internal forces and required material. Designers may not have intuition about the mechanics that govern structural stability, and determining the precise dimensions of a structure that guarantee stability can be a tedious task.

This thesis focuses on masonry materials, comprising stone and brick structures. Masonry is the traditional material for historic architecture such as cathedrals and mosques, and is also used in modern architecture such as the museum in Mapungubwe, South Africa (Figure 1-2) where local soil was used to manufacture the bricks. In contrast to contemporary steel or reinforced concrete, traditional masonry relies on forms which are inherently stable, because the material resists only axial compressive

Figure 1-2: Example of modern masonry architecture: Interpretation Center, Mapungubwe, South Africa [Peter Rich Architects, Photo source: www.cretique.com].

forces. The structural quality of a building is derived from its shape. Though we focus on the case of masonry, our approach can be used to minimize non-axial forces in general. Even with materials that resist tension, such as reinforced concrete or steel, a good structural form with reduced non-axial force requires less material, leading to cheaper, more environmentally-friendly, and robust buildings.

Outside of architecture, there are motivations for physical models of buildings in computer graphics applications. Content creation for virtual environments has become a bottleneck – 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.

1.2 Problem Statement and Contributions

We seek to bring consideration of structural stability to computer-aided design for architecture. We term this problem *inverse statics*: starting from an infeasible design, the goal is to determine shape modifications that will satisfy structural stability constraints. In contrast to a forward analysis method that determines the force state for a set geometry, our approach generates new geometry that improves feasibility. This thesis makes three primary contributions toward this goal in the areas of masonry analysis and modeling of structurally feasible architectural geometry:

1. **Measure of Infeasibility** We present a measure of infeasibility that determines how close a model is to being structurally sound. We introduce a penalty formulation to represent the presence of tension in rigid block assemblages. It is enabled by a quadratic programming formulation and agrees closely with previous theoretical results.

2. **Procedural Modeling** We generate structurally feasible procedural models of buildings through automatic parameter selection. 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.

3. **Free-Form Modeling** We extend our structural optimization to handle free-form shapes by parameterizing the geometry at a vertex level. We provide a closed form derivation for the gradient of the measure of infeasibility with respect to geometry modification. The performance enhancement offered by the analytic formulation enables a variety of interactive tools. We present prototype design tools for improving the stability of a model based on user-provided constraints and objectives. We demonstrate that our technique can lead to a variety of designs given the same input shape, based on the user's decisions.

1.3 Overview

As stated in §1.2, our three key contributions consist of a masonry analysis method, an optimization approach to structural design using procedural modeling, and an extension to structural design for free-form structures. The contributions and supporting discussions are organized as follows.

Chapter 2 reviews previous research in structural engineering, architecture, and optimization to explain the groundwork that this thesis builds upon.

Chapter 3 focuses on a new forward analysis tool to assess the soundness of a

masonry structure. The method is based on stability of the geometric configuration and whether it is in static equilibrium. We extend 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 each element is subject to static equilibrium constraints. 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 nonlinear optimization to find the appropriate parameters.

Chapter 4 introduces physical constraints into procedural modeling methods. We apply the forward analysis from chapter 3 to search for parameters of our procedural model that yield a stable building. 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. The method automatically "snaps" to feasible dimensions, while leaving control in the designer's hands for deciding which aspects of the model are variable.

Chapter 5 presents a method to compute the gradient of stability for masonry structures, and demonstrates how the gradient can be a powerful guide in the design of stable structures. We describe a design interface that is parameterized directly on the vertex positions which gives complete freedom to the user for designing free-form shapes. Building upon the instability metric introduced in chapter 3, we provide a closed form derivation of the gradient with respect to geometry modification. Further, user-defined constraints and objectives are incorporated to allow the user to explore a variety of solutions and designs based on preference.

Finally in chapter 6 we discuss conclusions of the thesis and outline avenues for future work.

Chapter 2

Literature Review

2.1 Introduction

The contributions presented in this thesis are highly interdisciplinary, touching on a wide range of topics in structural engineering, masonry mechanics, shape optimization and architectural modeling. In this chapter, we focus on select literature directly relevant to our work. For a more general presentation, we recommend classical textbooks such as Heyman [1995] for masonry mechanics, Cook [1995] for finite element methods, and LaPorte and Le Tallec [2003] for shape optimization. The review is organized under the two main themes of *analysis* and *design*: section 2.2 discusses techniques in masonry analysis, focusing on the subtopics of strength versus stability methods. Section 2.3 reviews previous work in structural design methods.

2.2 Masonry Analysis

Masonry analysis can be broadly grouped into methods that consider material strength and those based solely on geometry. Strength methods assess structures according to material properties: the relationship between stress and strain, and the limits of compressive and tensile strength. Structural failure is determined by identifying sites of material damage where these limits are surpassed. In contrast, stability methods consider only the geometry of the structure. Feasibility is assessed based on condi-

tions of equilibrium and material failure is not considered. A comprehensive review of analysis techniques for historic masonry structures is given by Lourenco [2002].

2.2.1 Strength Methods

Finite element (FE) modeling is the predominant method used for structural analysis in many fields of engineering. FE methods work by dividing a structure into a system of discrete elements connected by nodes. Given a discretization, material properties and load conditions, FE analysis calculates nodal displacements, which are used to calculate strain and stress values of the structure. The accuracy of the analysis depends on a discretization that represents how stresses will be distributed in the structure and how it will deform. Further, shape functions with appropriate degrees of freedom must be assigned to the nodes [Zienkiewicz 1971; Cook 1995; Bathe 2006]. Given a discretization, the steps in carrying out a FE analysis are as follows:

- Generate the stiffness matrix $\mathbf{k}$ for each element, which describes the force-displacement relationship. The general formula for stiffness is $\mathbf{k} = \int \mathbf{B}^T\mathbf{E}\mathbf{B}\ dV$, where $\mathbf{B}$ is the strain-displacement matrix, $\mathbf{E}$ is the constitutive matrix of the material relating stress and strain, and V is volume.

- Connect the elements by assembling the $\mathbf{k}$ element matrices into the global $\mathbf{K}$ stiffness matrix.

- Construct the matrix of loads $\mathbf{R}$.

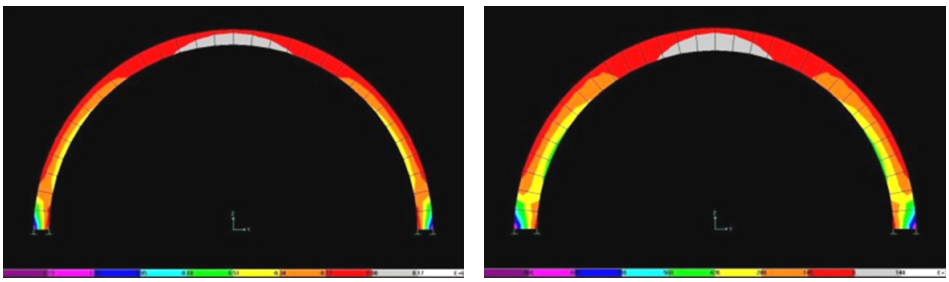
- Finally, solve the global equations $\mathbf{KD} = \mathbf{R}$ for the unknown displacements $\mathbf{D}$. The strains and stresses in the structure are computed directly from $\mathbf{D}$.

The advantage of FE analysis is that it is a general approach which can be applied to complex geometries that would be difficult or impossible to analyze using direct methods of structural mechanics (e.g. elementary formulas for beams or columns). As such, it is natural to attempt to apply FE analysis to the complex vaulted geometry of typical masonry structures. However there are a number of reasons linear elastic FE analysis has difficulty obtaining accurate and efficient results [DeJong 2009]:

- FE methods are tailored for a continuum, whereas masonry is inherently a discontinuous material: the joints between blocks are free to move, and failure within blocks is brittle fracture.

- Stresses are low in comparison to the compressive strength of masonry and elastic deformations are very small.

As an illustrative example, Figure 2-1 shows side-by-side results of a FE analysis for two arches with different thickness. The arches are modeled as a linear elastic continuum on fixed supports and are subject to self-weight only. In both cases tensile stresses are predicted, equivalent to bending moments at the supports [Block et al. 2006; Cook 1995]. In comparison, a static equilibrium analysis of the arches easily reveals that the thicker arch has a compression-only solution and is therefore feasible, while the thin arch will not stand unless the material resists significant tensile stress. Conventional linear-elastic analysis does not predict the infeasibility of the thinner arch.



(a) Thin arch: infeasible (b) Thick arch: feasible

Figure 2-1: Linear elastic FE analysis fails to differentiate between (a) an infeasible arch and (b) a feasible arch. Tension stresses appear in both analyses (green-purple) despite there being a compression-only equilibrium solution for the thicker arch. Reprinted from Block et al. [2006].

Further, FE methods do not reflect the *hyperstatic* properties – masonry structures are indeterminate, meaning an infinite number of equilibrium solutions exist. In FE analysis a single stress state is determined which may not be representative of the actual stresses in the structure.

Two examples of finite element methods applied to analysis of masonry vaults are Mark et al. [1973] and Milani et al. [2008]. In Mark et al., their physical test structure is constructed from epoxy which has a low modulus of elasticity and is a continuous shell. This differs from traditional masonry construction which is composed of discrete blocks and has a rigid block assumption (no elasticity). Beyond material differences, Mark et al. describe challenges in specifying boundary conditions and mesh parameterization in the finite element analysis. Indeed, FE results diverge from test results in regions close to the rib joints. Milani et al. [2008] use a homogenization technique that treats brick, mortar and block as a continuum, then introduce a novel triangular curved finite element for vault analysis.

In order to model discontinuity of masonry, some work has been done using micro-modeling, where separate finite elements are assigned to the masonry block elements and interface (mortar) elements [Bicanic et al. 2003]. However, micro-modeling leads to a large number of variables as well as a very stiff numerical system due to the vast difference in rigidity of stone versus joints. In this case convergence and performance become problematic [Molins and Roca 1998]. Further, FE methods show extreme sensitivity to support conditions and displacements in geometry. In a study by Loo and Yang [1991], they applied nonlinear stress-strain properties to models of arch bridges and found high sensitivity to small movements in the supports.

2.2.2 Stability Methods

Fundamentals Heyman's *The Stone Skeleton* [1966; 1995] is a foundational publication on behavior of masonry structures. Aside from offering an intuitive description of the structural actions of masonry, it provides analysis and collapse mechanisms for common structural elements (e.g. arches, vaults, domes, spires, and flying buttresses) which are used as test cases for verification of research results. The three basic assumptions of masonry behavior introduced by Heyman [1966] are:

1. Masonry has no tensile strength.

2. Stresses are so low that masonry effectively has unlimited compressive strength.

3. Sliding failure does not occur.

Intuitively, this means there is no 'glue' that holds blocks together under a pulling force, and that blocks can be considered infinitely rigid. These simplifying assumptions allow us to treat masonry as stacked rigid blocks. Stability depends only on the geometric arrangement, namely whether the blocks can stand in static equilibrium. The stability approach is commonly referred to as limit analysis, and it can be carried out with a variety of equilibrium approaches.

Analysis Methods Traditionally, as early as the 1700s, stability methods were applied to masonry analysis using graphical techniques called graphic statics [Huerta 2004]. Engineers solved for equilibrium conditions using geometric relationships between force vectors. Block et al. [2006] enabled interactive analysis of masonry structures by implementing graphic statics in the parametric geometry software Cabri Geometry (www.cabri.com). However, the analyses are constructed manually for each structure, whereas our analysis applies to arbitrary geometry. Moreover, graphic statics is mainly practical for two-dimensional slices of buildings.

Livesley [1978] set the foundation for a computational implementation of limit analysis of 2D masonry structures, with further work on a small class of 3D structures [Livesley 1992]. Livesley formulated the limit analysis problem (determining the collapse load) as a linear system, such that it could be solved with linear programming. Similar to our approach, the structure is divided into rigid block elements. The equilibrium conditions are expressed in terms of concentrated force resultants at inter-element boundaries, which are statically equivalent to the actual boundary stresses. While Livesley's method provides a yes/no answer on whether a structure is feasible, we extend this technique to provide a metric for infeasibility that can be useful as a cost function in structural optimization problems. In practice, the RING software applies limit state analysis to 2D masonry bridges [Gilbert 2001].

Continuing work on limit analysis for masonry structures, Harvey [1991] applies the rigidity assumption and equilibrium approach, but incorporates a zone of thrust to accommodate uncertainties in material properties and second order effects. Gilbert et

al. [1994; 2006] focus on friction behavior for rigid block structures and consider only two-dimensional problems. Similarly to Livesley, authors Boothby and Brown [1992] formulate their analysis using mathematical programming with the simplifying assumptions of masonry stated above. In contrast, their objective function is to minimize total potential energy under kinematic constraints rather than solving for the limit load. Zessin et al. [2010] study collapse conditions for masonry domes considering movement in the supports. They apply a quasi-2D thrustline analysis to assess the stability of 3D hemispherical domes and validate collapse predictions experimentally on scale models.

2.3 Design and Optimization Methods

In this section we review methods in design of architectural geometry, and assess progress in integrating structural objectives into the design process.

2.3.1 Architectural Modeling

There is a large body of work on Computer-Aided-Design (CAD) for Architecture, with applications such as AutoCAD, Rhino, and CATIA among the most sophisticated and widely used 3D design software. Other methods proposed for modeling architecture include data-driven techniques [Merrell et al. 2010] and procedural modeling [Parish and Müller 2001; Wonka et al. 2003; Müller et al. 2006; Müller et al. 2007; Talton et al. 2011] where grammars are used to produce variations of building designs. Structural considerations do not apply in these applications. Attar et al. [2009; 2010] have visited the use of physics in design with user-set constraints. However physics was used as an interaction and modeling tool rather than to create structurally feasible forms, for example, designing a surface using cloth simulations combined with collision objects or torsional forces. In Chapter 4 we revisit procedural modeling as a parameterization for creating structurally-sound buildings.

2.3.2 Shape Optimization

General Framework Delfour and Zolésio [2001] and Laporte and Le Tallec [2003] lay down the mathematical foundations for optimization problems where the control variable is a geometric domain. The optimization problem is formulated in abstract, mathematical terms as finding an optimal shape Ω^* such that:

$$\begin{aligned} y(\Omega^*) = \quad & \min_{\Omega} \quad y(\Omega, \mathbf{f}_\Omega) \\ & s.t. \quad S(\Omega) \leq \epsilon_0 \end{aligned}$$

Ω denotes the unknown shape to be optimized. $S(\Omega)$ is the set of *geometric constraints* imposed on the shape Ω. In our case this could be the thickness of a column or height of a ceiling. $\mathbf{f}_\Omega$ is a *state variable*, in our problem formulation this refers to the state of forces at the shape Ω. Function $y(\Omega, \mathbf{f}_\Omega)$ is the objective which must be minimized and depends both on the shape Ω and the solution of the state variable $\mathbf{f}_\Omega$. The problem is reduced to a standard form of mathematical programming by defining a discretization of Ω into a finite number of control parameters. In this dissertation, we express the stability of a building in this general framework and adapt it to address nontrivial issues specific to architectural modeling, masonry, and interaction with users.

Interactive Design Within a shape design context, optimization frequently appears as the underlying formulation. Welch and Witkin [1992] solve a constrained variational optimization for interactive modeling of free-form surfaces. Similar to our approach they formulate the surface energy as a quadratic program, and incorporate user-controlled constraints such as fixed points. 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]. More broadly, Harada et al. [1995] optimize constrained layout designs with physically based user interaction. However, these examples do not consider structural feasibility constraints.

2.3.3 Structural Design

Structural design refers to shape optimization driven by mechanical objectives and constraints. The predominant concern is to satisfy the laws of mechanics (e.g. supporting applied loads), while other objectives may be efficient material use (cost), or fabrication constraints. We review a few articles that represent progress in this area relevant to our approach.

Allaire et al. [2004] describe structural optimization combining shape derivatives with a level-set method. For a predefined topology, they are able to find the optimal material distribution under various boundary conditions such as a cantilever beam or anchor points with uniform loading. Contrary to our approach their method is based on properties of elasticity in the design material. Smith et al. [2002] developed a technique for automatic optimization of truss structures with user input on support conditions, applied loads and bounding volumes. Similar to our work, they consider static equilibrium analysis, however in trusses only axial forces are considered. Further, Smith et al. have a smaller solution space to search given that trusses are typically restricted to two levels of joints, compared to the arbitrary stacking of blocks in masonry architecture. Due to the restricted variables, they are able to consider topological variations by initiating their optimization with all possible beam connections in place, and merging unnecessary elements as the optimization progresses. The EifForm application [Shea 2000] is an example of integrating FEM with generative structural design methods. EifForm differs from our technique by use of a simulated annealing algorithm whereas we apply gradient-based optimization.

Many CAD modeling systems, such as CATIA (www.3ds.com/products/catia) and Revit (usa.autodesk.com/revit-architecture), integrate finite element analysis into the modeling package to shorten the pipeline between model creation and structural analysis. Visual feedback from FE analysis is provided which indicates the current state of stress. Although this allows designers to iterate through design options and quickly assess structural output, there is no guidance to the user on how to modify designs specifically for improving stability. Manual model adjustment is

required which can be ineffective for designers lacking intuition in mechanics.

Allen and Zalewski [2009] promote the use of graphic statics for generation of good structural form, and demonstrate several real-world constructions that employed graphical techniques. Block and Ochsendorf [2007] developed the Thrust Network Analysis method that extends graphic statics to 3D structures for interactive design of masonry shell structures and exploration of design variations. Similar to our approach, they use a stability method that finds a compression-only structure from an input mesh and boundary constraints. They consider equilibrium of resultant axial forces at block interfaces, and use the concept of dual force polygons to determine the thrust network. Their approach is specific to shell structures with topologies that can be projected onto a 2D plane, while we support arbitrary topology.

Computer Graphics Applications Outside of architecture, stability analysis has been applied to a wide range of model subjects in computer graphics applications with the notion that physical realism translates to a more realistic appearance. For example, Shi et al. [2007] use static equilibrium as a constraint for determining plausible character poses. They solve for minimum deformation of an input pose that positions the center of mass over the ground supports. In plant modeling, static analysis has been used to balance the weight of branches for creating realistic tree structures [Hart et al. 2003]. Their work solves a similar problem to our approach in chapter 4 of generating model geometry through procedural rules and determining parameters that satisfy structural constraints. However, the problem in Hart et al. [2003] is limited to a single variable for each branch specifying base diameter. More importantly, due to the tree hierarchy the dependence of each branch element on the rest of the structure is non-cyclical – each branch supports only its child branches – while in our structures elements are highly interdependent.

2.4 Summary

Analysis methods for masonry are well-suited to a rigid block assumption under equilibrium constraints. These assumptions remove a dependence on material strength, which is appropriate for masonry construction, where observed stresses are far lower than the compressive strength and deformation under stress is negligible. Our main contribution over previous analysis methods is that we define a continuous metric of infeasibility for structures that are not stable, whereas past work has offered only the binary information of whether a structure is feasible or not under stability constraints.

Within shape optimization of structures, generating structural forms that satisfy stability and efficiency objectives is a widely studied area. However, previous approaches are specific to certain classes of structures, such as trusses or elastic materials. There is still need for a shape optimization method that operates on general 3D masonry structures. The following chapters will describe our contributions in this area.

Chapter 3

Infeasibility Metric for 3D Static Equilibrium Analysis

3.1 Introduction

In order to impose structural feasibility on a building design, a forward analysis method is required to assess the soundness of a structure. Although many modern engineering analysis tools are readily available, they are predominantly based on finite element methods and elasticity theory [Zienkiewicz 1971]. Our analysis method focuses on masonry structures, which can be assumed to behave as rigid blocks [Heyman 1995]. As discussed in §2.2.1, finite element methods are not appropriate in this context because they focus on material failure and stress, where 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. For this reason, we extend an approach introduced by Livesley [1978], and we present a new forward structural analysis method based on optimization under linear constraints.

3.2 Overview

The analysis method solves a forward statics problem: given the geometry of a structure, the stress state is modelled 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 each element is subjected to linear static equilibrium constraints. We extend the analysis to return a measure of infeasibility by minimizing violation of failure criteria at the joints. This chapter 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 put to use later as an objective function in Chapters 4 and 5.

3.3 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.

3.3.1 Contact Forces

We model structures as an assemblage of rigid blocks, and analyze the force distributions at the interfaces between adjacent elements. Figure 3-1 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.

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_u^i and f_v^i. The direction of in-plane friction forces is determined independently at each block interface, with u aligned to an (arbitrary) edge of the block face. Friction forces on shared faces have opposite orientation.

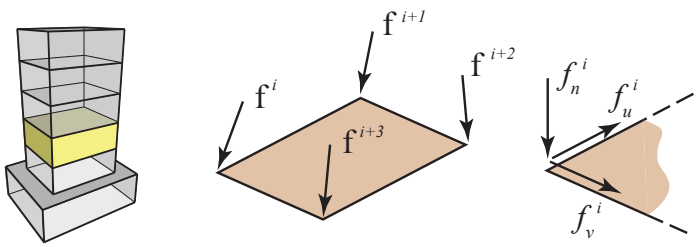


Figure 3-1: Model of contact forces at interfaces between blocks.

3.3.2 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]:

$$\mathbf{A}_{eq} \cdot \mathbf{f} + \mathbf{w} = \mathbf{0} \tag{3.1}$$

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 §A.1). The system has six rows per building block: three for the x, y, z components of the net force, and three for the net torque. In general, structures have a sparse $\mathbf{A}_{eq}$ matrix due to the small number of interactions between blocks – each 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.

3.3.3 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, 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 significant tensile strength to the construction. This condition is expressed as a

non-negativity constraint on the axial forces:

$$f_n^i \geq 0, \quad \forall\, i \in \text{ interface vertices} \tag{3.2}$$

3.3.4 Friction Constraints

A friction constraint is applied at all vertices of the block interfaces. For each triplet of forces $\{f_n^i, f_u^i, f_v^i\}$ 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:

$$|f_u^i|, |f_v^i| \leq \alpha f_n^i, \quad \forall\, i \in \text{ interface vertices} \tag{3.3}$$

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:

$$
\begin{aligned}
\mathbf{A}_{eq} \cdot \mathbf{f} &= -\mathbf{w} && \texttt{\textbackslash\textbackslash equilibrium} \\
\mathbf{A}_{fr} \cdot \mathbf{f} &\leq \mathbf{0} && \texttt{\textbackslash\textbackslash friction} \\
f_n^i &\geq 0, \quad \forall\, i \in \text{ interface vertices} && \texttt{\textbackslash\textbackslash compression-only}
\end{aligned}
\tag{3.4}
$$

3.4 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 (3.4) 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 (3.4) 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. 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]:

$$
\begin{aligned}
f_n^i &= f_n^{i+} - f_n^{i-} \\
f_n^{i+}, f_n^{i-} &\geq 0
\end{aligned}
\tag{3.5}
$$

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 (3.4) is then a quadratic

program:

$$\begin{aligned} \min_{\mathbf{f}} \quad & \sum_{i=0}^{n} (f_n^{i-})^2 \\ \text{s.t.} \quad & \mathbf{A}_{eq} \cdot \mathbf{f} = -\mathbf{w} \\ & \mathbf{A}_{fr} \cdot \mathbf{f} \leq \mathbf{0} \\ & f_n^{i+}, f_n^{i-} \geq 0, \quad \forall i \end{aligned} \tag{3.6}$$

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 (3.6) 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.

3.5 Robustness

The constraints in (3.6) 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 3-2), and applying the interface forces at the modified vertex positions, which provides a margin of safety.

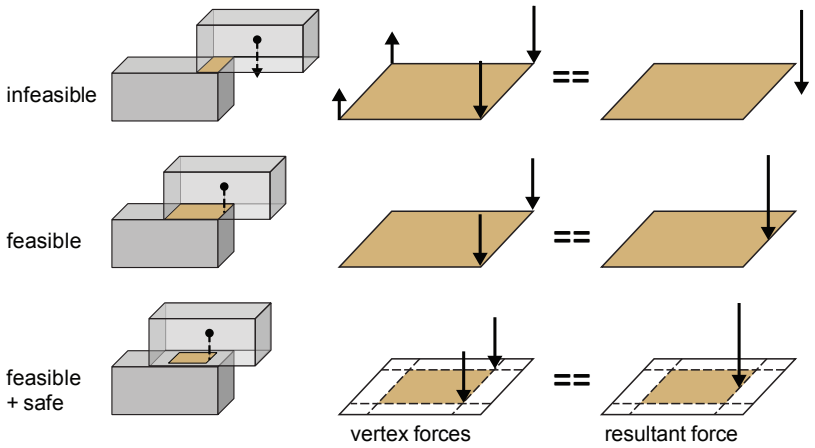


Figure 3-2: (Left) 2-block structure with self-weight acting at the centroid (black arrow). (Center) Interface vertex forces acting on the top block are displayed. (Right) The resultant force has the equivalent net force and torque contribution to all vertex forces. For compression-only solutions the 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 bridges which support heavy traffic, but less important for large 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 $\mathbf{w}$ in eq. (3.1).

3.6 Results

3.6.1 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 a rigid-block arch becomes infeasible. For

hemispherical arches 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.

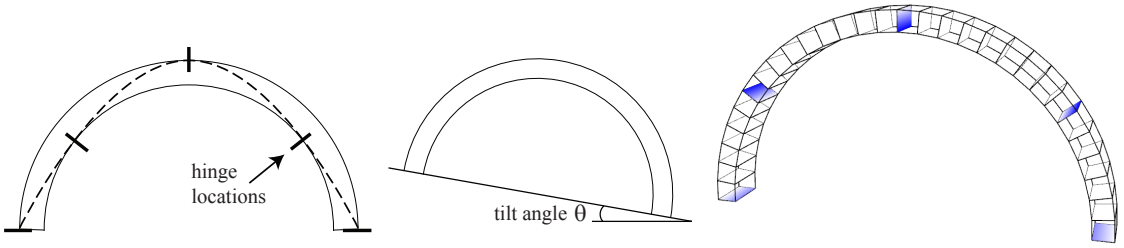
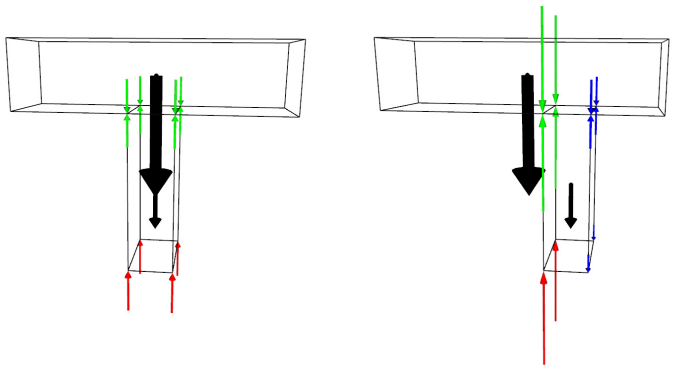


Figure 3-3: (Left) Published values on min-thickness semicircular arches was used for validation. Locations where the thrustline touches the block edges are where hinges will form in a collapse mechanism. (Center) Maximum tilt angle before collapse was a second validation. (Right) 30-block semicircular arch slightly thinner than the minimum thickness, demonstrating that tension forces appear in the expected hinge locations (blue interfaces) using our method.

3.6.2 Examples

The result of the infeasibility metric is shown for a selection of example models. In Figure 3-4 a 2-block T structure is modeled in both a feasible and infeasible configuration. In Figure 3-4(a) the structure is feasible since the center of mass (black arrow) of the top block is positioned over the base block. The resulting force vectors show that, accordingly, only compression forces are present at the joint between the two blocks (green arrows) and at the ground (red arrows). In Figure 3-4(b), tension forces appear in the infeasible arrangement (blue arrows). See Appendix B for numerical data of this model including vertex coordinates and force values.

The groin vault pictured in Figure 3-5 is infeasible. Tension forces (blue) indicate where potential hinge mechanisms might form during collapse.



(a) Forces in feasible model (b) Forces in infeasible model

Figure 3-4: (a) A feasible model where only compression forces (green arrows, red at ground) are present to balance the self-weight of the blocks (black arrows). (b) An infeasible model displaying the location of tension forces (blue arrows) where a potential hinge mechanism would form at collapse.

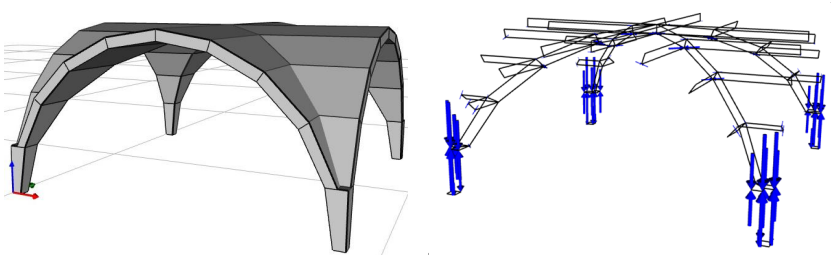


Figure 3-5: (Left) An infeasibly thin groin vault. (Right) Result of the quadratic program. The minimum tension solution places tension forces (blue arrows) around the base of the vault to counteract the outward push from accumulated weight of the blocks.

3.6.3 Limitations

Friction Our method for measuring infeasibility considers failure of the structure by the presence of tension forces at joints between blocks. Physically this is manifested as a hinging mechanism between blocks. 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. Further investigation is required for incorporating potential friction failure into the model of stability for masonry analysis.

Alternate Metrics We quantify infeasibility based on the magnitude of tension forces. Although tension is an indication of failure, using force magnitudes alone can lead to scenarios which falsely consider one structure more feasible than another. For example, consider the T-model in Figure 3-4(b): if the tension vertices of the bottom block were translated to the right (thickening the bottom block), an updated static analysis would show that the tension forces decrease in magnitude. This is due to the increased torque arm – less force is needed to balance the applied load from the top block. However, if we consider a different metric, such as distance of the resultant force from the face boundary (Fig. 3-2), we see that the resultant force does not move and change in feasibility is zero. Consideration of alternative definitions for feasibility is an area of future work.

3.7 Discussion

We have introduced a measure of infeasibility as a key objective for assessing the structural soundness of masonry 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 previous theoretical results validate the accuracy of the technique. We have introduced a penalty form that allows us to measure the degree of infeasibility of a structure. This metric can be used in a search procedure to yield a stable building which will be demonstrated in Chapters 4 and 5.

Chapter 4

Procedural Modeling of Structurally-Sound Buildings

4.1 Introduction

Procedural modeling has emerged as a powerful technique for generating architectural geometry [Parish and Müller 2001; Wonka et al. 2003; Müller et al. 2006; Müller et al. 2007; Talton et al. 2011]. These systems use grammars to produce variations of building designs, and are adaptations of the seminal work on shape grammars introduced by Stiny [1975; 1980]. Designs can be generated through random or user-selected parameter adjustment. However, existing techniques focus on visual realism and detail in the building façade, and do not account for the structural validity of the results. Users may not have intuitions about the mechanics that govern structural stability, or knowledge of traditional proportions used in building design. Determining the precise dimensions of a structure that result in stability can be a tedious task. In this chapter we present a method to automatically "snap" to feasible dimensions, while leaving control in the designer's hands to decide 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 build on the forward analysis method described in Chapter 3 to search for parameters of our procedural model that yield a stable building. In particular, we use the measure of infeasibility to define an energy function and use non-linear optimization to find the appropriate parameters.

Figure 4-1: 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 barrel-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.

4.2 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 4.2. 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 when a local minimum is reached.

4.3 Procedural Model

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). An example construction is shown in Figure 4-3.

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 structures.

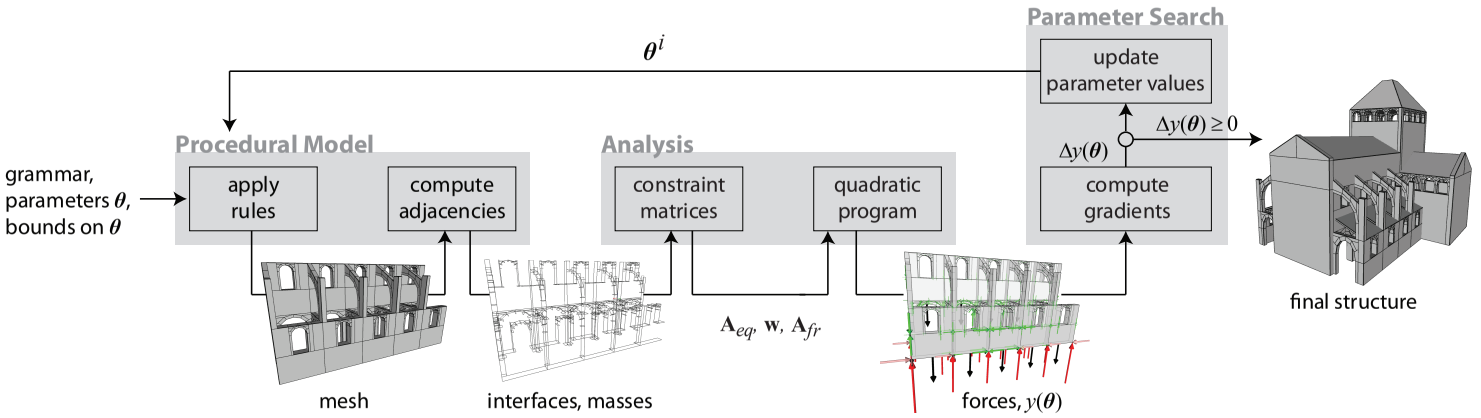


Figure 4-2: Pipeline. User input is a set of production rules, the selection of free parameters, and associated bounds.

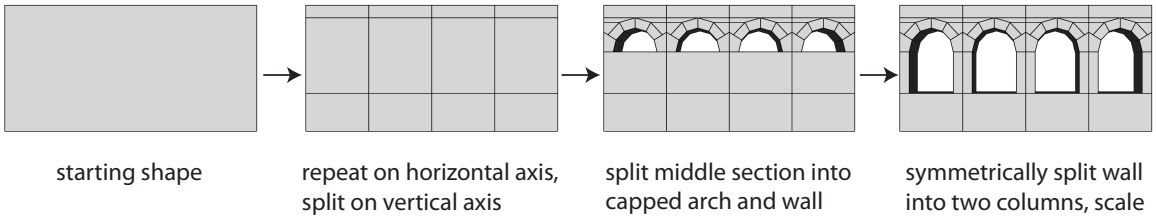


Figure 4-3: Procedural generation of a wall with four windows.

4.3.1 Free Parameters

The key feature of our approach is that we automatically choose parameter values according to physical constraints. For a given set of production rules, the user designates 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. The user can also place bounds on the parameter values. The user's selection of free parameters and associated limits defines the family of possible design variations.

4.3.2 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 historic masonry architecture (Fig.4-4). 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. Our method generalizes to any shapes composed of blocks. Accordingly, alternative libraries of primitives could be used to accomplish different styles.

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 (the cutting patterns that determine block geometry is termed *stereotomy*). A poorly-chosen interface orientation can result in an unstable structure. The effect of interface orientation on the solution space

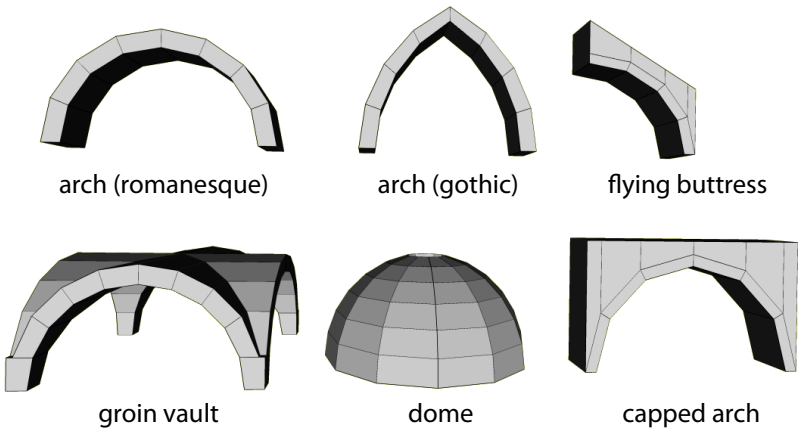


Figure 4-4: Template shapes specific to masonry architecture. Each has parameters to alter tessellation and block thickness.

is an open area for future research. The vertex-level technique in Chapter 5 proposes one way of improving interface orientations.

4.3.3 Nonstructural Shapes

Procedural modeling methods have the capability to tag shapes with different 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.

4.3.4 Adjacencies

Once the geometry of the building model has been generated, we compute contact surfaces between adjacent blocks as shown in Figure 4.2. 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.

4.4 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:

$$\begin{aligned} \underset{\boldsymbol{\theta}}{\text{argmin}} \quad & y(\boldsymbol{\theta}) \\ s.t. \quad & lb \leq \boldsymbol{\theta} \leq ub \end{aligned} \tag{4.1}$$

where $y(\boldsymbol{\theta})$ is the infeasibility metric from expression (3.6). 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.

Initial parameter values $\boldsymbol{\theta}^0$ are input by the user. At each iteration i of the optimization, the geometry of the procedural model is updated according to parameter values $\boldsymbol{\theta}^i$, we then use forward analysis to measure progress towards a feasible structure as described in §3.4, expression (3.6). In contrast to forward statics where forces are determined based on fixed geometry, our optimization procedure determines new geometry that satisfies constraints on the forces.

Figure 4-5 shows the energy landscape for a two-parameter structure and the corresponding optimization path. The example structure consists of a semi-circular arch with free thickness, supported on two columns with free width. The edge of the feasible region along the column width axis illustrates that for arbitrarily large columns (effectively acting as ground), the arch has local feasibility limits on thickness.

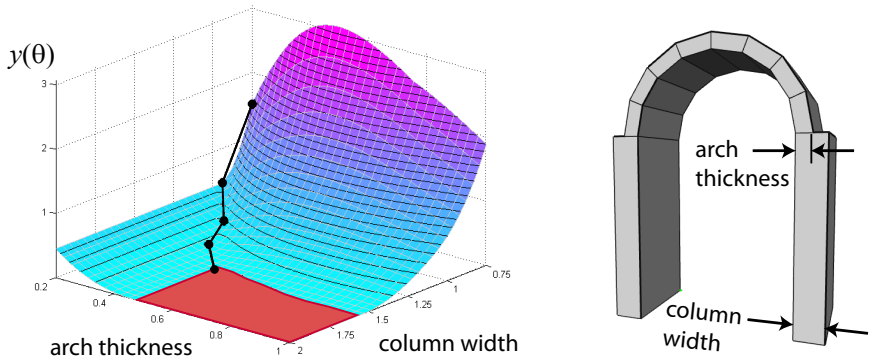


Figure 4-5: Energy landscape for a two-parameter structure: circular arch supported on columns. The feasible region is the zero plane highlighted in red.

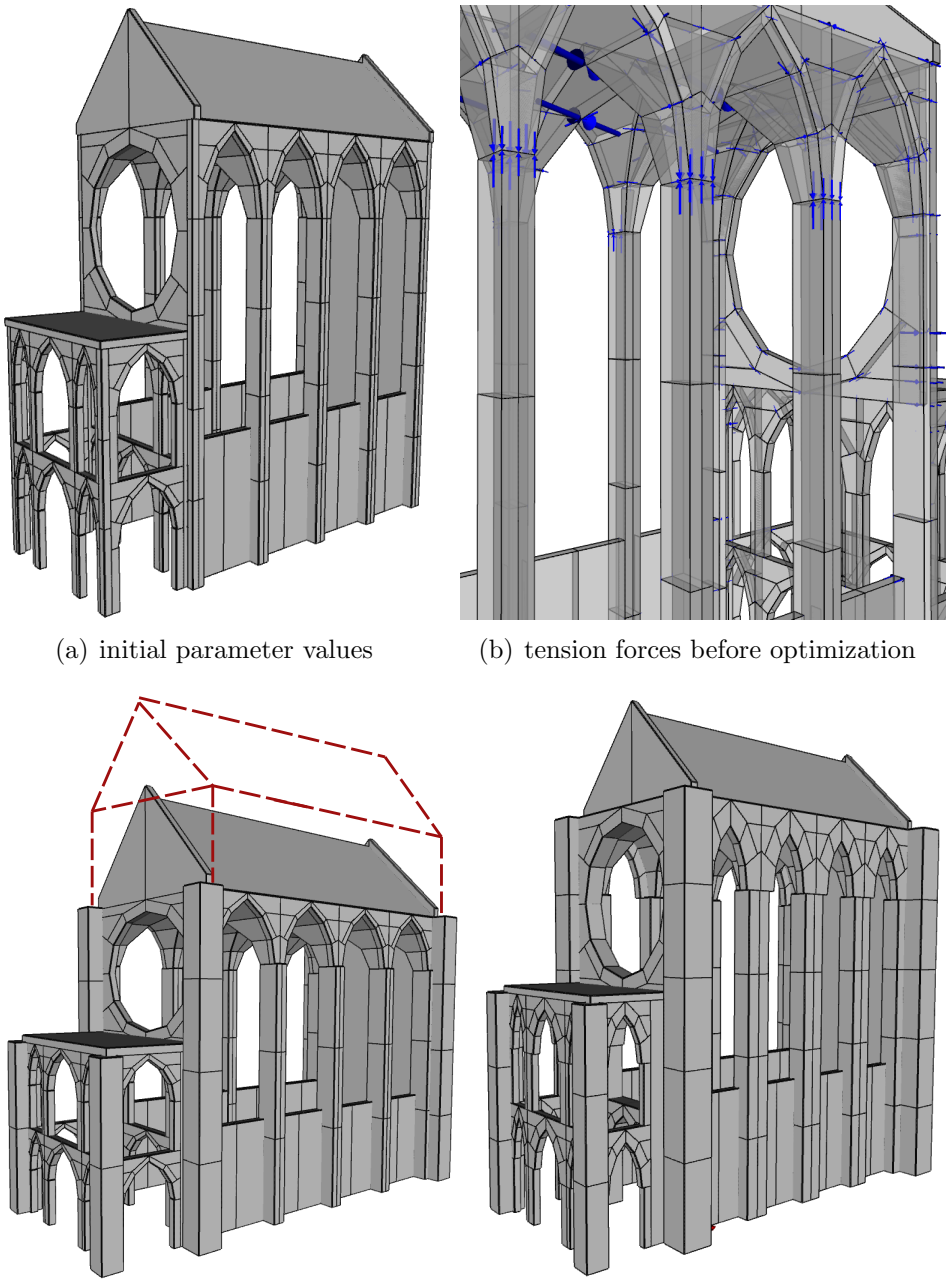
4.5 Results

4.5.1 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.

4.5.2 Modeling stable buildings

Figure 4-1 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. The user controls which parameters are free variables and can affect the structural trade-offs that yield a sound shape. The optimization in Figure 4-6(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 4-6(d)) finds a feasible solution at the original building height in exchange for smaller windows and thicker walls. Figure 4-7 shows a variety of structural models achieved by making



(a) initial parameter values

(b) tension forces before optimization

(c) 4-parameter optimization

(d) 10-parameter optimization

Figure 4-6: 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.

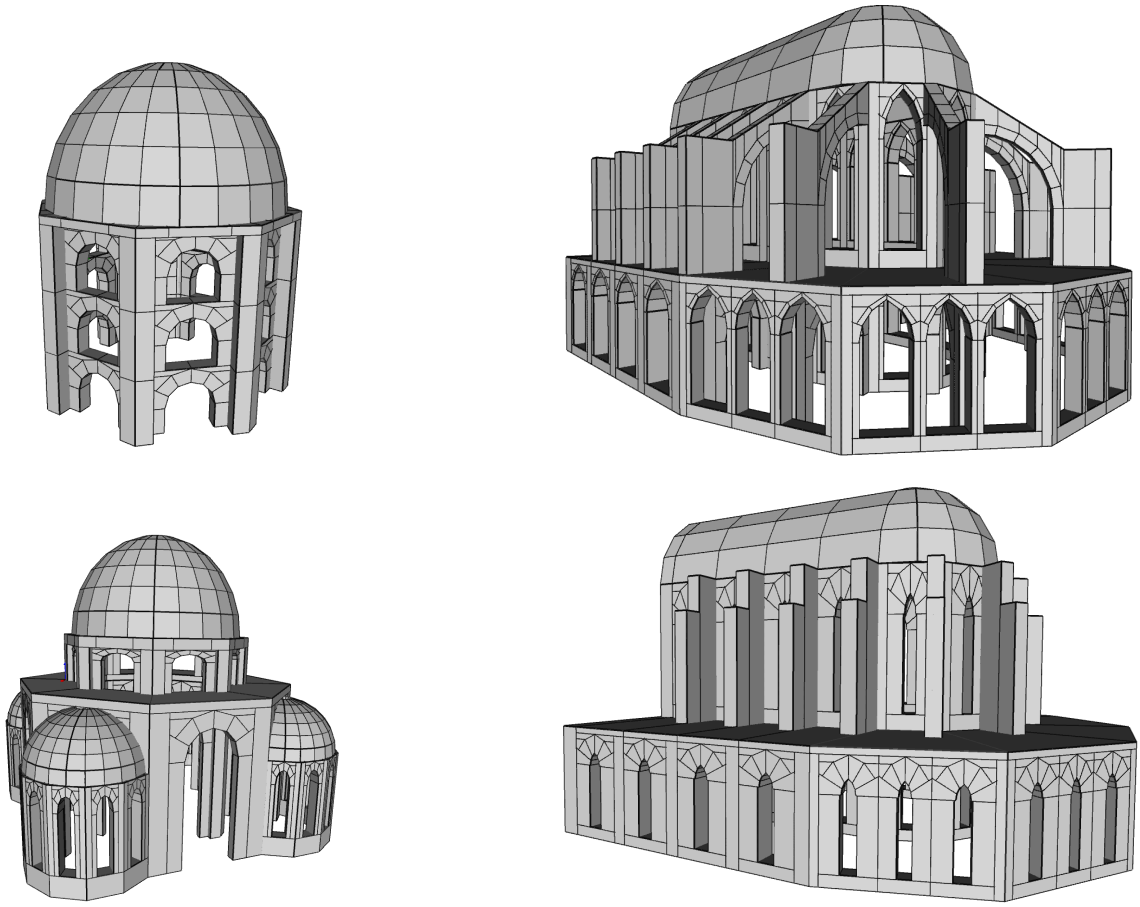


Figure 4-7: (Left-top) Model of a mosque generated with procedural rules, and optimized for feasibility; (left-bottom) generated by extending the grammar. (Right-top) Radial flying buttresses are optimized; (right-bottom) smaller windows are required when the buttresses are removed from the grammar.

modifications to the grammar. The optimized model in Figure 4-7(d) has small windows to support the domed ceiling. Extending the grammar to include flying buttresses (Figure 4-7(b)) transfers the load away from the walls, allowing for larger windows for increased natural light in the interior.

The tower in Figure 4-8 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: the horizontal position of each of the 16 levels is controlled independently by two free parameters for translation in x and y directions.

In Figure 4-9 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 4-9, left) the arches are too thin to stand. The parameter search generates arches resembling catenaries (Figure 4-9, 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.

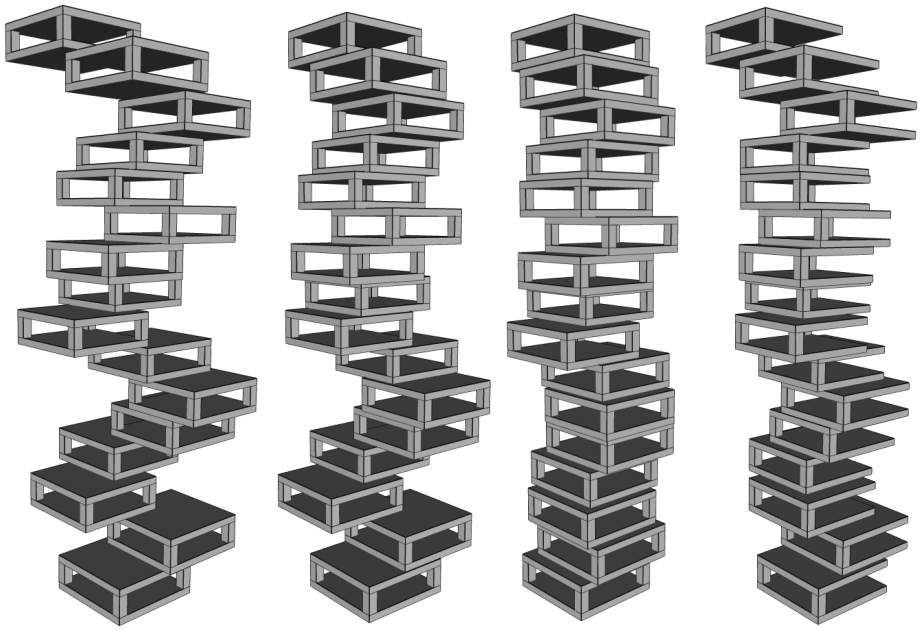
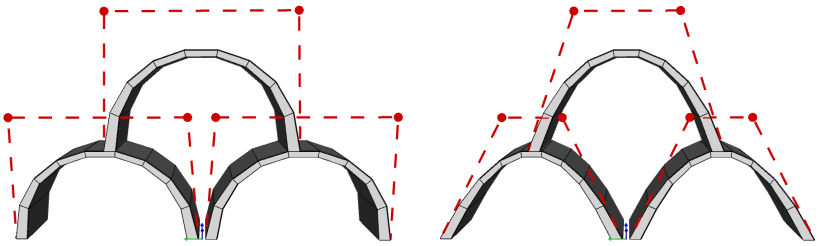


Figure 4-8: From the original tower input (a), 32 parameters were optimized to create a feasible stacking arrangement (b). The position of each level was controlled by two separate parameters for translations in x and y. Variations can be found using different initial positions. In (c) the interfaces were shrunk by 25% along each edge as a geometric factor of safety (§3.5). In (d), modifying the grammar provides further variation.

4.5.3 Performance

Table 4.1 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 $n_{it} n_\theta \, complexity$(BPMPD solver),



(a) infeasible input (b) feasible output

Figure 4-9: 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. (a) In the initial configuration the circular shape is infeasible. (b) The feasible result. The bottom arches are slightly asymmetrical to account for horizontal forces transferred from the top arch.

where n_{it} is the number of iterations in the parameter search, and n_θ is the number of free parameters. The linearity in the number of parameters is due to our use of finite differences for gradient computations.

Model	Blocks	Parameters	Iterations	Time/Iter.
Cluny (Fig. 4-1)	986	4	10	45.7 s
		5	5	57.3 s
		7	4	70.0 s
		9	9	106.6 s
arch (Fig. 4-5)	10	2	6	0.1 s
Sainte Chapelle (Fig. 4-6)	486	3	4	12.5 s
		5	9	26.5 s
		7	6	29.3 s
		10	8	40.1 s
tower (Fig. 4-8)	96	32	6	12.5 s
barrel vault (Fig. 4-10)	140	1	8	0.6 s

Table 4.1: Performance results for parameter search.

4.5.4 Editing Parameters

Interactive editing of parameters is another valuable usage scenario when speed permits. The user may edit a model by manually changing a parameter value, while our system automatically updates the free parameters to maintain structural feasibility. For example, in Figure 4-10, 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 in under five seconds for this model. Changes in a design can alter the loads on many other parts of the structure, and 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 updating free design parameters.

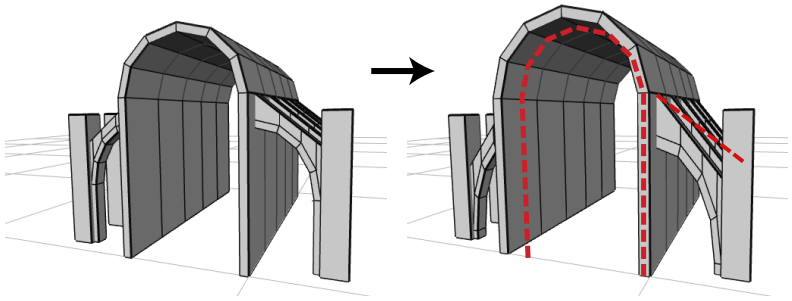


Figure 4-10: Interactive editing of parameters. As the user increases the span of the barrel vault roof, our system automatically selects the angle of the flying buttresses required to maintain stability. Red lines highlight the original structure.

4.5.5 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 4-11(top sequence) 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.

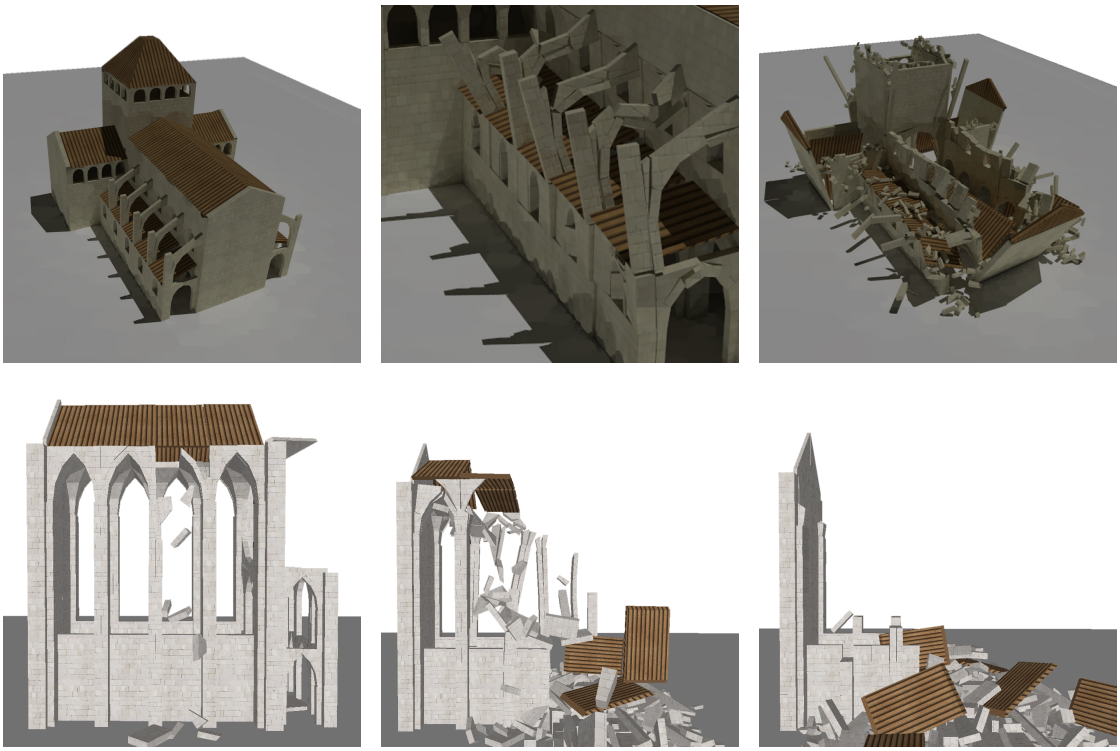


Figure 4-11: Structurally sound models can be manipulated in physically simulated environments. (Top) The Cluny model collapses after a ground shake is applied; (Bottom) the Sainte Chapelle model collapses after a central column is broken.

4.5.6 Limitations

Friction Cone Approximation To test the effect of the friction pyramid parameterization on feasibility, we performed parameter searches on the Sainte Chapelle model (Figure 4-6) 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.

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 a global minimum (zero tension) solution without

any aids, e.g. multiple starting points. We tested the tower model in Figure 4-8 by running the 32-parameter search from randomly generated starting points and all converged to a zero-tension solution.

Existence of a Solution 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.

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.

4.6 Discussion

We have introduced structural soundness as a key objective in procedural modeling of buildings. To achieve this, we rely on the quadratic programming formulation of the equilibrium equations introduced in Chapter 3. The penalty form allows us to measure the degree of infeasibility of a structure, which is 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 trade-offs.

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 in architectural design, in designing interactive virtual environments, and for applications in historical structures education. 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.

Chapter 5

Gradient-Based Structural Design with Constraint Authoring

5.1 Introduction

The goal of this section is to offer interactive tools that modify a current form with guarantees on improving stability while respecting constraints and objectives set by the designer. We describe a design interface that is parameterized directly on the vertex positions which gives complete freedom to the user for designing free-form shapes. Further, the flexibility of the constraints and objectives allows the user to explore a variety of solutions and designs based on their style and context. For example, Figure 5-10 shows how structural gradients can be used to modify the shape of an unstable shell to make a feasible, self-supporting design. As in chapters 3 and 4, we base our model of stability on static analysis [Livesley 1992], and seek to minimize an instability metric by modifying the geometry.

The heart of our approach is to compute the gradient of the stability metric with respect to geometry modification. We then visualize this gradient to inform the architect, or use it directly to derive a number of interactive tools that modify the shape to improve stability. Whereas the previous method in Chapter 4 aimed for a mostly automatic solution to find a stable structure, the approach described in this chapter gives the user fine-grained control over the produced structure and freedom

to explore design variations.

The designer starts with an initial model represented as an assemblage of solid blocks, and performs modifications to the geometry, informed by our structural gradient. We provide visualizations that suggest the directions in which vertices should move to improve stability. More importantly, we provide geometry modification tools that leverage the gradient of stability. Each tool relies on a number of user-specified constraints and objectives, such as the preservation of vertical or horizontal directions or the minimization of material usage. Using a graphical interface, the user selects parts of the geometry to modify, the constraints and the objective, as well as the magnitude of the modification. The geometry is updated interactively, enabling a fluid creative exploration.

5.1.1 Contributions

The following contributions are presented:

- We provide a closed form derivation of the gradient of a measure of stability with respect to geometry modification.
- We enable visualization of structural information based on the gradient.
- We supply a variety of interactive tools that enable the improvement of stability based on user-provided constraints and objectives.

We demonstrate that our technique can lead to a variety of designs given the same input shape, based on the user's decisions.

5.2 Analytic Structural Gradient

5.2.1 Overview

Given an infeasible model, we demonstrate how to compute *structural gradients* $\nabla y(\Omega)$ that inform how the feasibility changes as the geometry is modified. As de-

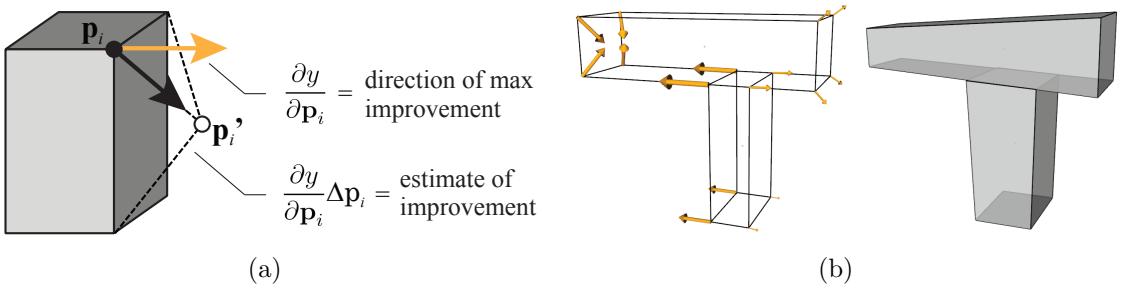


Figure 5-1: (a) Yellow arrows indicate the gradient of feasibility, parameterized on the model vertices $\mathbf{p}_i$. (b) The set of gradient vectors for an infeasible T-model. In the modified geometry the overall improvement in feasibility $\approx \sum_i \frac{\partial y}{\partial \mathbf{p}_i}\Delta \mathbf{p}_i$

picted in Figure 5-1, the value of the gradient is that it provides the direction of vertex displacements that maximally improve the structure's feasibility. The gradient also provides a way to estimate the change in energy for any change in vertex positions $\Delta \mathbf{p}$ using a Taylor series expansion: the gradient can be projected onto the actual change in geometry. This is a useful property since the designer may often apply edits to the geometry that do not follow the gradient exactly. Our method has the following steps:

- We derive a closed form expression for the infeasibility metric $y(\Omega)$, where $y(\Omega)$ is a quadratic program that we solve locally for a given geometry $\overline{\Omega}$. The closed form expression is derived by transforming active constraints into equalities at this local solution.

- Using the closed form expression for $y(\Omega)$, we derive an analytic gradient $\nabla y(\Omega)$. This is done by computing the Jacobian of the constraint matrix.

- We then describe our specific parameterization for geometry manipulation ω to detail the components of $\nabla y(\Omega)$.

5.2.2 Closed Form Energy

The infeasibility metric of the structure Ω is formulated as a quadratic program. Let the force vector $\mathbf{f}^*$ be the global minimizer for the infeasibility metric $g(\mathbf{f})$, as given

in Expression (3.6) and repeated here:

$$\begin{aligned} \min_{\mathbf{f}} \quad & g(\mathbf{f}) = \tfrac{1}{2}\ \mathbf{f}^T\mathbf{H}\mathbf{f} \\ \text{s.t.} \quad & \mathbf{A}_{eq} \cdot \mathbf{f} = -\mathbf{w} \\ & \mathbf{A}_{fr} \cdot \mathbf{f} \leq \mathbf{0} \\ & f_n^{i+}, f_n^{i-} \geq 0, \quad \forall i \end{aligned} \tag{5.1}$$

The set of *active constraints* is the set of constraints that are satisfied as equalities at $\mathbf{f}^*$. The active friction constraints are denoted by $\widetilde{\mathbf{A}}_{fr} \cdot \mathbf{f} = \mathbf{0}$, where $\widetilde{\mathbf{A}}_{fr}$ contains the rows of $\mathbf{A}_{fr} = \{\ i \mid \text{row}_i(\mathbf{A}_{fr}) \cdot \mathbf{f}^* = 0\}$. The active lower bounds are denoted by $\widetilde{\mathbf{I}}_{lb} \cdot \mathbf{f}^* = \mathbf{0}$, where $\widetilde{\mathbf{I}}_{lb}$ contains the rows of the identity matrix $= \{\ i \mid f_i^* = 0\}$.

The active constraints are identified using the Lagrange multipliers returned by the QP solver. We then combine the active constraints into a new set of equalities $\mathbf{C} \cdot \mathbf{f} = \mathbf{b}$, which is a concatenation of the static equilibrium constraints and the active inequality constraints.

$$\mathbf{C} = \begin{bmatrix} \mathbf{A}_{eq} \\ \widetilde{\mathbf{A}}_{fr} \\ \widetilde{\mathbf{I}}_{lb} \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} -\mathbf{w} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}$$

Given the active constraints at $\mathbf{f}^*$, we can reformulate $y(\Omega)$ as:

$$\begin{aligned} y(\Omega) = \quad & \min_{\mathbf{f}} \quad g(\mathbf{f}) = \frac{1}{2}\ \mathbf{f}^T\mathbf{H}\mathbf{f} \\ & \text{s.t.} \quad \mathbf{C} \cdot \mathbf{f} = \mathbf{b} \end{aligned} \tag{5.2}$$

which takes the form of an equality-constrained quadratic program. Recall that $g(\mathbf{f})$ is a quadratic weighting function for the force vector, where $\mathbf{H}$ is a diagonal matrix containing non-zero weights. In contrast to (3.6), which weighted only tension forces, this formulation places weights on all forces: $\mathbf{H}$ is a full-rank diagonal matrix with large penalty weight on the tension forces and low weight on the remaining forces (compression and friction). This is an intuitive method to account for the indeter-

minacy of structures – while many possible solutions of $\mathbf{f}$ may exist that satisfy the constraints, our formulation searches within that space for a solution that simultaneously minimizes tension and keeps overall force values low. An additional benefit of adding weights to all forces is that $\mathbf{H}$ becomes positive definite, which is necessary when deriving the analytic gradients for y.

By introducing the variable transformation $\mathbf{z} = \mathbf{H}^{1/2}\mathbf{f}$, the new equality constrained QP in expression 5.2 can be re-written as finding the minimum norm solution to a linear system [Bertsekas 1995] (§2.1.1):

$$\begin{aligned} y(\Omega) = \quad & \min_{\mathbf{z}} \quad \frac{1}{2}\left\|\mathbf{z}\right\|^2 \\ & \text{s.t.} \quad (\mathbf{C}\mathbf{H}^{-1/2})\mathbf{z} = \mathbf{b} \end{aligned} \tag{5.3}$$

The solution of this problem is obtained through the Moore-Penrose pseudoinverse:

$$\mathbf{z}^* = (\mathbf{C}\mathbf{H}^{-1/2})^+\mathbf{b}$$

where $\mathbf{G}^+ = \mathbf{G}^T(\mathbf{G}\mathbf{G}^T)^{-1}$ is the pseudoinverse for matrix $\mathbf{G}$ with full row rank. And through the transformation $\mathbf{f}^* = \mathbf{H}^{-1/2}\mathbf{z}^*$, the optimal solution in closed form is:

$$\mathbf{f}^*_\Omega = \mathbf{H}^{-1}\mathbf{C}^T(\mathbf{C}\mathbf{H}^{-1}\mathbf{C}^T)^{-1}\mathbf{b} \tag{5.4}$$

It can be seen that $\mathbf{f}^*_\Omega$ as given above satisfies $\mathbf{C}\mathbf{f} = \mathbf{b}$ as required. Note that $\mathbf{H}$ must be positive definite, and the rows of $\mathbf{C}$ must be linearly independent [Bertsekas 1995]. Since we know the structure of the constraint matrices $\mathbf{A}_{eq}$ and $\mathbf{A}_{fr}$, we remove linearly dependent rows from $\mathbf{C}$.

5.2.3 Energy Derivatives

We make the assumption that the set of active constraints stays fixed for small displacements of the model geometry Ω. The expression for the derivative of $\mathbf{f}^*$ is then

obtained by differentiating expression (5.4):

$$\frac{\partial \mathbf{f}_\Omega^*}{\partial \omega} = \mathbf{H}^{-1}\left(\frac{\partial \mathbf{C}^T}{\partial \omega}\mathbf{E}^{-1}\mathbf{b} + \mathbf{C}^T\mathbf{E}^{-1}\left(\frac{\partial \mathbf{b}}{\partial \omega} - \frac{\partial \mathbf{E}}{\partial \omega}\,\mathbf{E}^{-1}\mathbf{b}\right)\right)$$

where $\mathbf{E} = \mathbf{C}\mathbf{H}^{-1}\mathbf{C}^T$. The expression for the derivative of $y(\Omega)$ is then given by:

$$\begin{aligned}\frac{\partial y(\Omega)}{\partial \omega} &= \frac{1}{2}\,\mathbf{f}^{*T}\mathbf{H}\frac{\partial \mathbf{f}_\Omega^*}{\partial \omega} \\ &= \frac{1}{2}\,\mathbf{f}^{*T}\left(\frac{\partial \mathbf{C}^T}{\partial \omega}\mathbf{E}^{-1}\mathbf{b} + \mathbf{C}^T\mathbf{E}^{-1}\left(\frac{\partial \mathbf{b}}{\partial \omega} - \frac{\partial \mathbf{E}}{\partial \omega}\,\mathbf{E}^{-1}\mathbf{b}\right)\right)\end{aligned} \tag{5.5}$$

where ω is a parameterization of the structure's geometry Ω, for example, the position of individual vertices on each block face. Our chosen parameterization is described in Section 5.3. $\mathbf{H}$ is the weighting matrix for the objective function and is held constant. The derivative of $\mathbf{E}$ is $\partial\mathbf{E}/\partial\omega = \partial\mathbf{C}/\partial\omega\ \mathbf{H}^{-1}\mathbf{C}^T + \mathbf{C}\ \mathbf{H}^{-1}\partial\mathbf{C}^T/\partial\omega$ by application of the chain rule.

The terms $\partial\mathbf{C}/\partial\omega$ and $\partial\mathbf{b}/\partial\omega$ describe how the constraints change as the geometry changes according to parameterization ω. These will be computed next.

5.2.4 Constraint Derivatives

The constraint matrix $\mathbf{C}$ is a concatenation of the matrix $\mathbf{A}_{eq}$ (static equilibrium), the active friction-cone inequalities of the matrix $\mathbf{A}_{fr}$, and the active lower bounds on $\mathbf{f}$. The friction constraints and lower bound constraints are not dependent on block geometry, giving $\partial\widetilde{\mathbf{A}}_{fr}/\partial\omega = \mathbf{0}$ and $\partial\widetilde{\mathbf{I}}_{lb}/\partial\omega = \mathbf{0}$.

$$\frac{\partial \mathbf{C}}{\partial \omega} = \begin{bmatrix} \partial\mathbf{A}_{eq}/\partial\omega \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix},\ \frac{\partial \mathbf{b}}{\partial \omega} = \begin{bmatrix} -\partial\mathbf{w}/\partial\omega \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{5.6}$$

The derivation of $\partial\mathbf{A}_{eq}/\partial\omega$ and $\partial\mathbf{w}/\partial\omega$ can be obtained by applying the chain rule. We detail the derivation in Appendix A.2.

5.2.5 Weighted Energy Function

In any structure of reasonable complexity there will be a large range in the magnitude of forces observed. For example, blocks at the base of a structure carry the weight of the entire building, while in comparison, blocks surrounding the highest level of windows will have negligible forces acting on them. The energy function stated as (5.2) places weight on all forces: compression, tension and friction.

$$g(\mathbf{f}) = \frac{1}{2}\sum_i h_1 \left(f_n^{i-}\right)^2 + h_2 \left(f_n^{i+2} + f_u^{i\,2} + f_v^{i2}\right) \quad \text{, vertices } i$$

However, under this formulation, very large compression forces could overpower small tension forces ($h_1 \left(f_n^{i-}\right)^2 \ll h_2 \left(f_n^{j+}\right)^2$) and undermine the intention of the gradient to reduce overall tension in the structure. For this reason, it is crucial to weight the energy function $y(\Omega)$ according to expected magnitudes.

$$g(\mathbf{f}) = \frac{1}{2}\sum_i h_1 \left(f_n^{i-}\right)^2 + \alpha_i h_2 \left(f_n^{i+2} + f_u^{i\,2} + f_v^{i2}\right)$$

where α_i is proportional to $(\textit{compression force})^{-1}$ acting on the incident face. The magnitude of the compression force is estimated using forces values $\mathbf{f}^*$ from the previous iteration. The first iteration is run with the default uniform weights ($\alpha_i^0 = 1$).

5.3 Parameterization

For computing the gradient of infeasibility $\nabla y(\Omega)$ we parameterize Ω using a basis of vertex modifications. The fundamental constraint in choosing the parameterization is to maintain planarity of the block faces. Although contact surfaces between blocks could be represented as a triangular mesh with no planarity constraints, this would break assumptions in structural behavior. Non-planar joint geometry would eliminate the smooth friction surfaces and result in interlocking block faces which are rare in masonry construction.

We use the following parameterization which accounts for five degrees of freedom:

two degrees of freedom for in-plane vertex translation and three for orientation of the face plane (See Figure 5-2).

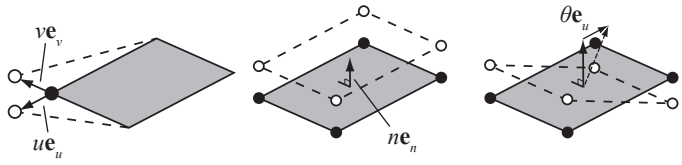


Figure 5-2: Gradient parameterization. (Left) In-plane vertex translation. (Center) Face translation along normal. (Right) Face rotation by way of rotation of the normal vector.

5.3.1 In-plane Vertex Translation

The two in-plane orthogonal vectors, $\hat{\mathbf{e}}_u$ and $\hat{\mathbf{e}}_v$, are the basis for translation on the face plane. Partial derivatives $\frac{\partial y}{\partial u}$ and $\frac{\partial y}{\partial v}$ are computed for each vertex in the structure, where the derivatives of vertex position, $\mathbf{p}$, are

$$\partial\mathbf{p}/\partial u = \hat{\mathbf{e}}_u, \quad \partial\mathbf{p}/\partial v = \hat{\mathbf{e}}_v \tag{5.7}$$

Derivatives of the face coordinate system are zero since the face plane is unchanged: $\partial\hat{\mathbf{e}}_i/\partial\omega = \mathbf{0}$ for $\omega = u, v$ and $i = n, u, v$. That is, the $\hat{\mathbf{e}}_u$, $\hat{\mathbf{e}}_v$ and $\hat{\mathbf{e}}_n$ vectors are not modified when the vertex coordinates change. The component of the gradient $\nabla y(\Omega)$ for in-plane translations are

$$\nabla_u y(\Omega) = \left[\frac{\partial y}{\partial u_1} \cdots \frac{\partial y}{\partial u_m}\right]^T, \ \nabla_v y(\Omega) = \left[\frac{\partial y}{\partial v_1} \cdots \frac{\partial y}{\partial v_m}\right]^T \tag{5.8}$$

where m is the number of vertices in the structure.

5.3.2 Normal translation

The gradient is computed w.r.t. translation of a single block face along the face normal $\hat{\mathbf{e}}_n$. The gradient $\frac{\partial y}{\partial n}$ is computed for each block face in the structure. The derivatives

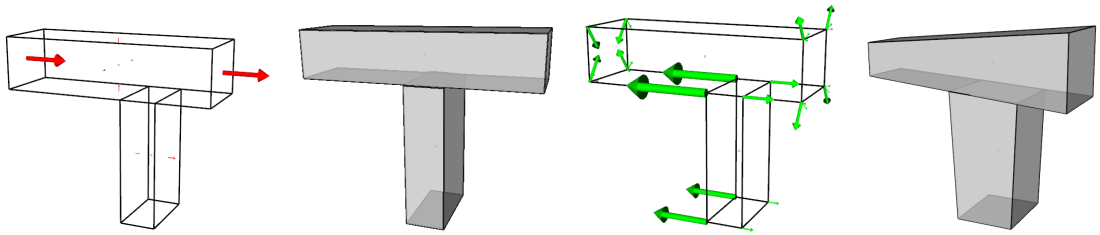
of vertex position, $\mathbf{p}$, are

$$\partial \mathbf{p}/\partial n = \hat{\mathbf{e}}_n \tag{5.9}$$

Again, derivatives of the face coordinate system are zero since the face normal is constant: $\partial \hat{\mathbf{e}}_i/\partial n = \mathbf{0}$ for $i = n, u, v$. The vectors $\hat{\mathbf{e}}_u$, $\hat{\mathbf{e}}_v$ and $\hat{\mathbf{e}}_n$ vectors are not modified. The component of the gradient $\nabla y(\Omega)$ for normal translations is

$$\nabla_n y(\Omega) = \left[\frac{\partial y}{\partial n_1} \dots \frac{\partial y}{\partial n_\ell}\right]^T \tag{5.10}$$

where ℓ is the number of faces in the structure.



(a) Face gradients for translation along normal (b) Improved geometry (c) Vertex gradients for in-plane translation (d) Improved geometry

Figure 5-3: Parameterized gradients for an asymmetrical T-model. Instability arises from unbalanced torque caused by the left overhanging portion of the T. (a,c) Gradients point in the direction of improved stability. (b,d) Stable output produced by moving the vertices along the gradient.

5.3.3 Normal Rotation

Two angle parameters, θ and ϕ, are the basis for rotation of the face plane normal. We set the center of rotation at the centroid, $\mathbf{c}$, of the block face. Gradients $\frac{\partial y}{\partial \theta}$ and $\frac{\partial y}{\partial \phi}$ are computed for each block face in the structure. The derivatives of the normal vector and vertex positions are

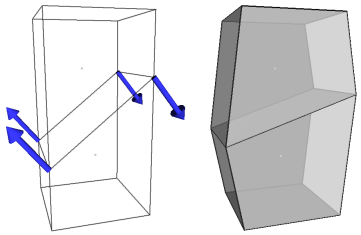
$$\begin{aligned} \partial \mathbf{p}/\partial \theta &= \hat{\mathbf{e}}_v \times (\mathbf{p} - \mathbf{c}) \\ \partial \mathbf{p}/\partial \phi &= \hat{\mathbf{e}}_u \times (\mathbf{p} - \mathbf{c}) \end{aligned} \tag{5.11}$$

$$\partial\hat{\mathbf{e}}_n/\partial\theta = \hat{\mathbf{e}}_u, \quad \partial\hat{\mathbf{e}}_u/\partial\theta = \hat{\mathbf{e}}_n, \quad \partial\hat{\mathbf{e}}_v/\partial\theta = \mathbf{0}$$

$$\partial\hat{\mathbf{e}}_n/\partial\phi = \hat{\mathbf{e}}_v, \quad \partial\hat{\mathbf{e}}_u/\partial\phi = \mathbf{0}, \quad \partial\hat{\mathbf{e}}_v/\partial\phi = \hat{\mathbf{e}}_n$$

The components of the gradient $\nabla y(\Omega)$ for normal rotations are

$$\nabla_\theta y(\Omega) = \left[\frac{\partial y}{\partial\theta_1} \cdots \frac{\partial y}{\partial\theta_\ell}\right]^T, \ \nabla_\phi y(\Omega) = \left[\frac{\partial y}{\partial\phi_1} \cdots \frac{\partial y}{\partial\phi_\ell}\right]^T \tag{5.12}$$



(a) Vertex gradients for face rotation

(b) Improved geometry

Figure 5-4: Gradients computed w.r.t. face plane rotation show how the orientation of interfaces between blocks affects the stability. In this 2-block example, rotating the interface toward a horizontal orientation reduces the infeasibility due to friction failure.

5.3.4 Gradient w.r.t. Vertex Position

At vertex $\mathbf{p}_i$, the partial derivative of the energy y with respect to the vertex position is expressed as the sum:

$$\frac{\partial y}{\partial \mathbf{p}_i} = \sum_k \left(\frac{\partial y}{\partial u_{k,i}} \frac{\partial u_{k,i}}{\partial \mathbf{p}_i} + \frac{\partial y}{\partial v_{k,i}} \frac{\partial v_{k,i}}{\partial \mathbf{p}_i} + \frac{\partial y}{\partial n_k} \frac{\partial n_k}{\partial \mathbf{p}_i} + \frac{\partial y}{\partial \theta_k} \frac{\partial \theta_k}{\partial \mathbf{p}_i} + \frac{\partial y}{\partial \phi_k} \frac{\partial \phi_k}{\partial \mathbf{p}_i} \right)$$

over all block faces k that contain vertex i. The partial derivatives of the parameters with respect to vertex position are:

$$\frac{\partial u_{k,i}}{\partial \mathbf{p}_i} = \hat{\mathbf{e}}_{u_k}, \quad \frac{\partial v_{k,i}}{\partial \mathbf{p}_i} = \hat{\mathbf{e}}_{v_k}, \quad \frac{\partial n_k}{\partial \mathbf{p}_i} = \frac{\hat{\mathbf{e}}_{n_k}}{\#verts_{\text{face k}}}$$

$$\left[\frac{\partial \theta_k}{\partial \mathbf{p}_i} \cdots \frac{\partial \theta_k}{\partial \mathbf{p}_{i+n_k}}\right] = \left[\frac{\partial \mathbf{p}_i}{\partial \theta_k} \cdots \frac{\partial \mathbf{p}_{i+n_k}}{\partial \theta_k}\right]^{-1}$$

$$\left[\frac{\partial \phi_k}{\partial \mathbf{p}_i} \cdots \frac{\partial \phi_k}{\partial \mathbf{p}_{i+n_k}}\right] = \left[\frac{\partial \mathbf{p}_i}{\partial \phi_k} \cdots \frac{\partial \mathbf{p}_{i+n_k}}{\partial \phi_k}\right]^{-1}$$

To summarize, the gradient of the infeasibility metric $\nabla y(\Omega)$ is a vector of length N, where $N = (\#\text{block faces}) \times (3 + 2 \times \#\text{vertices on each face})$.

To illustrate the parameterization, Figure 5-3 shows the gradient for normal and in-plane translations on an unstable two-block T structure, and Figure 5-4 shows the gradient for normal rotation on a 2-block structure with an inclined interface. Our gradient computation takes into account two properties:

1. **Mass:** the top block reduces imbalanced torque by widening vertices on the right, and thinning vertices on the left.

2. **Joint Geometry:** Increasing the width of the bottom block decreases the overhang of the top block, thus decreasing instability. This works because joint vertices (i.e. contact polygon between adjacent blocks) have a positional derivative that is computed along with the block vertex derivatives. Joint vertices define where the contact forces are applied.

5.3.5 Geometric Implementation

Partial derivatives are computed for each block face. A basic assumption is that topology remains consistent under a differential change in the block geometry: adjacent blocks will remain in contact, disconnected blocks remain disconnected, and the number of vertices in each contact polygon will remain the same. These assumptions are necessary to ensure the number of variables is constant (recall that the size of the system force vector is proportional to the number of vertices over all contact

polygons), which is required for the quadratic program $y(\Omega)$ to be differentiable.

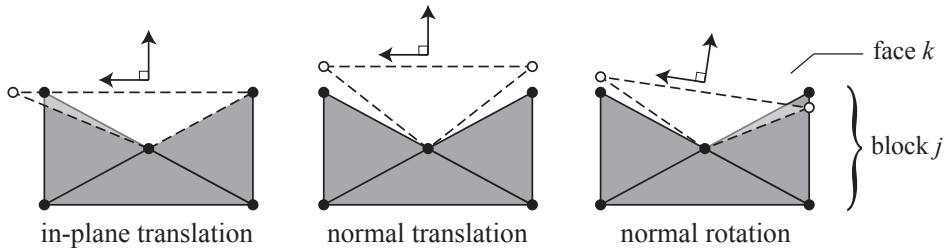


Figure 5-5: Geometric interpretation of the partial derivatives for each parameterization, illustrated in 2D. Only the tetrahedrons incident to the modified face are affected.

For example, in computing the derivative of the equilibrium matrix $\partial \mathbf{A}_{eq}/\partial n_k$, only the rows associated with block j are non-zero (any adjacent blocks are unaffected). Likewise, for any block face $\ell \neq k$, the local coordinate frame $(\hat{\mathbf{e}}_{n_\ell}, \hat{\mathbf{e}}_{u_\ell}, \hat{\mathbf{e}}_{v_\ell})$ and joint vertices remain constant. Derivatives of block weight and centroid position are computed for the quadrilateral pyramid connecting the four vertices of face k to the original centroid position (Fig. 5-5).

5.4 Modes of Interaction

Given the gradients of infeasibility w.r.t. the chosen parameterizations, we now show how the user can apply the gradients as a guide for improving the stability of a structure while preserving properties of its design. We frame the task as an optimization problem that steps towards a more structurally feasible design while allowing the user to specify constraints on the desired shapes and range of acceptable changes.

Our approach to user control contrasts significantly with other shape design and optimization methods. Comparisons can be broadly grouped into three categories:

1. Only Analysis: Gradients are not provided, instead it is up to the user to determine how a shape can be modified to improve on the design objective. This is the typical approach for structural analysis software – e.g. finite element analysis software returns a stress profile of the input shape, but in order to reduce stress, the user must modify the geometry by trial and error.

2. Black-box Optimization: An optimization runs according to the optimal strategy, such as steepest descent gradient-based methods. User control is limited to inputs at the initiation of the optimization. This was the approach taken in chapter 4.

3. Guided Optimization: Our approach in this chapter is to let the user guide the path of the optimization in numerous ways. They may apply a new set of constraints at any point in the optimization, modify the step size at each iteration, or terminate the optimization early if an appealing design has been reached.

5.4.1 Snapping To Gradient

The first type of interaction is to modify vertex positions in the direction of the computed gradient. At each vertex in the structure, we compute a displacement vector $\Delta\mathbf{p}$ that combines all contributions from in-plane movement, face translation and face rotation. We define an optimization that solves for vertex positions best matching the desired gradient vector while maintaining planarity constraints of the block faces and coincidence constraints of contiguous blocks.

$$\begin{aligned} \mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \ & \|\mathbf{p} - \mathbf{p}_0 + \Delta\mathbf{p}\|^2 \\ \text{s.t.} \ & \text{faces remain planar} \\ & \text{block interfaces remain coincident} \end{aligned}$$

Under the assumption of quad-faced blocks, we adopt a planarity constraint that enforces the angles of each face to sum to 2π [Liu et al. 2006]:

$$f_{planar}(\mathbf{p}) = \sum_F \|\phi_F^1 + \ldots + \phi_F^4 - 2\pi\|^2$$

Our coincidence constraint ensures that each pair of coincident faces $(F, G) \in C$ between two adjoining blocks remains connected:

$$f_{coin}(\mathbf{p}) = \sum_{(F,G)\in C} \sum_{j\in 1...4} \|\hat{\mathbf{n}}_F(p_G^j - p_F^1)\|^2$$

Due to the nonlinearity of these constraints, we enforce them using a penalty function technique.

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \|\mathbf{p} - \mathbf{p}_0 + \Delta\mathbf{p}\|^2 + \lambda_1 f_{planar}(\mathbf{p}) + \lambda_2 f_{coin}(\mathbf{p})$$

We use a Gauss-Newton iteration approach to solve for the best vertex positions satisfying our constraints. Characteristic to penalty techniques, the constraints are satisfied within a set tolerance. In practice we set this tolerance to 5e-3, which can be modified with trade-offs between accuracy and performance.

5.4.2 User Control

While the full gradient provides useful information on how a structure can be optimally modified to improve feasibility, a user may be interested in incorporating constraints that express design intent. Our system features a set of UI controls for setting high-level constraints, such as block thickness or floor orientation, which allow the user to explore a range of designs. We describe how these controls can be implemented by modifying the gradient with low-level constraints on point positions and face normals.

Fix block thickness This is achieved by constraining the distance between two opposing faces (F, G) of a block to be constant.

$$f_{thickness}(\mathbf{p}) = \sum_{(F,G)} \sum_{j=1...4} \left\| th(F, G, j, \mathbf{p}) - th(F, G, j, \mathbf{p}_0) \right\|^2$$

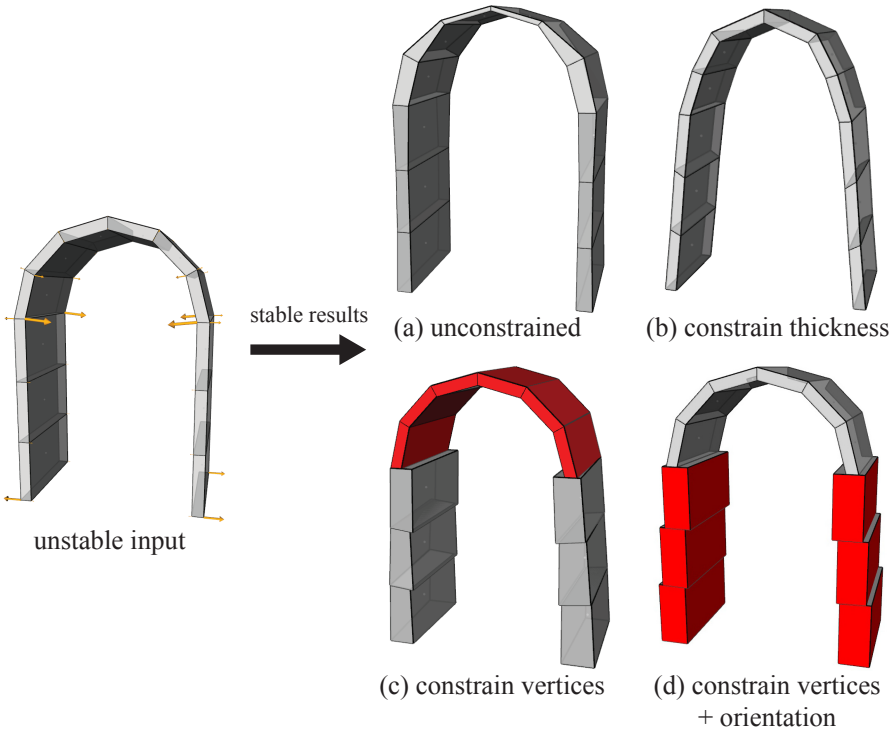


Figure 5-6: The user may explore possible designs with improved stability by modifying constraints on the gradient. The initial arch is unstable, arrows indicate gradients for improving feasibility. The initial gradient is shown (∇y_0). (a) Feasible unconstrained result. (b) Feasible output with constrained thickness, total #iterations = 3. (c) The arch blocks are fixed resulting in slightly tapered column blocks. (d) In addition to constrained arch vertices, the column blocks have fixed orientation.

where $th(F, G, j, \mathbf{p}) = \hat{\mathbf{n}}_F \cdot (\mathbf{p}_F^j - \mathbf{p}_G^j)$. In Figure 5-6(b) the arch has constrained thickness, resulting in a catenary-like output.

Fix vertices This is implemented by setting gradients of unaffected vertices to zero and pinning them to a specific location in the optimization. Alternatively, we can remove these vertices from the optimization altogether. In Figure 5-6(c) the arches are kept constant, only column vertices are modified.

$$f_{point}(\mathbf{p}) = \sum_{(i,\mathbf{p}')} \left\| \mathbf{p}^i - \mathbf{p}' \right\|^2$$

Fix face orientation We fix horizontal floors and vertical walls by zeroing rotational gradients and constraining a face to have a specific normal vector. For example,

in Figure 5-6(d) all column block faces are orientation-constrained to maintain vertical faces.

$$f_{normal}(\mathbf{p}) = \sum_{(F,\hat{\mathbf{n}})} \sum_{j=2\ldots4} \left| \hat{\mathbf{n}} \cdot \frac{\mathbf{p}_F^j - \mathbf{p}_F^1}{\|\mathbf{p}_F^j - \mathbf{p}_F^1\|} \right|^2$$

Fix block volume To constrain volume, we divide each block i into five tetrahedrons $T_{i,j}$ and constrain the sum of the volumes of the five tets to remain constant.

$$f_{volume}(\mathbf{p}) = \sum_i \left\| \sum_{j=1\ldots5} v_{T_{i,j}}(\mathbf{p}) - v_{T_{i,j}}(\mathbf{p}_0) \right\|^2$$

where $v_T(\mathbf{p})$ is the volume of the tetrahedron T given vertex positions $\mathbf{p}$.

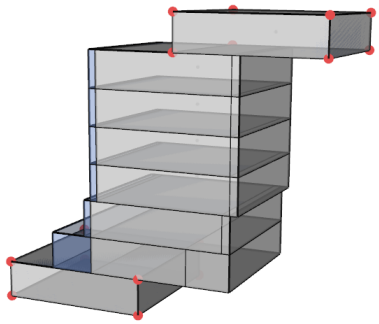
Additionally, we can achieve other design constraints through combination of the penalty functions. For example, restricting a block to rigid translation (Figure 5-9(b)) involves applying a thickness and orientation constraint to each pair of opposing faces.

5.4.3 User Objective Functions

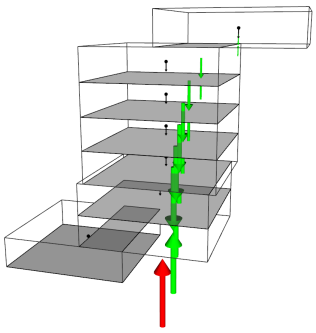
In addition to constraints on the gradient direction, the user may incorporate other design criteria by setting custom objective functions. We demonstrate this capability with a volume minimization example. The new multi-objective function becomes a weighted combination of the infeasibility metric g and the total volume v.

$$\nabla y(\Omega) = \nabla g(\Omega) + \gamma \nabla v(\Omega)$$

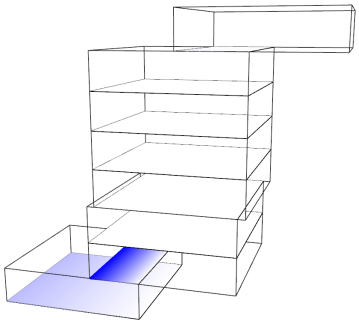
where γ is a weighting on the volume minimization. The derivative of v w.r.t. each parameterization is identical to that used for weight in the $\mathbf{w}$ vector without the constant term for density (see §A.2). Figure 5-8 shows an example of this objective applied to a structure of stacked blocks.



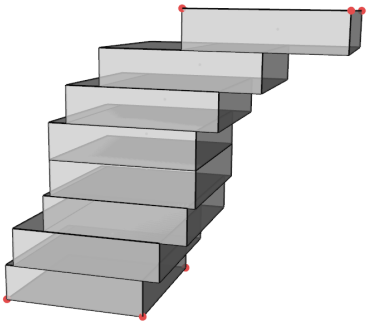
(a) Infeasible input with constraints



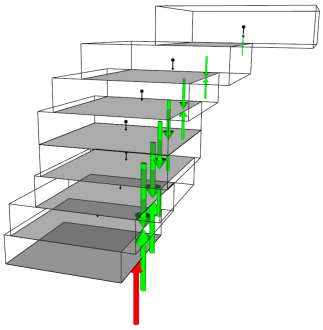
(b) Infeasible input displaying resultant forces



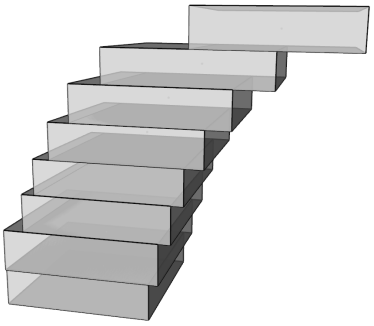
(c) Infeasible input displaying tension faces (blue)



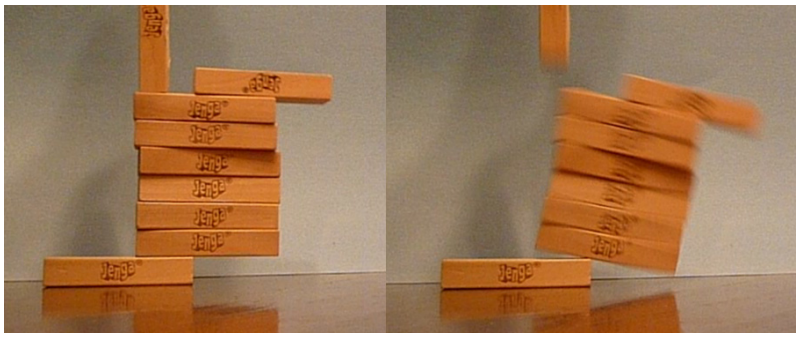
(d) Feasible output constrained to rigid translation



(e) Feasible output displaying resultant forces

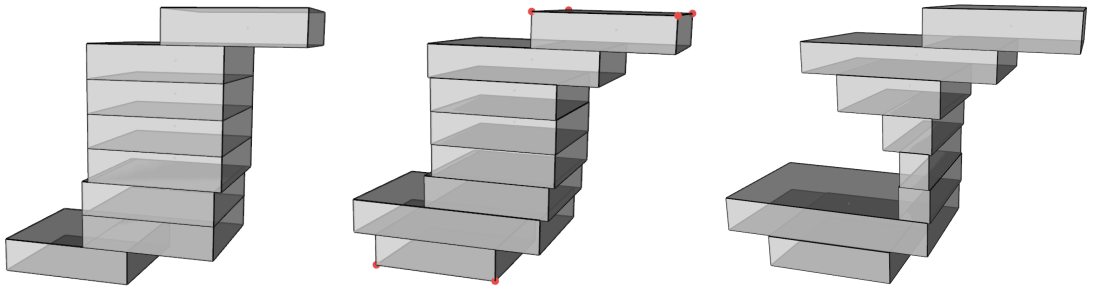


(f) Alternate feasible output with 5% factor of safety



(g) Unstable input falls after release

Figure 5-7: (a-c) Input: unstable stack of 8 identical blocks. The base and overhanging vertices are fixed (red) and the remaining blocks are constrained to rigid translation. The ghost geometry (blue) displays the first iteration along the gradient. (d,e) A stable structure after multiple iterations. All resultant forces lie inside the block interfaces. (f) An alternate solution with a small safety factor by shrinking interfaces 5%. (g) A physical model of the unstable input stacking. The hinge point at failure corresponds to the interface with greatest tension (intensity of blue shading in (c)).



(a) Input (b) without volume minimization (c) with volume minimization

Figure 5-8: Variations based on volume minimization. In both cases the constraints are face orientation and vertical thickness (horizontal stretching is permitted): (a) Infeasible input, weight = 160 units. (b) Feasible output without volume minimization, weight = 174 units. (c) Stable output with volume minimization, weight = 127 units, initial $\gamma_0 = 0.5$.

5.5 Results

5.5.1 Implementation

We use the IBM CPLEX quadratic program solver. Intel MKL is used for nonlinear optimization of block planarity under user constraints.

Numerical Stability In section 5.2.3 it is known that solving the normal equations $\mathbf{x} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{b}$ explicitly is prone to numerical errors [Trefethen and Bau 1997]. For improved numerical stability we use QR decomposition to compute the inverse $\mathbf{E}^{-1} = (\mathbf{C}\mathbf{H}^{-1}\mathbf{C}^T)^{-1}$. The decomposition is $\mathbf{Q}\mathbf{R} = (\mathbf{C}\mathbf{H}^{-1/2})^T$ where $\mathbf{Q}$ has orthonormal columns and $\mathbf{R}$ is upper triangular. It can be shown that $\mathbf{E} = (\mathbf{C}\mathbf{H}^{-1/2})(\mathbf{C}\mathbf{H}^{-1/2})^T = \mathbf{R}^T\mathbf{R}$, so that $\mathbf{E}^{-1}$ can be solved by forward and back substitution of $\mathbf{R}$ rather than explicitly forming $\mathbf{E}$ and its inverse. The pseudoinverse can be computed as $(\mathbf{C}\mathbf{H}^{-1/2})^+ = \mathbf{Q}\mathbf{R}^{T^{-1}}$ with $\mathbf{Q}$ and $\mathbf{R}$ in reduced form [Trefethen and Bau 1997].

Matrix Multiplication Performance depends heavily on the order of matrix multiplications when computing the energy derivatives in §5.2.3. In practice, operations are ordered to favor matrix-vector multiplications, and so that matrix inverses are

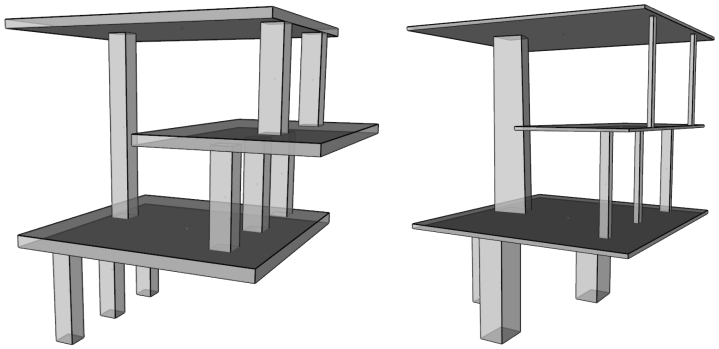
solved for vector variables.

5.5.2 Modeling Stable Structures

We show example structures with visualizations of the computed infeasibility gradients, and various ways in which the user might modify the structure.

Figure 5-6 shows a variety of outputs for an infeasible model of an arch balanced on two columns. We compare results from unconstrained optimization, constant thickness and fixing the arch vertices.

Figures 5-7 and 5-8 show various results for an unstable stack of blocks. In 5-7 the constraints include rigid translation using a combination of thickness and orientation constraints. We show visualizations of the resultant forces at each interface, illustrating that resultants lie within the joints for the feasible result. We also apply a small factor of safety by shrinking the effective interfaces between blocks. Figure 5-8 applies only orientation constraints and vertical thickness, so that the blocks can stretch horizontally. An alternative design using volume minimization is provided.



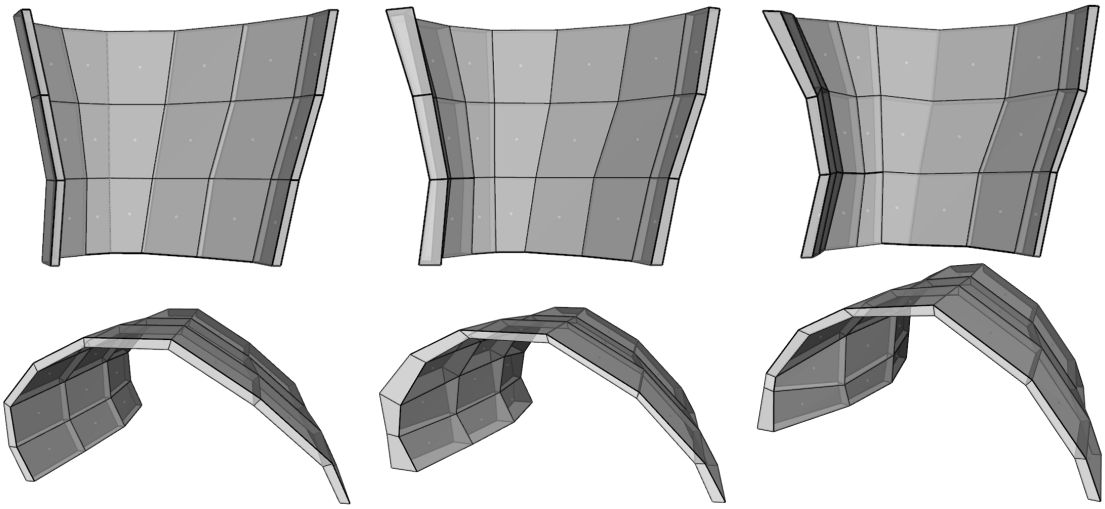
(a) Unstable input

(b) Feasible output (rigid base columns)

Figure 5-9: (a) An infeasible building is shown with columns supporting a system of slabs (y_0 = 7.65e5). (b) Stable output with fixed floors and rigidity constraints on the base columns (y^* = 8.6e-7).

In Figure 5-9, the clearly unstable input model consists of three slabs and a series of columns. The result in Figure 5-9(b) had constraints on slabs orientation, and rigid translation of the base columns. There is noticeable thinning and shortening of

the slabs to reduce weight. Columns on the over-hanging portion are thinned while the left-hand column is thickened for counter-balance. The slabs and columns were fixed part-way through the design when a minimum preferred thickness was reached. The base columns are translated to the center of the floorplan so that the resultant load of the upper levels falls within their support region.

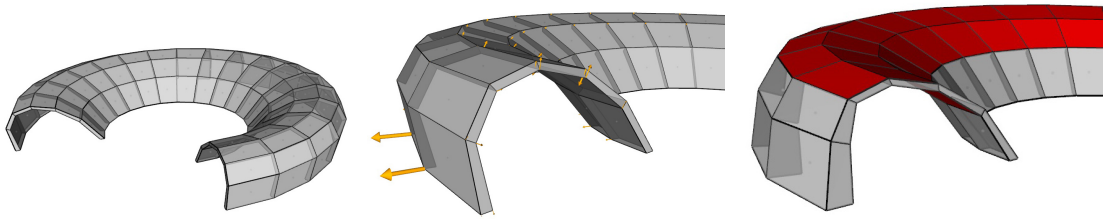


(a) Infeasible input (b) Feasible output (unconstrained) (c) Feasible output with thickness constraints

Figure 5-10: Various modification starting from a shell that bulges over the supports (a) The infeasible input ($y_0 = 25.8$). (b) Unconstrained feasible output ($y^* = 1.4\text{e-}4$). (c) Feasible output with all blocks constrained to constant thickness ($y^* = 1.5\text{e-}4$).

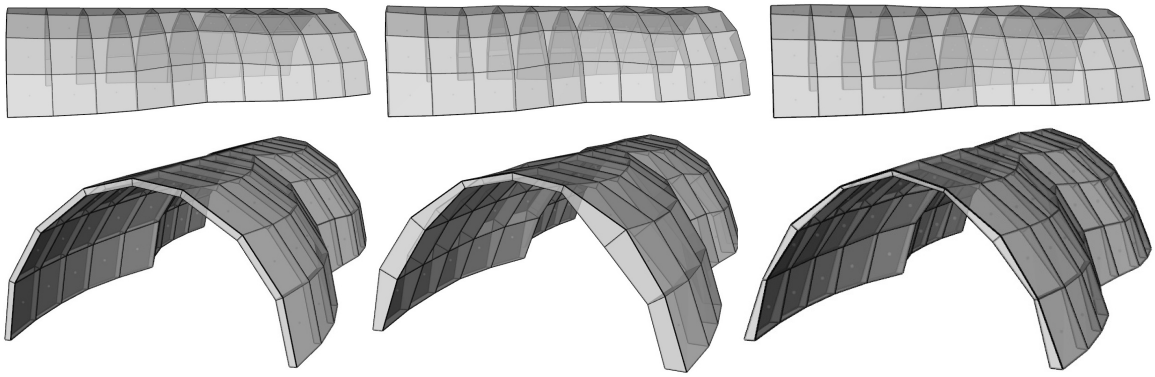
In Figure 5-10(a) the shell is infeasible due to a bulging shape over the support region. Result 5-10(b) shows an output from unconstrained optimization where the profile is significantly thickened. In 5-10(c), the thickness of the shell is constrained, resulting in a modified shape closer to a traditional arch. In Figure 5-11 the shell is extended to a partial torus as a surface of revolution.

The starting shape in Figure 5-12 is an extruded arch profile with varying floor span. The unconstrained result significantly thickens the blocks. Adding a constraint to maintain the volume of all blocks results in a feasible structure by stretching the profile toward a catenary shape. Also visible in the side view is that the height of the feasible structure undulates slightly along its length. Both feasible results were generated with a single gradient step.



(a) (b) (c)

Figure 5-11: (a) An infeasible model consisting of a bezier curve surface of revolution. (b) Yellow arrows represent the gradient. (c) The feasible result thickens the base blocks supporting the overhanging outer curve. Red blocks were constrained in thickness.



(a) Infeasible input (b) Output: unconstrained (c) Output: constrained volume

Figure 5-12: A semi-circular arch profile with varying span (front and side views). (a) The infeasible input model. (b) Feasible output unconstrained. (c) Feasible output with each block constrained to constant volume.

5.5.3 Performance

Our formulation for analytic gradients provides a significant performance improvement over finite differencing techniques, which is necessary for interactive applications. See Table 5.1 for results and comparison. Performance times for finite differencing are equivalent to the time for solving the quadratic program (§3.4) multiplied by the number of components in the gradient vector ∇y. Times for analytic gradients include the time for computing partial derivatives (§5.2) added to the time for computing planar vertex offsets (§5.4.1). While computing the analytic derivatives takes some time, it is orders of magnitude faster than running the quadratic program solver multiple times as in the finite differencing approach.

Model	Blocks	length(∇y)	Analytic gradient	Finite diff.
arch (Fig. 5-6)	12	792	0.95 s	127 s
block stack (Fig. 5-7)	8	528	0.75 s	84.5 s
shell (Fig. 5-10)	24	1584	1.55 s	269 s
cut torus (Fig. 5-11)	108	7128	14.8 s	1920 s

Table 5.1: Performance results for computing the analytic structural gradient.

The plot in Figure 5-13 shows the progress of the infeasibility metric for an sample optimization. The structure being modified is the unconstrained arch from Figure 5-6. At each iteration the value of infeasibility decreases, and converges toward $y(\Omega) = 0$ (feasibility condition) after a small number of gradient steps.

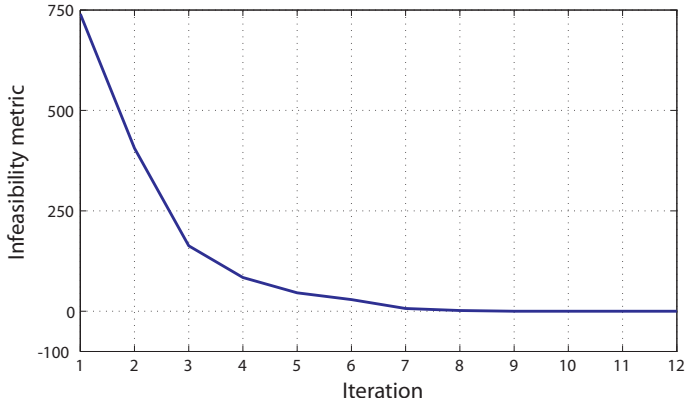


Figure 5-13: Plot of Infeasibility vs. Iteration for the unconstrained arch (Fig. 5-6(a)). Infeasibility values: $y_0 = 740$, $y_{final} = 1.5\text{e-}4$.

5.5.4 Validation

We tested the accuracy of the analytic gradient by comparing results from finite differencing. In the 2-block T model of Figure 5-3, we applied a small delta to select vertices, and compared the change in constraint matrices $\Delta\mathbf{A}_{eq}$ and $\Delta\mathbf{w}$ with the analytic result $(\Delta\mathbf{A}_{eq})_{analytic} = \Sigma_i(d\mathbf{A}_{eq}/d\mathbf{p}_i) \cdot \Delta\mathbf{p}_i$ over selected vertices i. Varying groups of vertices were tested with shifts along x, y, z axes. In all cases the error was below 1%, with error measured as:

$$\%\text{ error} = \|(\Delta A_{eq})_{findiff} - (\Delta A_{eq})_{analytic}\|/\|(\Delta A_{eq})_{findiff}\|$$

In the 2-block model of Figure 5-4, we measured the energy change from a small shift in the angle of the interface. The result of a forward finite difference was accurate to 2.3% of the analytic value for ∇y.

5.5.5 Limitations

Planarity and user constraints may compete with the displacement vector $\Delta\mathbf{p}$ and in some cases may have the effect of pushing vertices away from the gradient direction. In this case $\Delta\mathbf{p}$ may decrease the feasibility of the structure.

Also notable is that the gradient $\nabla y(\Omega)$ is dependent on the choice of active constraints by the QP solver. Recall the assumption that active constraints remain active in the neighborhood of the current solution for $\mathbf{f}_\Omega$. As a consequence, if the normal force at a vertex i is zero (i.e. lower bounds on f_n^{i+}, f_n^{i-} are both active), then the partial derivatives $\partial f_n^{i-}/\partial\omega$ and $\partial f_n^{i+}/\partial\omega$ will also be zero.

5.6 Discussion

We introduce a technique that can lead to a variety of feasible designs given the same geometric input, based on the user's decisions. We provide a suite of user-controlled constraints that guide the optimization of a structure according to aesthetic preferences.

Our method is general – the gradient operates on arbitrarily shaped structures composed of quad-faced blocks. We require no higher-level parameterization of the model. This allows a greater flexibility for the variety of input models, compared to the procedural modeling technique in Chapter 4. Moreover, our analytic formulation for gradients improves performance significantly over finite differencing techniques, which enables interactive applications.

Chapter 6

Conclusion

6.1 Contributions

This thesis has introduced consideration of structural stability in computer-aided design for architecture. Three primary contributions were presented in the areas of masonry analysis and modeling of structurally feasible architectural geometry:

- **Measure of Infeasibility** We presented a measure of infeasibility that determines how close a model is to being structurally sound. We introduced a penalty formulation to represent the presence of tension in rigid block assemblages. The new metric is enabled by a quadratic programming formulation and agrees closely with available theoretical results.

Based on this new method for quantifying infeasibility, two approaches were introduced applying the metric to structural design:

- **Procedural Modeling** We introduced the idea of generating structurally feasible *procedural* models of buildings through automatic parameter selection. The measure of infeasibility was used as an energy function, and gradient-based optimization was applied to select rule parameters that satisfy structural stability constraints. We showed examples of procedural models of buildings with both internal and external structure that are consistent with mechanics.

- **Analytic Gradient** We provided a closed form derivation for the gradient of the measure of infeasibility with respect to geometry modification. The geometry is parameterized at a vertex level, removing the dependence on procedurally specified geometry. The performance enhancement offered by the analytic formulation over finite differencing enables a variety of efficient interactive tools. We presented prototype design tools for improving the stability of a model based on user-provided constraints and objectives. We demonstrated that our technique can lead to a variety of structurally-sound designs given the same input shape, based on the user's decisions.

6.2 Future Work

This thesis has contributed new methods for incorporating structural analysis into the design of architectural models. A few examples of remaining open questions in structural design and masonry analysis are as follows:

6.2.1 Analysis of Laser Scan Data

In order to make analysis and design tools more widely applicable, it would be valuable to integrate them with automated processes for modeling existing structures. Laser range scan technology is capable of capturing accurate geometry for cathedrals and other complex structures. However, there are several key challenges in applying our methods to the captured geometry:

- Point cloud model formats make no distinction between materials to indicate non-structural components (e.g. glass windows) versus structural elements (e.g. solid stone wall).

- Assumptions may have to be made about the thickness of vaults, ribs, or other structural details where portions of the bounding surfaces are not visible to the scanner. For example, it is common for gaps to exist between a vaulted ceiling

and the roof, such that no sight lines exist to the extrados of the ceiling from either the interior or the exterior of the building.

- There is a lack of texture data to indicate interfaces between individual blocks. Texture-mapped models are sometimes available, but the image resolution may not be fine enough to discern block edges, and image registration with the model may be inaccurate.

- Laser scans also suffer from imperfect conditions for data acquisition. Occlusions cause gaps in the mesh (shadows in Fig.6-1), and proximal objects such as scaffolding are sometimes included in the scans.

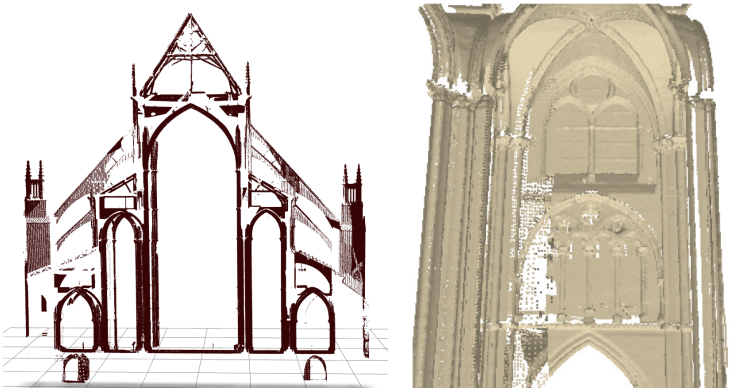


Figure 6-1: Detail of a laser range scan of Bourges Cathedral. [*Source: A. Tallon, Vassar College*].

6.2.2 Stability under Displacements

In historic structures, the capacity for displacements is important in addition to load capacity. While most engineering analysis measures safety by the fraction of additional load that can be applied before failure, for historic masonry structures it is necessary to understand the effect of movement on the stability of the structure since large displacements are common. For example, settlements in foundations and consolidation of materials occur over time. An important direction of future work would be to characterize the range of possible movements and the effects of those displacements in a mathematical programming framework.

6.2.3 Intuitive Design Interfaces

Further work could be done on parameterizations that contain information about important features of a building design, without requiring tedious input from the user. For example, Igarashi and Hughes [2001] developed 3D drawing tools enabling users to manipulate designs with symmetries and repeated substructures. Lipp et al. [2008] visualized editable parameters in procedural models, and provided the capability to directly manipulate parameters for more intuitive design exploration. It would be valuable to combine the manipulation and visualization of these two methodologies with the structural design approaches presented in this thesis.

6.2.4 Friction Failure

The lower bound theorem [Drucker 1954] states that if a static equilibrium solution exists to support a structure under loads $\mathbf{w}$, then the structure is safe and $\mathbf{w}$ is a lower bound for the collapse load. However, this theorem applies only to hinging failure under tension. Further investigation is necessary to determine safety under friction failure.

6.2.5 Discretization

Using a coarse discretization (large blocks representing many smaller units) may over-estimate the stability of the final structure. Our method analyses structures by finding force values at the boundaries of blocks. This is an approximation to the force distribution within a structure, and it is possible for tension forces to develop as the discretization is refined. We tested the Sainte Chapelle model (Figure 4-6) by choosing parameters that were "just stable" (small variations make the structure unstable), then varied the number of blocks. The chapel remained feasible when we increased the block count from 486 to 876 by subdividing the columns, arched windows, and circular window. The chapel became infeasible when we further subdivided the groin vaulted ceiling. The trade-off between accuracy and computation speed is an area for future investigation.

6.2.6 Alternative Energy Functions

In chapter 3 we proposed a method for quantifying infeasibility of masonry structures based on the magnitude of tension forces. Although tension is an indication of infeasibility, it would be valuable to consider other measures of infeasibility. One possibility is to measure distance of each resultant force from the boundary of its incident block face, as depicted in Figure 3-2. However, difficulty arises with pure tension faces since the resultant force is negative and the position remains inside the face despite being infeasible. A second possibility is to measure torque contributions from tension forces, however this is relevant only when hinging is the failure mechanism. Further investigation is needed to determine the effectiveness of alternative energies.

Appendix A

Static Equilibrium Constraints

A.1 Matrix Structure

We detail the matrix equation for static equilibrium (3.1). 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. 3.5), 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.

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

where $\mathbf{f}^i = [f_n^i \ f_u^i \ f_v^i]^T$, $\mathbf{a}_{k_x} = [\hat{\mathbf{e}}_{n_{k_x}} \ \hat{\mathbf{e}}_{u_{k_x}} \ \hat{\mathbf{e}}_{v_{k_x}}]$ and $\mathbf{b}_{i,j,k_x} = [(\hat{\mathbf{e}}_{n_k} \times \mathbf{v}_{i,j})_x \ (\hat{\mathbf{e}}_{u_k} \times \mathbf{v}_{i,j})_x \ (\hat{\mathbf{e}}_{v_k} \times \mathbf{v}_{i,j})_x]$. Unit vectors $\hat{\mathbf{e}}_{n_k}$, $\hat{\mathbf{e}}_{u_k}$ and $\hat{\mathbf{e}}_{v_k}$ are the normal vector and friction basis vectors for face k (see Figure A-1). 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.

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.

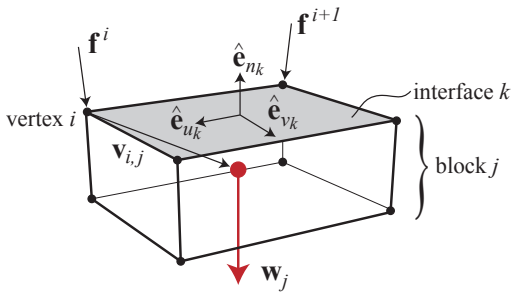


Figure A-1: Indexing for equations of static equilibrium. Vector $\hat{\mathbf{e}}_{n_k}$ is the unit normal for interface k, and $\hat{\mathbf{e}}_{u_k}$ and $\hat{\mathbf{e}}_{v_k}$ are the directions of in-plane friction forces. Vector $\mathbf{v}_{i,j}$ is the relative position of vertex i w.r.t. the centroid of block j. $\mathbf{w}_j$ is the 3D weight vector for block j.

The number of non-zero elements in each row j is $\sum_k v_k(\#\textit{forces per vertex})$, over all interfaces k on block j.

$\mathbf{A}_{fr}$: square with dimension = $\textit{length}(\mathbf{f})$. 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.

A.2 Partial Derivatives

The partial derivatives of coefficients for net force equilibrium, $\mathbf{F}_k$, on face k are:

$$\frac{\partial \mathbf{F}_k}{\partial \omega} = \frac{\partial [\hat{\mathbf{e}}_n \ \hat{\mathbf{e}}_u \ \hat{\mathbf{e}}_v]_k}{\partial \omega}$$

where ω is a parameter from the set $\{u_{i,k}, v_{i,k}, n_k, \theta_k, \phi_k\}$ as described in §5.3. The partial derivatives of coefficients for net torque equilibrium, $\mathbf{T}_k$, on face k are:

$$\begin{aligned}
\frac{\partial \mathbf{T}_{i,j,k}}{\partial \omega} &= \frac{\partial [(\hat{\mathbf{e}}_n \times \mathbf{v}_{i,j}) \ (\hat{\mathbf{e}}_u \times \mathbf{v}_{i,j}) \ (\hat{\mathbf{e}}_v \times \mathbf{v}_{i,j})]_k}{\partial \omega} \\
\frac{\partial (\hat{\mathbf{e}}_n \times \mathbf{v}_{i,j})_k}{\partial \omega} &= \hat{\mathbf{e}}_n \times \frac{\partial \mathbf{v}_{i,j}}{\partial \omega} + \frac{\partial \hat{\mathbf{e}}_n}{\partial \omega} \times \mathbf{v}_{i,j} \\
&= \hat{\mathbf{e}}_n \times \left(\frac{\partial \mathbf{p}_{i,j}}{\partial \omega} - \frac{\partial \mathbf{c}_j}{\partial \omega} \right) + \frac{\partial \hat{\mathbf{e}}_n}{\partial \omega} \times \mathbf{v}_{i,j}
\end{aligned}$$

The derivative of the centroid position $\mathbf{c}_j$ for block j is:

$$\frac{\partial \mathbf{c}_j}{\partial \omega} = \frac{\partial}{\partial \omega}\left(\frac{\sum v_{T_{i,j}} \mathbf{c}_{T_{i,j}}}{\sum v_{T_{i,j}}}\right)$$

where $v_{T_{i,j}}$ is the volume of tetrahedron i on block j and $\mathbf{c}_{T_{i,j}}$ is the centroid of tetrahedron i.

$$\begin{aligned}
\frac{\partial v_{T_{i,j}}}{\partial \omega} &= \frac{1}{6} sign(\mathbf{a}_0 \cdot (\mathbf{a}_1 \times \mathbf{a}_2)) \frac{\partial}{\partial \omega}(\mathbf{a}_0 \cdot (\mathbf{a}_1 \times \mathbf{a}_2)) \\
\frac{\partial \mathbf{c}_{T_{i,j}}}{\partial \omega} &= \frac{1}{4} \frac{\partial}{\partial \omega}(\mathbf{a}_0 + \mathbf{a}_1 + \mathbf{a}_2)
\end{aligned}$$

where coordinates $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2$ are three corners of tetrahedron i, offset such that the fourth coordinate lies at the origin $(0, 0, 0)$.

The derivative of the weight vector $\partial \mathbf{w}/\partial \omega$ for block j is given by:

$$\frac{\partial \mathbf{w}_j}{\partial \omega} = \rho \frac{\partial v_j}{\partial \omega} \hat{\mathbf{g}} = \rho \left(\sum_i \frac{\partial v_{T_{i,j}}}{\partial \omega}\right) \hat{\mathbf{g}} \tag{A.1}$$

where ρ is the block density and $\hat{\mathbf{g}}$ is the direction of gravity, and v_j is the volume of block j.

Appendix B

Data: Static Analysis

For comparison in future implementations and experiments, we provide analysis results for the 2-block T model in Figure 3-4. The geometry is specified in obj format, and following we provide force values for all block interfaces.

T model geometry (obj format):

```
v 0.0000000000 0.0000000000 2.0000000000
v 2.0000000000 0.0000000000 2.0000000000
v 2.0000000000 0.0000000000 -2.0000000000
v 0.0000000000 0.0000000000 -2.0000000000
v 0.0000000000 7.0000000000 2.0000000000
v 2.0000000000 7.0000000000 2.0000000000
v 2.0000000000 7.0000000000 -2.0000000000
v 0.0000000000 7.0000000000 -2.0000000000
v -7.0000000000 7.0000000000 2.0000000000
v 5.0000000000 7.0000000000 2.0000000000
v 5.0000000000 7.0000000000 -2.0000000000
v -7.0000000000 7.0000000000 -2.0000000000
v -7.0000000000 10.0000000000 2.0000000000
v 5.0000000000 10.0000000000 2.0000000000
v 5.0000000000 10.0000000000 -2.0000000000
v -7.0000000000 10.0000000000 -2.0000000000
g baseblock
f 0 1 2 3
f 4 7 6 5
```

```
f 0 4 5 1
f 7 4 0 3
f 1 5 6 2
f 2 6 7 3
g topblock
f 8 9 10 11
f 12 15 14 13
f 8 12 13 9
f 15 12 8 11
f 9 13 14 10
f 10 14 15 11
```

Block weights:

```
baseblock: weight = [0.0 -560.0 0.0]
topblock: weight = [0.0 -1440.0 0.0]
```

Block interfaces:

```
interface 0 {
       v 0.0000000000 0.0000000000 2.0000000000  // interface vertices
       v 2.0000000000 0.0000000000 2.0000000000
       v 2.0000000000 0.0000000000 -2.0000000000
       v 0.0000000000 0.0000000000 -2.0000000000
       g baseblock // adjacent block and incident face
       f 0 1 2 3
       g ground
}
interface 1 {
       v 0.0000000000 7.0000000000 2.0000000000  // interface vertices
       v 0.0000000000 7.0000000000 -2.0000000000
       v 2.0000000000 7.0000000000 -2.0000000000
       v 2.0000000000 7.0000000000 2.0000000000
       g baseblock // adjacent block and incident face
       f 4 7 6 5
       g topblock
       f 8 9 10 11
}
```

Values of forces at block interfaces (vector structure as described in §A.1):

f =

```
1.0e+003 *
[1.2200
 0.0000
 0.0000
-0.2200
 0.0000
 0.0000
-0.2200
 0.0000
 0.0000
 1.2200
 0.0000
 0.0000
 1.0800
 0.0000
 0.0000
 1.0800
 0.0000
 0.0000
-0.3600
 0.0000
 0.0000
-0.3600
 0.0000
 0.0000]
```

Appendix C

Data: Optimization Parameters

We provide the rule set for the 10-parameter chapel model in Figure 4-6. Free parameters are indicated in the rules as $*\theta_i*$. Note that several parameters are defined as a functions of free parameters. Following the rules we provide input and output values for $\boldsymbol{\theta}$. No safety factor was used in the optimization.

Starting Shape:

```
hall S(10, 5.5, 13) T(0, 0, 0) R(0, 0, 0)
```

Production Rules:

1. hall --> {roof}: Replace Faces {5} with thickness 3
2. roof --> {roof}: Symmetric split across axis Y
3. roof --> {roof}: Replace with S(1r, 1.1r, 1r) T(0, 0, 0) R(0, 0, 0)
4. roof --> {roofside}: Replace Faces {0, 1} with thickness 0.2
5. roofside --> {roofside}: Replace with S(1r, 1r, 1.1r) T(0, 0, 0) R(0, 0, 0)
6. hall --> {wall}: Replace Faces {0} with thickness $*\theta_0*$
7. hall --> {wall_4}: Replace Faces {1} with thickness $*\theta_1*$
8. wall --> {wall_5 | flattoparch(0.4)}: Subdivide along axis Z with params {0.55, 0.45}
9. flattoparch --> {flattoparch(0.42)}: Symmetric split across axis Z
10. hall --> {column_{100}}: Replace Edges {8, 9, 10, 11} with thickness $*\theta_2*$

11. hall --> {bay}: Repeat along axis X with absolute size 2.5

12. bay --> {bay | groinvault(0.46)}: Subdivide along axis Z with params {0.8, 0.2}

13. groinvault --> {flattoparch(*θ_3*)}: Replace Faces {2, 3} with thickness *θ_4*

14. bay --> {wall_2}: Replace Faces {2, 3} with thickness 0.2

15. wall_2 --> {wall_2}: Symmetric split across axis Y

16. wall_2 --> {wall | column}: Subdivide along axis Y with params {*θ_5*, *$1-\theta_5$*}

17. column --> {column}: Replace with S(3.1r, 1r, 1r) T(1r, 1r, 0) R(0, 0, 180)

18. column --> {column}: Replace with S(*θ_6*r, 1r, 1r) T(1r, 1r, 0) R(0, 0, 180)

19. column --> {column}: Repeat along axis Z with absolute size 3

20. wall --> {wall}: Replace with S(1r, 1r, 0.4r) T(-2r, 0r, 0) R(0, 0, 0)

21. wall_5 --> {hall}: Replace Faces {1} with thickness 2.5

22. hall --> {hall}: Replace with S(1r, 0.9r, 1r) T(0, 0.05r, 0) R(0, 0, 0)

23. hall --> {wall_6}: Replace Faces {2, 3} with thickness 0.2

24. wall_6 --> {wall | wall_6}: Subdivide along axis Y with params {*$(\theta_2-0.15)/2.5$*, *$1-(\theta_2-0.15)/2.5$*}

25. hall --> {wall_5}: Replace Faces {1} with thickness 0.2

26. wall_5 --> {wall_6 | wall_6 | wall_6}: Subdivide along axis Y with params {0.25, 0.5, 0.25}

27. wall_6 --> {wall_5 | wall_5}: Subdivide along axis Z with params {0.5, 0.5}

28. wall_5 --> {wall_5 | flattoparch(*θ_7*)}: Subdivide along axis Z with params {0.5, 0.5}

29. wall_5 --> {wall_5}: Symmetric split across axis Y

30. wall_5 --> {wall_5}: Replace with S(1r, *θ_8*r, 1r) T(1r, 1r, 0) R(0, 0, 180)

31. hall --> {wall_{10}}: Replace Faces {5} with thickness 0.2

32. wall_{10} --> {wall_{10}}: Replace with S(1.1r, 1.1r, 1r) T(0r, -0.05r, 0) R(0, 0, 0)

33. hall --> {column_{100}}: Replace Edges {9, 10} with thickness *θ_9*

34. column_{100} --> {column}: Repeat along axis Z with absolute size 3

Input parameters $\boldsymbol{\theta}^0$ with bounds [lb, ub]:

θ_0: 0.20 [0.20, 1.00]

θ_1: 0.20 [0.20, 1.00]

θ_2: 0.20 [0.20, 2.00]

θ_3: 0.46 [0.10, 0.49]

θ_4: 0.20 [0.20, 1.00]

θ_5: 0.85 [0.20, 0.90]

θ_6: 1.00 [1.00, 4.00]

θ_7: 0.45 [0.10, 0.49]

θ_8: 0.20 [0.10, 1.00]

θ_9: 0.20 [0.20, 2.00]

Output parameters $\boldsymbol{\theta}^*$:

θ_0: 0.3324

θ_1: 0.2276

θ_2: 0.4598

θ_3: 0.4649

θ_4: 0.3454

θ_5: 0.7331

θ_6: 1.0941

θ_7: 0.4660

θ_8: 0.1637

θ_9: 0.4778

Appendix D

Data: Structural Gradient

We provide values for the gradient vectors of the 2-block T model in Figure 5-1. See Appendix B for the geometry in obj format.

$\nabla_u y, \nabla_v y$: In-plane vertex translation (length = # block faces × # verts per face)
$\nabla_n y$: Normal translation (length = # block faces)
$\nabla_\theta y, \nabla_\phi y$: Normal rotation (length = # block faces)

$[\nabla_u y\ \nabla_v y\ \nabla_n y\ \nabla_\theta y\ \nabla_\phi y] =$

```
1.0e+005 *
   -1.3556    0.0095    0.0000    0.0000    0.0000
    0.2882    0.0160    0.0292    0.0000    0.0000
    0.2882   -0.0160    0.0510    0.0000    0.0000
   -1.3556   -0.0095   -0.0528    0.0000    0.0000
   -0.0095    1.9560    0.2570    0.0000    0.0000
    0.0095    1.9560    0.0510    0.0000    0.0000
    0.0160   -0.6948   -0.4695    0.0000    0.0000
   -0.0160   -0.6948   -0.4695    0.0000    0.0000
   -0.0054    0.0062   -0.3521    0.0000    0.0000
    0.0054    0.0062   -1.1655    0.0000    0.0000
    0.0091   -0.0449    0.9308    0.0000    0.0000
   -0.0091   -0.0449   -0.3521    0.0000    0.0000
   -0.0031    0.0018
```

```
 0.0031   0.0018
 0.0031  -0.0018
-0.0031  -0.0018
-0.0128   0.0224
 0.0128   0.0224
 0.0128  -0.0224
-0.0128  -0.0224
-0.0091   0.0449
 0.0091   0.0449
 0.0054  -0.0062
-0.0054  -0.0062
 0.1604  -0.2191
 0.1017   0.0430
 0.1017  -0.0430
 0.1604   0.2191
 0.2191  -0.1604
-0.2191  -0.1604
 0.0430  -0.1017
-0.0430  -0.1017
 0.2921  -0.1604
-0.2921  -0.1604
 0.0573  -0.1017
-0.0573  -0.1017
 0.4811  -0.6415
-0.4811  -0.6415
-0.4811   0.6415
 0.4811   0.6415
-0.4067   0.3050
 0.4067   0.3050
 0.4067  -0.3050
-0.4067  -0.3050
-0.0573   0.1017
 0.0573   0.1017
-0.2921   0.1604
 0.2921   0.1604
```

Bibliography

Allaire, G., Jouve, F., and Toader, A.-M. 2004. Structural optimization using sensitivity analysis and a level set method. *Journal of Computational Physics 194*, 1, 363–393.

Allen, E., and Zalewski, W. 2009. *Form and Forces: Designing Efficient, Expressive Structures*. Wiley.

Attar, R., Aish, R., Stam, J., Brinsmead, D., Tessier, A., Glueck, M., and Khan, A. 2009. Physics-based generative design. In *CAAD Futures 2009 Conference Proceedings: CAAD Futures Foundation*, 231–244.

Attar, R., Aish, R., Stam, J., Brinsmead, D., Tessier, A., Glueck, M., and Khan, A. 2010. Embedded rationality: A unified simulation framework for interactive form-finding. *International Journal of Architectural Computing* (December).

Bathe, K. J. 2006. *Finite Element Procedures*. Prentice Hall, New Jersey.

Bertsekas, D. P. 1995. *Nonlinear Programming*, second ed. Athena Scientific.

Bertsimas, D., and Tsitsiklis, J. N. 1997. *Introduction to Linear Optimization*. Athena Scientific.

Bicanic, N., Stirling, C., and Pearce, C. J. 2003. Discontinuous modelling of masonry bridges. *Computational Mechanics 31*, 1–2, 60–68.

Block, P., and Ochsendorf, J. 2007. Thrust network analysis: a new methodology for three-dimensional equilibrium. *Journal of the International Association for Shell and Spatial Structures 48*, 3, 167–173.

Block, P., Ciblac, T., and Ochsendorf, J. 2006. Real-time limit analysis of vaulted masonry buildings. *Computers & Structures 84*, 29–30, 1841–1852.

Boothby, T. E., and Brown, C. B. 1992. Stability of masonry piers and arches. *Journal of Engineering Mechanics 118*, 2, 367–383.

Cook, R. D. 1995. *Finite Element Modeling for Stress Analysis*. John Wiley & Sons, Inc., ch. 2.5.

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

DeJong, M. J. 2009. *Seismic Assessment Strategies for Masonry Structures*. PhD dissertation, Massachusetts Institute of Technology, Department of Architecture.

Delfour, M. C., and Zolésio, J.-P. 2001. *Shapes and Geometries: Analysis, Differential Calculus, and Optimization*. SIAM.

Drucker, D. C. 1954. Coulomb friction, plasticity, and limit loads. *Journal of Applied Mechanics, Transactions ASME 21*, 1, 71–74.

Gilbert, M., and Melbourne, C. 1994. Rigid-block analysis of masonry structures. *The Structural Engineer 72*, 21, 356–361.

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.

Harvey, W. J. 1991. Stability, strength, elasticity and thrustlines in masonry structures. *The Structural Engineer 69*, 9, 181–184.

Heyman, J. 1966. The stone skeleton. *International Journal of Solids and Structures 2*, 249–279.

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

Huerta, S. 2004. *Arcos bóvedas y cúpulas. Geometría y equilibrio en el cálculo tradicional de estructuras de fábrica*. Instituto Juan de Herrero, Madrid.

Igarashi, T., and Hughes, J. F. 2001. A suggestive interface for 3d drawing. *UIST'01 Proceedings of the 14th Annual Symposium on User Interface Software and Technology*, 173–181.

Laporte, E., and Le Tallec, P. 2003. *Numerical Methods in Sensitivity Analysis and Shape Optimization*. Birkhäuser.

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.

Loo, Y.-C., and Yang, Y. 1991. Cracking and failure analysis of masonry arch bridges. *Journal of Structural Engineering 117*, 6, 1641–1659.

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

Mark, R., Abel, J. F., and O'Neill, K. 1973. Photoelastic and finite-element analysis of a quadripartite vault. *Experimental Mechanics* (August).

Merrell, P., Schkufza, E., and Koltun, V. 2010. Computer-generated residential building layouts. *ACM Transactions on Graphics 29*, 6, 181.

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.

Milani, E., Milani, G., and Tralli, A. 2008. Limit analysis of masonry vaults by means of curved shell finite elements and homogenization. *International Journal of Solids and Structures 45*, 20 (October), 5258–5288.

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

Molins, C., and Roca, P. 1998. Capacity of masonry arches and spatial frames. *Journal of Structural Engineering 124*, 6, 653–663.

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 dissertation, 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.

Shea, K. 2000. EifForm: a generative structural design system. In *ACSA Technology Conference: The Intersection of Design and Technology*.

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.

Stiny, G. 1975. *Pictorial and Formal Aspects of Shape and Shape Grammars*. Birkhauser Verlag, Basel.

Stiny, G. 1980. Introduction to shape and shape grammars. *Environment and Planning B 7*, 343–361.

Talton, J., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. 2011. Metropolis procedural modeling. *ACM Trans. Graphics 30*, 2 (April).

Trefethen, L. N., and Bau, D. 1997. *Numerical Linear Algebra*. SIAM, ch. 19.

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.

Zessin, J., Lau, W., and Ochsendorf, J. 2010. Equilibrium of cracked masonry domes. *ICE Engineering and Computational Mechanics 163*, 3 (September), 135–145.

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