Pattern-Aware Shape Deformation Using Sliding Dockers

Martin Bokeloh* MPI Informatik
Michael Wand* Saarland University and MPI Informatik
Vladlen Koltun† Stanford University
Hans-Peter Seidel* MPI Informatik

Figure 1: *Edit example (left to right): The user places constraints (blue) and manipulates the object by moving constraints. Our deformation model maintains continuous and discrete patterns and adapts the repetition count of discrete patterns, inserting and deleting elements as needed to minimize distortion. Discrete changes are highlighted in orange.*

Abstract

This paper introduces a new structure-aware shape deformation technique. The key idea is to detect continuous and discrete regular patterns and ensure that these patterns are preserved during free-form deformation. We propose a variational deformation model that preserves these structures, and a discrete algorithm that adaptively inserts or removes repeated elements in regular patterns to minimize distortion. As a tool for such structural adaptation, we introduce sliding dockers, which represent repeatable elements that fit together seamlessly for arbitrary repetition counts. We demonstrate the presented approach on a number of complex 3D models from commercial shape libraries.

CR Categories: I.3.5 [Computing Methodologies]: Computer Graphics—Computational Geometry and Object Modeling;

Keywords: shape deformation, shape analysis, symmetry, structural regularity

Links: DL PDF

1 Introduction

Content creation is one of the main bottlenecks in contemporary computer graphics. While sophisticated methods for processing and rendering three-dimensional content are widely available, the creation of detailed custom 3D geometry still requires significant expertise. The issue is not merely of creative ability, but also with the process of directly manipulating detailed 3D models. Such 3D models often feature structural relationships on multiple scales, which need to be manually restored whenever a significant manipulation is performed on the model. Tedious adjustment is often required multiple times in a single modeling session.

Consequently, recent research has begun to investigate *structure-aware* shape editing tools that aim to automate the detailed manipulation required to preserve the structural relationships in a shape as it undergoes manipulation [Kraevoy et al. 2008; Gal et al. 2009; Huang et al. 2009; Wang et al. 2011; Zheng et al. 2011]. Such algorithms analyze the input shape to extract structural features and use the learned structure to assist interactive 3D modeling. They can improve the efficiency of content creation professionals and can assist inexperienced users in adapting existing content to their needs.

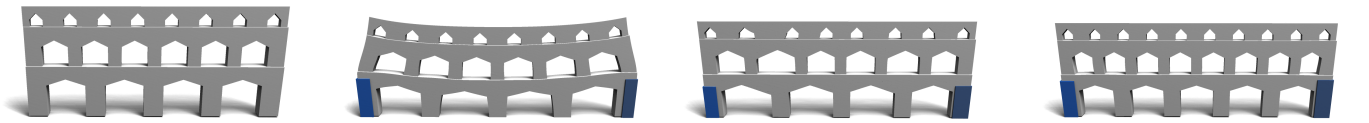
In this paper, we present a structure-aware shape editing technique that detects discrete and continuous patterns in the shape and preserves these patterns under free-form deformation. A key distinguishing feature of our approach is that it can change the structure of the object by adding or removing local elements along regular patterns. This structural adaptation is integrated into a global free-form deformation framework that minimizes the overall stretch of the object.

In our approach, the user specifies a small set of constraints and the system computes a new shape that meets these constraints while preserving structural properties of the original model, as shown in Figure 1. As invariants, we extract 1-parameter groups of partial symmetries. In other words, we detect geometry that is replicated in regular patterns. This includes continuous symmetries such as straight lines as well as repeated discrete elements such as windows in a building. We formulate non-local rigidity constraints to maintain these symmetry properties in the output, and allow for adapting the number of discrete repetitions in order to reduce distortions.

In order to add and remove elements along discrete patterns with minimal distortion, we introduce *sliding dockers*. A sliding docker is an element in a local, repeated structure that interfaces with the rest of the model in such way that the structure can be independently replicated with minimal distortion. We develop an algorithm that automatically finds collections of sliding dockers that repeat in one common translational direction and adapts the replication count in a way that minimizes distortions in the overall object.

We evaluate the presented technique on models taken from commercial 3D model libraries and demonstrate that the presented tech-

*e-mail:{mbokeloh,mwand,hpseidel}@mpi-inf.mpg.de
†e-mail:vladlen@stanford.edu



(a) input object (b) elastic deformation (c) maintaining continuous patterns (d) discrete patterns, adapted repetition counts

Figure 2: *Overview of our approach. Given an input shape (a) and a free-form deformation applied by the user (b), our deformation model preserves continuous patterns in the shape (c) and adapts the repetition counts of discrete patterns to minimize distortion (d).*

nique is able to naturally adapt the discrete structure of regular patterns in the objects in response to free-form manipulation. In summary, this work makes the following key contributions:

- We develop a new deformation model formulated in terms of regular patterns on the shape.
- We introduce the concept of sliding dockers for analyzing partial regularity. This allows for the first time to automatically insert and delete repetitive elements within a free-form shape deformation tool.
- We develop a robust and efficient numerical framework for implementing the technique for real-time shape editing.

2 Related Work

In our approach, the user interacts with 3D shapes in a free-form deformation system, which has a long tradition in computer graphics. Early techniques use smooth basis functions for interpolation [Sederberg and Parry 1986; Coquillart 1990]. Recent work constructs bases specific to a set of control points or a control cage [Ju et al. 2005; Joshi et al. 2007; Lipman et al. 2008; Ben-Chen et al. 2009]. Many of the techniques are based on variational calculus: local regularizers are traded off against the user's constraints. The regularizers aim to maintain local similarity to the input. Elastic deformation models [Terzopoulos et al. 1987], which minimize the non-rigidity of the deformation, are particularly popular [Botsch and Sorkine 2008]. Variants include volume preservation [von Funck et al. 2006], similarity transforms [Liu et al. 2008], and thin-plate splines [Allen et al. 2003]. We use elastic deformation as a "base regularizer" to diffuse stretch and to preserve geometry for which no structural information could be inferred. Our implementation adopts the technique of Sorkine and Alexa [2007] that preserves co-rotated distance vectors in a least-squares sense. We extend this to a volumetric subspace formulation [Zhou et al. 2005; Huang et al. 2006; Sumner et al. 2007; Adams et al. 2008]. This allows interactive handling of large meshes and provides robustness against unfavorable mesh topology so that we can handle "triangle soup."

Local regularizers do not recognize higher-level structural properties in the shape. Consequently, these techniques still expose a large number of degrees of freedom to the user, who has to manually ensure that important structural properties are maintained. This is acceptable for many organic shapes such as creatures, but highly structured objects, such as many man-made objects, are difficult to handle. Kraevoy et al. [2008] use an elastic-type model that adapts to the vulnerability of the local content. Three global stretch axes are fixed, which avoids bending artifacts but also limits the applicability of the technique to axis-aligned stretching. The continuous part of our deformation model could be regarded as an extension of their approach. The major difference is that we determine local stretch directions automatically, based on continuous patterns, rather than fixing them to the global coordinate system.

Xu et al. [2009] introduce slippage analysis for free-form shape deformation, using it to construct a joint-aware deformation model. We also use slippage analysis, but employ it to discover continuous symmetries that are used to maintain the pattern structure of the input. The influential iWires system [Gal et al. 2009] maintains global structural properties of the shape by building constraints that preserve similarity of symmetric parts [Mitra et al. 2006; Podolak et al. 2006; Simari et al. 2006], as well as parallelity and orthogonality of salient feature lines. Huang et al. [2009] apply similar ideas to 2D vector graphics, and Zheng et al. [2011] propagate editing operations based on similarity of components. Using such global knowledge greatly facilitates shape editing, but a key limitation remains: The deformation function is still a continuous, bijective map between input and output. This does not allow the insertion or removal of elements, which can be desirable in response to significant stretch. This issue is the main motivation for our work. A further contribution of our work is that it is based on a single, low-level assumption of preserving partial 1-parameter symmetry groups, rather than a complex set of rules.

A number of approaches have been developed for recombining shapes out of parts. Approaches that utilize manual part composition have been described [Funkhouser et al. 2004; Pauly et al. 2005; Kraevoy et al. 2007], as well as automatic methods for detecting structural regularity [Pauly et al. 2008]. Recently, techniques have appeared that compute rules for discrete changes automatically [Mitra and Pauly 2008; Bokeloh et al. 2010]. However, none of the automatic techniques provides free-form deformation editing. In addition, structural relationships treated in previous work are often limited: The technique of [Bokeloh et al. 2010] can detect regular patterns in 3D geometry and create shape variations by inserting or removing pieces, but the detected dockers must partition the model globally into two disjoint pieces by a symmetric cut. The technique thus cannot handle 3D models with local patterns that do not globally partition the shape. We extend the docking approach of Bokeloh et al. [2010] and introduce *sliding dockers*, which require only a partial partitioning and are designed to operate as part of a structure-aware free-form deformation framework. The recent method of Wang et al. [2011] infers a scene graph structure for an unannotated 3D mesh to allow for both continuous and discrete parameter variations. However, the method does not provide a constraint-based free-form deformation interface.

Wu et al. [2010] describe an image resizing method that summarizes symmetry structure in the image and uses it to add or remove columns or rows of repeated elements in response to resizing operations. Their method tackles challenges that arise in image processing, such as perspective distortion or illumination changes. Our work focuses on manipulation of three-dimensional geometry, which imposes challenges such as producing seamless surfaces after adding or removing geometric elements, and dealing with general deformation and multiple independent resizing directions.

3 Overview

Our technique is designed to preserve regular patterns in the input shape. We detect such patterns in a preprocessing step, described in

Section 4. We then apply a continuous deformation model that tries to maintain the detected structure, as described in Section 5. In order to reduce distortions, we automatically insert or delete repeated elements using sliding dockers, developed in Section 6. In this section we give a brief overview of each component of our approach. These components are illustrated in Figure 2.

Input: Our technique accepts a general triangle mesh $\mathcal{S} \subseteq \mathbb{R}^3$ as input (Figure 2a), with no restrictions on geometry or topology (in other words, "triangle soup"). In addition, the user can select an arbitrary number of handles $\mathcal{H}_i \subseteq \mathcal{S}$ and can move and rotate them to new positions (this is an interactive process, with real-time feedback by the system). In the following, we will use $l(\mathcal{S})$ to denote the maximum side length of an axis aligned bounding box of $\mathcal{S}$; this value is used to scale relative parameters automatically.

Deformation model: The basis of our technique is a standard elastic shape deformation model [Terzopoulos et al. 1987; Sorkine and Alexa 2007]. It computes a deformation field f that minimizes the deviation from the user's constraints and tries to keep the object as rigid as possible. In other words, the model diffuses stretch (stress tensors) as uniformly as possible across the object surface under the given constraints (Figure 2b). We use this behavior as a "base regularizer" with low weight, aiming at just dissipating the stretch induced by the constraints.

Shape analysis: We perform a shape analysis step in preprocessing, in order to identify regular patterns in the input geometry. We model regular patterns as one-parameter partial group structures in the symmetry structure of the object: We find parts $\mathcal{P} \subseteq \mathcal{S}$ that show up multiple times, replicated by a series of transformations $\mathbf{T}^x$, where x ranges over a continuous or integral range $\mathcal{I} \subset \mathbb{R}$, leading to continuous and discrete patterns.

Sliding dockers: For discrete patterns, we find *sliding dockers*, which are building blocks that can be replicated when the object is locally stretched. Sliding dockers are cut out of the input surface in a way that the boundaries fit seamlessly when multiple pieces are attached to each other regularly. In addition, the boundaries of this repeated region are slippable, such that any repetition count yields closed geometry. In our current approach, discrete changes are limited to pattern with one degree of freedom only.

Continuous, structure-aware deformation: Using the structural knowledge gained in preprocessing, we add constraints to our deformation model that aim at preserving the patterns in the shape. These constraints are given a higher weight than the elastic regularizer, thus dominating the deformation results (Figure 2c).

Discrete relaxation: We measure the stretch in the continuously deformed model and automatically insert or delete sliding dockers to relax the stretch in the model. Such automatic structure adaptation allows a broader range of deformations to be applied without violating the natural appearance of the object (Figure 2d).

4 Pattern-Based Structure Model

Our approach begins with a preprocessing phase, which analyses the input geometry to detect structural regularity in the form of partial regular patterns. These patterns will be kept invariant in the later editing process. Regular patterns are defined with respect to a *group of admissible transformations*:

Transformations: Let $\mathcal{G}$ be a group of bijective, continuous mappings $\mathbb{R}^3 \to \mathbb{R}^3$. Throughout this paper, we will restrict our consideration to translations.

Replications: For a transformation $\mathbf{T} \in \mathcal{G}$, let $\mathbf{T}^x$ denote the x-fold application of $\mathbf{T}$, where $x \in \mathbb{R}$ is a continuous value. As we are dealing with translations, which form a linear space, this corresponds to a multiplication by x. However, we will stick to the more general group notation because this shows more clearly the conceptual structure. It also indicates how our framework could be generalized to more complex groups of transformations (for example, including rotation). For a set $A \subseteq \mathbb{R}$, we will use the the notation $\mathbf{T}^A := \{\mathbf{T}^x | x \in A\}$ in order to denote the set of powers of $\mathbf{T}$. Furthermore, for $\mathcal{P} \subset \mathbb{R}^3$, we write $\mathbf{T}^A(\mathcal{P}) := \cup_{x \in A} \mathbf{T}^x(\mathcal{P})$ to denote replications of $\mathcal{P}$. In particular, $\mathbf{T}^{\mathbb{R}}(\mathcal{P})$ denotes the *extrusion surface* that replicates $\mathcal{P}$ continuously.

4.1 Partial Regular Patterns

Our goal is to preserve the symmetry structure of the input $\mathcal{S}$ under deformations while admitting insertions and deletions of parts. The first step is to look at the global symmetries of $\mathcal{S}$:

Symmetry groups: The set of all operations $\mathbf{T} \in \mathcal{G}$ that maps $\mathcal{S}$ to itself, i.e., $\mathbf{T}(\mathcal{S}) = \mathcal{S}$ forms a subgroup of $\mathcal{G}$. For commutative groups, such as the translations we are considering, symmetry groups are isomorphic to infinite (potentially continuous) regular lattices [Pauly et al. 2008]. To deal with finite models and partial regularity, we include symmetric structures that are only excerpts of a larger grid. Due to commutativity, we can factor more general patterns into overlapping 1-parameter grids. Overall, this leads to the following model of a regular *regular pattern*:

Partial 1-parameter symmetry groups ("patterns"): Consider $\mathcal{P} \subseteq \mathcal{S}$ and a *generator transformation* $\mathbf{T} \in \mathcal{G}$, and let $\mathcal{I} \subset \mathbb{R}$ be a real interval. If we have $\mathbf{T}^{\mathcal{I}}(\mathcal{P}) \subseteq \mathcal{S}$, we have found a *continuous partial 1-parameter symmetry group* of $\mathcal{S}$. If $\mathcal{I} \subseteq \mathbb{Z}$ is an integer interval with at least three elements, we have found a *discrete* partial 1-parameter symmetry group. For brevity, we will call these structures continuous and discrete *(regular) patterns*, respectively.

4.2 Computational Framework

Discrete Patterns: We compute the discrete patterns by a symmetry analysis similar to Bokeloh et al. [2010] (see their paper for details): We detect sharp creases in the input mesh and combine pairs of adjacent, non-collinear creases to form "bases". Two base pairs are potentially corresponding if they have matching length and enclose the same angle. Very small feature lines (below 2.5% $l(\mathcal{S})$) are removed for efficiency reasons. We now use a RANSAC procedure to compute regular patterns: Random pairs of potentially corresponding bases are chosen and the relative transformation $\mathbf{T}$ is computed. We search for all potentially corresponding bases that are located at positions $\mathbf{T}^x$ for some $x \in \mathbb{R}$. This gives us initial pattern candidates. The next step is to extract generator transformations $\mathbf{T}$ that generate the 1-parameter groups $\mathbf{T}^i, i \in \mathcal{I} \subset \mathbb{Z}$: We look at all pairwise transformations between candidate bases. For each pair, we compute the number of bases that lie on the grid $\mathbf{T}^{\mathbb{Z}}$ induced by the two bases. We output the choice of generator that yields the largest integer interval $\mathcal{I}$ of matching bases and exclude these from further processing. We iterate until no more valid patterns are found.

Continuous patterns: Continuous symmetries are detected by slippage analysis, following the algorithm of [Gelfand and Guibas 2004], which can be trivially restricted to translational motions. Parts $\mathcal{P}$ that have the same continuous symmetry properties are extracted by simple region growing (see [Gelfand and Guibas 2004] for implementation details).

Normalization: To remove overlapping, partial patterns, we always choose maximal sets $\mathcal{P}$ for the geometry involved and the smallest possible generating transformation (shortest translation

vector). In the discrete case, this is straightforward. In the continuous case, patterns are computed with a segmentation using slippage analysis [Gelfand and Guibas 2004]. Thus, the maximal surface parts $\mathcal{P}$ are regions with the same slippage properties. Using this representation, the interval $\mathcal{I}$ in the definition above vanishes and will be omitted for continuous symmetries in the following.

Regular Patterns of $\mathcal{S}$: Using these conventions, we obtain a finite set of discrete regular patterns $R_D = \{(\mathcal{P}_1, \mathbf{T}_1, \mathcal{I}_1), ..., (\mathcal{P}_N, \mathbf{T}_N, \mathcal{I}_N)\}$ and another finite set of continuous regular patterns $R_C = \{(\mathcal{P}_1, \mathbf{T}_1), ..., (\mathcal{P}_M, \mathbf{T}_M)\}$.

5 Deformation Model

In this section, we describe the global continuous deformation model that serves as the basis for our deformation framework. First, we describe the representation of the deformation function (Section 5.1). Second, we review the standard elastic deformation model, which serves as our base regularizer (Section 5.2). Third, we introduce additional structure-aware constraints in order to preserve regular patterns (Section 5.3).

5.1 Representation

In order to compute a deformation, we embed the surface $\mathcal{S}$ into a volume $\mathcal{V} \subset \mathbb{R}^3$, $\mathcal{S} \subset \mathcal{V}$, and deform this volume using a deformation field $f : \mathcal{V} \to \mathbb{R}^3$. This approach has the benefit of making the deformation independent of the representation of $\mathcal{S}$ so that arbitrary types of input geometry and general surface topology can be handled easily. Following [Huang et al. 2006; Sumner et al. 2007], we use a subspace method to discretize f, i.e., we use a low-dimensional basis for representing the deformation: We create a number of nodes $\mathbf{x}_1, ..., \mathbf{x}_k \subset \mathbb{R}^3$ and center radial basis functions b_i around them to define the deformation field:

$$f(\mathbf{x}) = \sum_{i=1}^{K} \mathbf{u}_i b_i(\mathbf{x}) \tag{1}$$

Here, $\mathbf{u}_i \in \mathbb{R}^3$ are the deformed target positions of the nodes $\mathbf{x}_i$. As basis functions, we employ moving-least-squares (MLS) meshless basis functions of linear precision, based on a finite support Wendland kernel, as proposed in [Adams et al. 2008]. These functions are able to represent smooth deformation fields with a small number of nodes. We place the nodes by discretizing $\mathcal{V}$ to a regular grid of user specified spacing h. We set the support of the basis function to $2h$ to make sure that at least two basis functions overlap each surface point in x-, y-, and z-direction. The volume $\mathcal{V}$ itself is created by offsetting $\mathcal{S}$ by h in all directions (i.e., a Minkowski sum of a sphere of radius h and $\mathcal{S}$). This guarantees that the basis functions and their derivatives are well defined on $\mathcal{S}$.

Remark: In the following, we use two basic numerical discretization constants. The first, h, determines the resolution of the deformation field, which is typically in the range of $5\%\ l(\mathcal{S})$. In addition, we also use smaller constant ϵ (in the range of $1\%\ l(\mathcal{S})$) for discretizing other functions, such as symmetry information and to form neighborhoods for slippage analysis.

f is determined by a variational approach: We set up an energy function $E(f)$ that is minimized by an optimal f:

$$E = E_u + \lambda_r E_r + \lambda_c E_c + \lambda_d E_d \tag{2}$$

E_u describes user constraints and E_r is the base-regularizer that creates elastic behavior. These two terms correspond to a standard elastic shape deformation approach. We then add two additional terms to preserve the pattern structure: E_c preserves continuous patterns such as straight lines, and E_d preserves discrete patterns. The λ_* control the influence of the different regularizers relative to the user constraints. We typically use $\lambda_r = 0.01$ and $\lambda_c = \lambda_d = 1$.

5.2 Elastic Deformation

The first energy term E_u accounts for user constraints. We use the standard "handle" model where parts $\mathcal{H}_i \subseteq \mathcal{S}$ of the input surface can be translated and rotated in space:

$$E_u(f) = \sum_{\mathcal{H}_i \in \mathcal{H}} \int_{\mathcal{H}_i} \left(f(\mathbf{x}) - (\mathbf{R}_i^{(\mathcal{H})} \mathbf{x} + \mathbf{t}_i^{(\mathcal{H})}) \right)^2 d\mathbf{x} \tag{3}$$

The second term E_r is the elastic deformation energy. We employ a standard formulation based on a Poisson system [Sorkine et al. 2004] with co-rotated local frames [Müller et al. 2002; Sorkine and Alexa 2007], adapted to the volumetric settings [Zhou et al. 2005]: We connect all pairs of nodes with overlapping shape functions and preserve their distance vectors:

$$E_r(f) = \sum_{i=1}^{K} \sum_{j \in N(i)} \omega_{i,j} \left(\mathbf{u}_i - \mathbf{u}_j - \frac{1}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{x}_i - \mathbf{x}_j) \right)^2 \tag{4}$$

Here, $N(i)$ denotes accordingly the set of indices of nodes adjacent to node $\mathbf{x}_i$. The $\omega_{i,j}$ are the weights of their coupling, which we set according to the Wendland kernel of the MLS basis (see [Zhou et al. 2005] for a more sophisticated scheme). The variables $\mathbf{R}_i$ are rotation matrices at each node $\mathbf{x}_i$ that are optimized along with the node displacements.

Numerical solution: In order to solve for a minimum of the energy, we determine the derivative with respect to the $\mathbf{u}_i$, which is a linear, Laplacian-type system, and set it to zero. Next, we update the rotation $\mathbf{R}_i$ by estimation from their neighbors. This procedure is iterated until convergence. Details can be found in [Sorkine and Alexa 2007]. Because of the special structure of this system, only the right-hand side changes during the iterations. Therefore, it is possible to prefactor the matrix so that the inversion can be solved by sparse matrix-vector products, leading to a substantial speed-up. As suggested in [Sorkine and Alexa 2007], we employ the TAUCS library for sparse Cholesky factorization [Toledo 2003].

5.3 Structure Aware Deformation

We now augment our deformation model so that it better preserves the structure of the deformed geometry. We first employ a general anisotropic deformation model in order to favor a local preservation of pattern structures. Secondly, we add global constraints that preserve continuous and discrete patterns.

5.3.1 Local Constraints

Locally, we would like geometry to deform in a way that preserves continuous symmetries. If we look at this from the local perspective of the elastic regularizer, this means that we would like the deformation to happen along slippable motions rather than orthogonal to them, because this will only change the parametrization, but not the geometric shape. Accordingly, we augment Equation 4 by using an anisotropic error quadric in order to weight deformations. We replace the isotropic error term $(\mathbf{u}_i - \mathbf{u}_j - \frac{1}{2}(\mathbf{R}_i + \mathbf{R}_j)(\mathbf{x}_i - \mathbf{x}_j))^2 =: (\mathbf{d}_{i,j}^{el})^2$ by:

$$\left(\mathbf{d}_{i,j}^{el}\right)^T \frac{1}{2} \left(\mathbf{Q}_s(\mathbf{x}_i) + \mathbf{Q}_s(\mathbf{x}_j)\right) \left(\mathbf{d}_{i,j}^{el}\right) \tag{5}$$

where $\mathbf{Q}_s(\mathbf{x})$ is computed by a translational slippage analysis:

$$\mathbf{Q}_s(\mathbf{x}) = \int_{N_h(\mathbf{x})} \mathbf{n}(\mathbf{y}) \cdot \mathbf{n}(\mathbf{y})^T d\mathbf{y} + 0.01 \cdot \mathbf{I}. \tag{6}$$

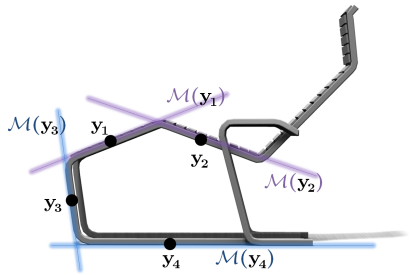


Figure 3: *Constraint manifolds are constructed to preserve discrete (purple) and continuous (blue) pattern structures.*

Here, $\mathbf{n}(\mathbf{x})$ is a unit surface normal at $\mathbf{x} \in \mathcal{S}$, and $N_h(\mathbf{x})$ is the Euclidean h-neighborhood of $\mathbf{x}$ in $\mathcal{S}$. Intuitively, this can be explained as an average of planar constraints: At each point, the outer products create quadrics that penalize deviations in normal direction only; tangential motions have zero cost. For complex geometry, the costs in different directions are averaged. Thus, a straight line will penalize anything but motions in its tangential direction and irregular geometry will resist any deformation. As we still need a base regularizer that diffuses stretch, even along perfectly straight lines, we add 1% of the identity matrix.

5.3.2 Global Constraints

The effect of the local model weakens with distance: Extended objects such as straight lines or flat planes can still show significant global bending. Increasing the weights could in principle solve this problem but would lead to an impractically ill-conditioned numerical system. Therefore, we introduce explicit global constraints to maintain general patterns globally.

We address the continuous patterns first. The discrete case is discussed afterwards and requires only a few minor modifications. Let $\mathcal{P}$ be a part of constant slippage, which can have one or two translational degrees of freedom $\mathbf{T}_1, \mathbf{T}_2$. For each point $\mathbf{y} \in \mathcal{P}$ we then consider the line or plane

$$\mathcal{M}(\mathbf{y}) = \mathbf{T}_1^{\mathbb{R}} \mathbf{T}_2^{\mathbb{R}}(\mathbf{y}), \tag{7}$$

which is the affine *constraint subspace* $\mathcal{M}(\mathbf{y})$ for point $\mathbf{y}$ (see Figure 3). Let $\mathbf{t}_1, \mathbf{t}_2$ be tangent vectors of this space. We form the quadric

$$\mathbf{Q}_{\mathcal{M}}(\mathbf{y}) = \mathbf{I} - \mathbf{t}_1(\mathbf{y})\mathbf{t}_1(\mathbf{y})^T + \mathbf{t}_2(\mathbf{y})\mathbf{t}_2(\mathbf{y})^T, \tag{8}$$

which penalizes displacements that would take point $\mathbf{y}$ out of the constraint subspace. For the 1-slippable case, the same construction is made with a single tangent vector.

During editing, only the orientation of the constraint spaces $\mathcal{M}(\mathbf{y})$ is fixed and translations of the complete pattern as a whole not penalized. This is obtained by expressing the constraints in terms of difference vectors, as described below.

Numerical implementation: Given k slippable parts $\mathcal{P}_1, ..., \mathcal{P}_k$ and corresponding sets of motions, we now build a global constraint energy that preserves continuous symmetries but nevertheless permits moving the patterns freely in space.

We identify the region $\mathcal{P}_i$ of each slippable part and sample them uniformly with points $\mathbf{q}_j^{(i)}$, $j = 1, \ldots, n_i$ of spacing h using Poisson disc sampling. We then connect the points with their centroid $\mathbf{c}^{(i)}$ and form distance vectors between the centroid and all other sample points, which yields a star geometry.

The original, constant distance vectors are $\mathbf{d}_j^{(i)} = \mathbf{q}_j^{(i)} - \mathbf{c}^{(i)}$. The distance vectors in the deformed model are given by $f(\mathbf{d}_j^{(i)}) := f(\mathbf{q}_j^{(i)}) - f(\mathbf{c}^{(i)})$. We then minimize the differences in a least squares sense:

$$E_c = \sum_{i=1}^{k} \sum_{j=1}^{n_i} \left[f(\mathbf{d}_j^{(i)}) - \mathbf{d}_j^{(i)} \right]^T \mathbf{Q}_{\mathcal{M}}(\mathbf{q}_j^{(i)}) \left[f(\mathbf{d}_j^{(i)}) - \mathbf{d}_j^{(i)} \right] \tag{9}$$

Weighting by the error quadric $\mathbf{Q}_{\mathcal{M}}(\mathbf{q}_j^{(i)})$ constrains the deviation to the tangent space of the constraint manifolds. Again, only constants in Equation 9 change, so that only the right-hand side of the linear system needs to be updated. This permits prefactorization, which is crucial for achieving real-time performance.

Discrete patterns: In the discrete case, we use almost exactly the same constraints. We obtain 1-dimensional constraint manifolds as $\mathcal{M}(\mathbf{y}) = \mathbf{T}^{\mathbb{R}}(\mathbf{y})$, where $\mathbf{T}$ is the transformation that links two elements in the discrete pattern. For continuous patterns, moving surface points along their constraint manifold usually does not change the geometry substantially. In the discrete case, however, tangential drift is clearly noticeable because we have complex, non-slippable geometry being replicated. We therefore modify the constraints to enforce a constant step size: We use quadrics $\mathbf{Q}_{\mathcal{M}}(\mathbf{y}) = \mathbf{I}$ of full rank and use an equidistant stepping $\mathbf{T}^{\mathbb{Z}}(\mathcal{P})$ to constrain difference vectors between corresponding parts.

6 Sliding Dockers

In this section, we describe how our framework adapts the repetition count of discrete patterns in order to reduce stretch. In Sections 6.1 and 6.2, we examine the discrete patterns more closely and try to decompose their geometry into *sliding dockers* that allow changing the repetition count seamlessly. In Section 6.3, we describe how sliding dockers are integrated into the deformation framework.

The interactive deformation proceeds in two steps. First, we let the user deform the object. In areas covered by discrete patterns, the anisotropic deformation weights (Equation 6) are changed such that deformation along motion field $\partial_x \mathbf{T}^x$ incurs minimal penalties. In this step, the pattern area acts as a *placeholder*, allocating space for sliding dockers along the pattern's motion direction. In the second step, we compute the stretch within the placeholder, round it, and insert an adapted number of instances. Then the deformation is recomputed for the new composition of the object. The two deformation steps are always performed in sequence and only the adapted shape is presented to the user.

6.1 Defining Sliding Dockers

Our first task is to identify pieces of geometry that can be replicated. Let $\mathbf{T}^{\mathcal{I}}(\mathcal{P}) \subset \mathcal{S}$ be a discrete pattern, as computed in Subsection 4.2. We now need to determine whether it contains elements that can be replicated while continuously interfacing with each other and with existing geometry.

As shown in Figure 4, such elements have to meet two types of boundary conditions. First, boundaries orthogonal to the motion field have to match each other; we therefore require *symmetry* of this geometry with respect to $\mathbf{T}$. Second, in direction tangential to the motion, we require *slippability* with respect to $\mathbf{T}$; by changing the repetition count, the boundaries of the pattern and the rest of the geometry will slide with respect to each other, and slippability will ensure that we always have matching geometry.

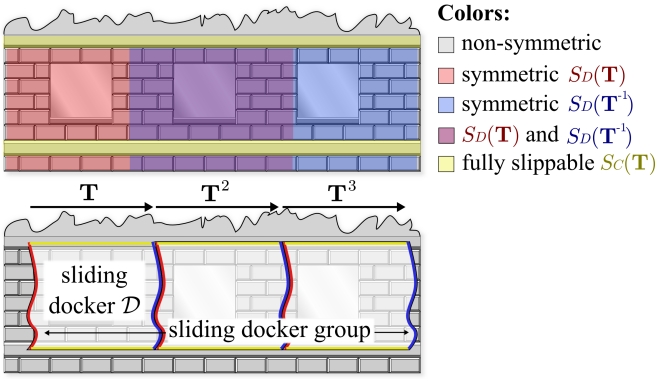


Figure 4: *Boundary conditions for sliding dockers.*

We perform symmetry analysis to find all geometry within $\mathcal{S}$ that is symmetric with respect to $\mathbf{T}$. We denote this geometry by $S_D(\mathbf{T})$:

$$S_D(\mathbf{T}) := \{\mathbf{x} \in \mathcal{S} | \mathbf{T}(\mathbf{x}) \in \mathcal{S}\} \tag{10}$$

It is easy to see that the image of $S_D(\mathbf{T})$ under $\mathbf{T}$ is $S_D(\mathbf{T}^{-1})$; in other words, this is the area the symmetry transform maps to. By slippage analysis, we obtain the subset of $\mathcal{S}$ that is slippable with respect to $\mathbf{T}$. We denote this set by $S_C(\mathbf{T})$.

Consider a piece of geometry $\mathcal{D} \subset \mathcal{S}$. We say that $\mathcal{D}$ is a *sliding docker* with respect to $\mathbf{T}$ if the following two conditions hold. First, the boundary $\partial\mathcal{D}$ must be located entirely in either $S_D(\mathbf{T})$, $S_D(\mathbf{T}^{-1})$, or $S_C(\mathbf{T})$. Second, for every point $\mathbf{x} \in \partial\mathcal{D}$ in $S_D(\mathbf{T})$, the corresponding point $\mathbf{T}(\mathbf{x}) \in S_D(\mathbf{T}^{-1})$ must be included in the boundary $\partial\mathcal{D}$, and vice versa. In other words, we cut out a sliding docker by cutting through symmetric geometry and slippable area along the motion of the pattern; when cutting through the symmetric area, we need to use matching cut lines within $S_D(\mathbf{T})$ and $S_D(\mathbf{T}^{-1})$ so that the pieces fit together seamlessly later (see Figure 4).

We can easily extend this definition to a whole array of sliding dockers. In order to find n matching dockers simultaneously, we require that the two boundary conditions are met by n replicated pieces along the motion direction, namely $\{\mathcal{D}, \mathbf{T}(\mathcal{D}), \mathbf{T}^2(\mathcal{D}), ..., \mathbf{T}^{n-1}(\mathcal{D})\}$. We call such an ensemble a *sliding docker group*.

6.2 Finding Sliding Dockers

In order to find sliding dockers, we first need to compute the symmetry information. We use the same computational framework as Bokeloh et al. [2010]: Transformation candidates are estimated by matching feature lines, and we obtain the slippable motions from slippage analysis (Section 4.2).

Motion space transform: In order to simplify further computations, we perform a transformation into *motion space*. In this space, one axis corresponds to the (translational) motion $\mathbf{T}^x$, while the other two axes y, z are two remaining Euclidean coordinate axes. For translations $\mathbf{T}$, this is just a linear transform. As a notational convention, we will denote the motion coordinate as the x-axis of the motion space.

Next, we build a voxel grid in motion space to represent the symmetry information (see Figure 5 for an illustration). The side length in the motion dimension and Euclidean dimensions is chosen such that the spacing in world coordinates is not larger than the discretization

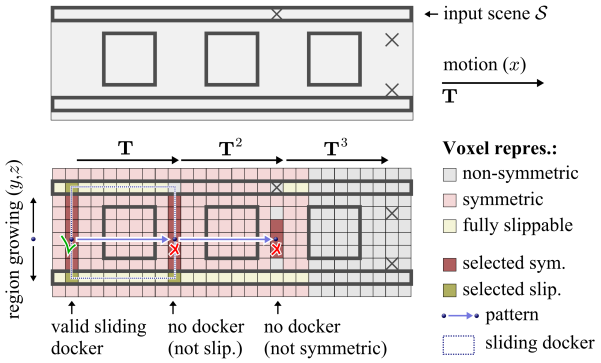


Figure 5: *Sliding dockers are extracted by region growing in motion space. Starting from a pattern base, we grow orthogonally to the motion direction, using only symmetric voxels. We proceed until either hitting only fully slippable voxels (success) or a non-symmetric voxel (failure).*

constant ϵ. Furthermore, we denote by $l_{\mathbf{T}}$ the (integer) number of voxels that represent one application of the motion $\mathbf{T}$.

For each discrete regular pattern $(\mathcal{P}, \mathbf{T}, \mathcal{I}) \in R_D$, we transform the scene into the motion space of $\mathbf{T}$ (this is sped up by collecting all patterns that have the same motion space). We now retrieve the geometry in every non-empty voxel (i, j, k) and match the content against voxels $(i + l_{\mathbf{T}}, j, k)$, corresponding to the transformed geometry. Matching voxels are tagged as symmetric. Next, we compute the slippability of each voxel and check for each $\mathbf{T}$-slippable voxel (i, j, k) whether all voxels $(i, j, k), (i + 1, j, k), ..., (i + l_{\mathbf{T}}, j, k)$ are $\mathbf{T}$-slippable as well. If so, we mark the voxel as *fully slippable*. This means that the geometry at this voxel v, as well as all geometry along the motion $\mathbf{T}^{[0,1]}(v)$ is slippable, which is what we need to cut out a sliding docker. We perform this analysis for all non-empty voxels, as well as for empty voxels that are direct neighbors of occupied ones. Empty voxels must map to other empty voxels in order to be fully slippable.

Extracting sliding dockers: After this precomputation, finding a sliding docker is simple. We start at a symmetric voxel and grow in the (y, z)-plane of the motion space until we either hit a non-symmetric voxel, or a fully slippable voxel. If we hit a single non-symmetric voxel, we dismiss the whole attempt. If we only end at fully slippable voxels (including the empty fully slippable ones), we have found a sliding docker: We can just cut out an extrusion of the visited region in x-direction in motion space. By transforming back into world coordinates, we obtain the final sliding docker. When performing this computation, we always try to find a maximal sliding docker group by checking for symmetry and full slippability with respect to $\mathbf{T}, \mathbf{T}^2, ...$ simultaneously (this corresponds to testing voxels separated by multiples of $l_{\mathbf{T}}$ in the x-direction).

The whole computation is attempted for each base of a detected pattern. This yields a large number of sliding dockers, most of which overlap. In order to remove overlapping pieces, we use a simple greedy algorithm: We take the largest sliding docker group (i.e., the one with the highest repetition count) and delete all overlapping sliding docker groups. This is iterated until no more sliding docker groups are found.

6.3 Using Sliding Dockers

We can now integrate the sliding dockers into our deformation framework. First, we have to set up the first of the two deformation steps. We mark all areas that are covered by a sliding docker group.

At each such point, we deactivate all regularizers except from the elastic deformation energy. Let $\mathbf{t}(\mathbf{y})$ be a normalized vector parallel to the constant tangent $\partial_x \mathbf{T}^x(\mathbf{y})$ of the motion field. We then set the error quadric of the elastic deformation model (Equations 5, 6) to $\mathbf{I} - \mathbf{t}\mathbf{t}^T + 0.01 \cdot \mathbf{I}$. This makes the geometry easily stretchable in the pattern direction. For the rest of the model, we use all energy terms as previously described, including global and local pattern preserving constraints.

We then solve the resulting system. In the result, we measure the stretch of the pattern region by integrating along lines of the motion direction: We connect corresponding points in neighboring instances of the pattern elements and compute the average length. Dividing the value for the deformed and undeformed state gives us a *stretch factor* F. We multiply this factor by the number of original repetitions and round it to the nearest integer to determine the number of elements to insert.

For inserting elements, we again use the motion space. We scale the elements by the inverse stretch factor in the x-direction of the motion space, concatenate the pieces, and backtransform into world coordinates. We then replace the original pattern with the adapted one.

Next, we need to make sure that the elastic deformation model undoes the stretch: If we add more elements, this means that we squeeze smaller replicas into the original space. The energy of Equation 4 would then try to preserve this configuration in an as-rigid-as-possible manner. Therefore, we augment the distance vectors: Instead of the distances of the squeezed elements, we employ the original distance vectors. For basis functions that overlap regions that are stretched by different factors, we compute a weighted average according to the respective kernel function of that node.

As error quadrics, we use full rank identity matrices, aiming at preserving the original shape of the inserted pattern elements. A small detail helps at this point to improve the quality of the results: At the boundary between prestreched and unstreched geometry, the elastic deformation model tends to produce artifacts. Therefore, we set different error quadrics for pairs of nodes that connect across normal geometry and sliding docker areas. We use a quadric $\mathbf{I} - \mathbf{t}\mathbf{t}^T$ in order to make the boundary slidable, not diffusing the errors introduced by the stress discontinuity.

Assembling a new shape by inserting stretched patterns creates shapes that are only C^0-continuous at the boundaries. The elastic deformation model will aim at undoing the deformation, but the subspace model cannot represent high frequency details, which implies that visible artifacts at the boundaries can remain. In order to avoid this problem, we need to make sure that the new base shape that we create is actually smooth and the deformation is low frequency. Therefore, we use a windowing function $g(x)$ in the direction of the motion. We transition from the constant stretch factor of 1 to a different stretch s using a smooth step function. We employ a cosine step function $(1 - 0.5s \cos x)$ to transition from stretch 1 to a new constant stretch of s, and a similar cosine step leads back to 1. This function can be integrated analytically (to obtain the positions, rather than their derivatives) and inverted so that we can compute the inner stretch s that makes all instances fit into the placeholder. We fix the support of the smooth steps to always cover a support of at least $2h$ each, thereby creating a low-frequency distortion that remains within the Nyquist limit of the deformation model.

7 Implementation and Results

We have implemented the described shape editing system in C++ and evaluated it on a commodity workstation with an Intel Core-2 Quad CPU with 2.6GHz cores and 8GB of RAM. Our implementation is single-threaded. As benchmark data, we have collected a number of models from commercial 3D model libraries. We use models from the Digimation Archive (www.digimation.com) and from Dosch Design (www.doschdesign.com). We also include examples from [Kraevoy et al. 2008] and [Bokeloh et al. 2010]. For models with large triangles, we perform one or more 1:4 subdivision steps to obtain a sufficiently densely sampled mesh such that even elastic deformation with bending can be accommodated. The resulting models are output as collections of triangles that generally do not form watertight meshes, and are thus in general "polygon soups."

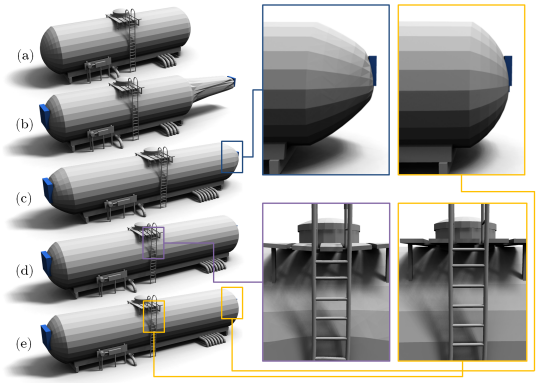


Figure 6: *The effect of individual energy terms in the variational deformation framework. Leaving out any single energy term leads to artifacts. (a) input model, (b) no elasticity, (c) no local constraints, (d) no global constraints, (e) full energy.*

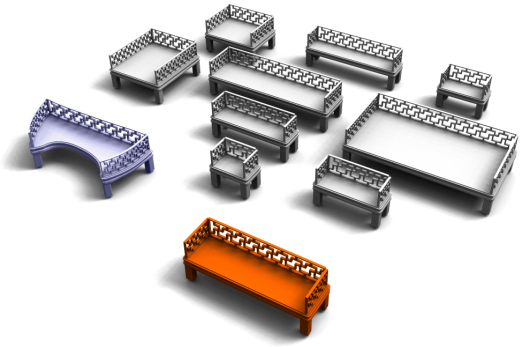
Figures 2 and 7 show a number of example models that have been edited using our approach. Please refer to the accompanying video for a demonstration of interactive editing. The deformation results produced by the technique are quite plausible; for many of the examples, it would be challenging to identify the original model without the highlighting. Some minor artifacts can be seen due to small-scale irregularities in the input geometry, which cause some patterns to only be detected in chunks, leading to a small amount of residual deformation.

The blue examples use only the elastic energy term (Equation 2), with discrete relaxation still enabled. Despite strong bending, our approach reliably adapts the repetition count of the patterns without visible seams, discontinuities, gaps, or similar artifacts. In some models, the triangulation becomes visible; this could be resolved by a better adaptive mesh subdivision scheme.

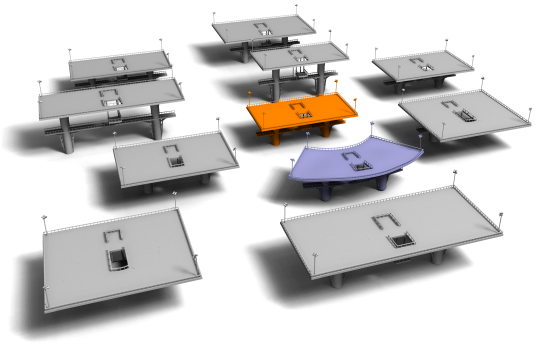
Parameters: Our algorithm is not very sensitive to parameter settings, and we mostly use default parameters everywhere. Only one parameter has a strong effect on the quality of the results: The error threshold for matching approximately symmetric line features. For complex models, we also increase the resolution of the subspace deformation model (the Westminster Palace model uses $2.5\%\ l(\mathcal{S})$ instead of $5\%\ l(\mathcal{S})$, which leads to a reduced interactive frame-rate) and lower the minimum size of relevant features.

Timings: As shown in the video, editing can be done interactively for all presented models. The structure analysis in preprocessing takes up to a few minutes for each model, and prefactorization of the linear systems adds ten more seconds.

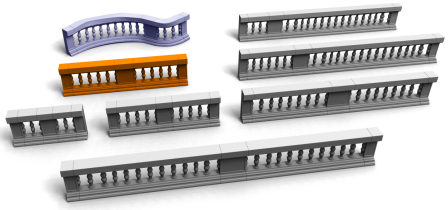
Effect of individual energy terms: In Figure 6, we show that all ingredients of the variational framework are necessary to obtain good results. Deactivating the elastic energy means that stretch is not diffused. The local constraints are helpful as they provide a



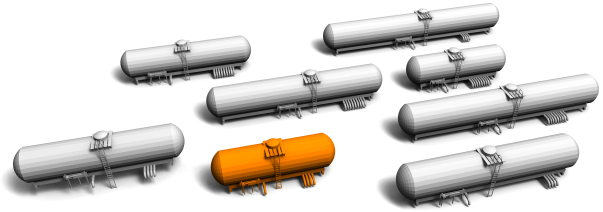
bench



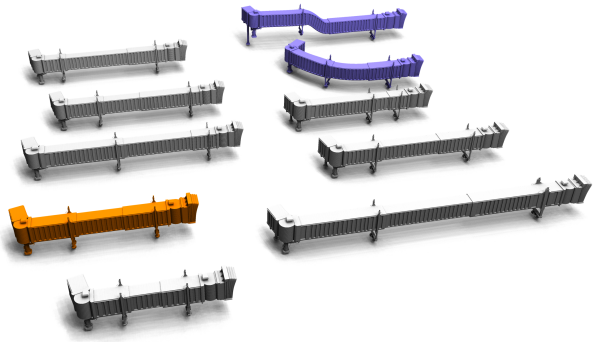
oil platform



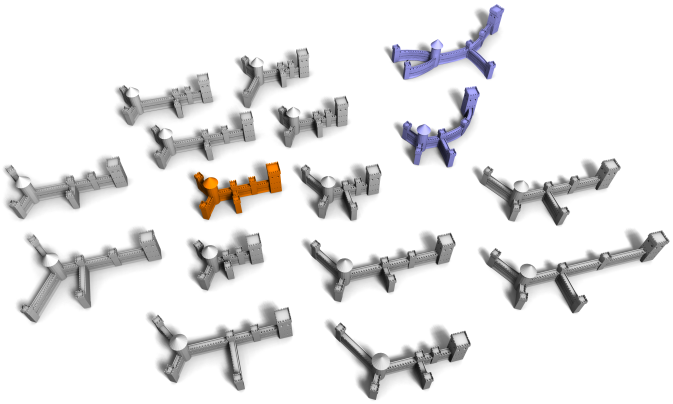
balustrade



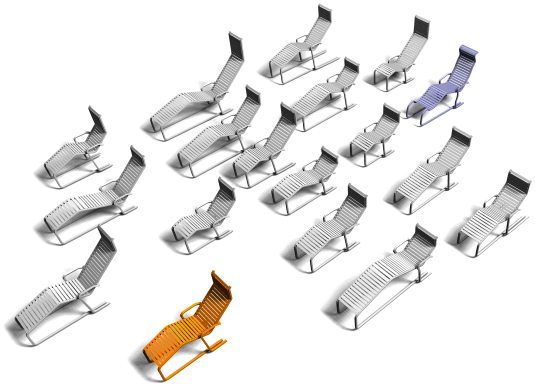
oil tank



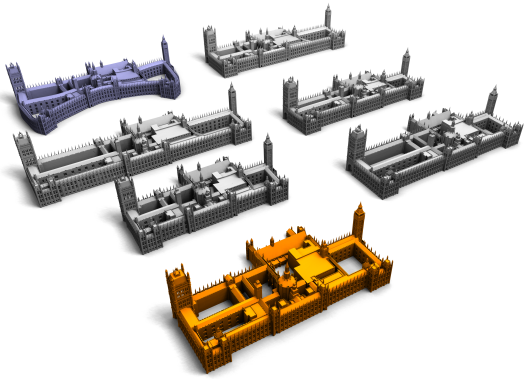
air bridge



castle

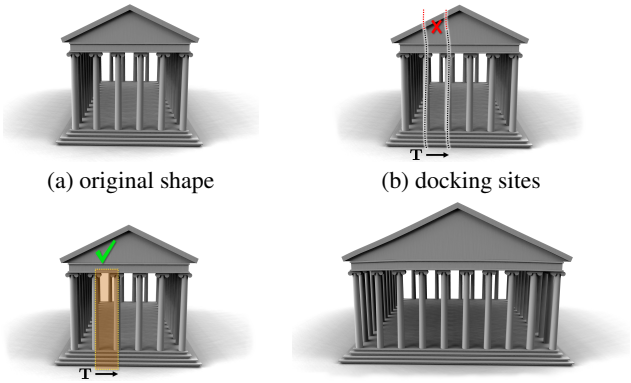


canvas chair



Westminster Palace

Figure 7: *Interactive pattern-aware shape editing. The original input is shown in orange and editing results are shown in grey and blue. The repetition counts of discrete patterns in the edited shapes were automatically adapted by the framework. For the blue models, pattern preservation constraints were disabled and only the elastic energy was used, in order to allow for more severe deformation.*



(a) original shape (b) docking sites

(c) sliding dockers (d) deformation result

Figure 8: *Sliding dockers are detected where no docking sites can be found by the technique of Bokeloh et al. [2010]. Given an input shape, shown in (a), the existence of docking sites is contingent on symmetric cuts that partition the shape into disconnected pieces. As illustrated in (b), such cuts are not found in the temple due to the shape of the roof. On the other hand, sliding dockers do not require global symmetries and are successfully detected (c) and applied by our technique (d).*

better base regularizer for object parts where no patterns are found. The global constraints are necessary to keep objects straight; without them, global bending cannot be prevented.

Comparison to related work: Figure 7 shows the "oil tank" model used by Kraevoy et al. [2008]. Our technique achieves comparable results, while detecting the stretch axes fully automatically. (The shape does not need to be aligned with the global coordinate axes.) Since we only penalize structural deviations in a least-squares sense, a small amount of residual bending remains. The "castle" example in Figure 7 demonstrates that our approach is more general: The castle can be stretched in non-orthogonal directions that are determined fully automatically by our pattern-aware structure model.

In comparison to Bokeloh et al. [2010], our approach can adapt the discrete structure of the model in real time in response to continuous free-form deformation, instead of being driven by manually specified rigid shape operations. Furthermore, sliding dockers are found in examples where the analysis technique of Bokeloh et al. fails to detect global cuts, such as the arches in Figure 2 and the columns in Figure 8. In the latter example, the temple roof is not partially symmetric under transformation T and thus no docking sites can be detected, as shown in Figure 8(b). On the other hand, sliding dockers do not rely on global cuts, allowing the temple to be resized as shown in Figure 8(d).

8 Discussion

We have presented a structure-aware deformation technique that uses the elementary assumption of preserving regular patterns, which we model as 1-parameter partial symmetry groups. We have developed a variational optimization technique that preserves such structures in a least squares sense, while distributing the remaining stretch uniformly. In addition, we introduced *sliding dockers* that allow the technique to fully automatically insert or delete repeated elements in discrete patterns in order to minimize distortions due to free-form deformation. Furthermore, we have presented a numerical framework that uses a subspace formulation with prefactored linear systems to implement the presented approach efficiently and robustly in a real-time system.

Limitations: One limitation of the current approach is the handling of small-scale irregularities in the input 3D model. Objects that appear perfectly regular often have geometric inconsistencies because the artist did not accurately align parts of the original model; errors of 10% are not uncommon. While we can compensate for this by introducing a small numerical threshold in the pattern detection algorithm, irregularities in the input can cause the shape analysis stage to overlook visually salient patterns. For the Westminster Palace model, we had to manually adjust the original geometry in one place to repair a single discrete pattern. Making the analysis stage more robust to approximate regularity is a natural avenue for future work, possibly using a feature graph matching approach [Bokeloh et al. 2009]. A further limitation, and another interesting avenue for future work, is that all sliding docker groups currently need to be mutually disjoint and can thus have only a single repetition parameter (one-parameter grids). Handling discrete changes of two- and three-parameter grids as well as more general overlapping and hierarchical patterns could extend the applicability of our approach to more complex structures.

Our current implementation uses a regular sampled deformation field, which can lead to distortions when opposing constraints are spatially close. In Figure 9, we show the most noticeable artifacts of the current system, caused by insufficient resolution in the deformation field. Here, opposing pattern constraints act on the same deformation nodes and the system opts for an equilibrium between the constraints, resulting in undesired distortions. This could be alleviated using an adaptive deformation field that increases the resolution locally at difficult locations.

Another limitation of the presented approach is that it only handles translations. In future work, we would like to investigate structure models that utilize more invariant notions of similarity, possibly incorporating rotation, scaling, or more general invariants such as intrinsic isometries. Our current implementation also imposes discrete constraints on individual sliding docker groups and does not connect multiple groups that form linked patterns. Finally, the presented work focuses on 1-parameter patterns: Future work could pursue a more comprehensive representation of the algebraic structure of partial symmetries for shape deformation. Lifting these limitations can meaningfully advance the capabilities of interactive shape editing tools.

Acknowledgments

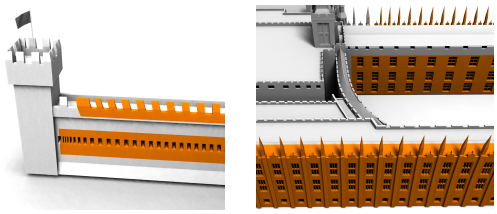
This work has been supported by the cluster of excellence "Multimodal computing and interaction" and the Max-Planck-Center for visual computing and communication.

References

ADAMS, B., OVSJANIKOV, M., WAND, M., SEIDEL, H.-P., AND GUIBAS, L. J. 2008. Meshless modeling of deformable shapes and their motion. In *Symposium on Computer Animation*.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, ACM, New York, NY, USA, 587–594.

BEN-CHEN, M., WEBER, O., AND GOTSMAN, C. 2009. Variational harmonic maps for space deformation. *ACM Transactions on Graphics 28*, 3.



(a) (b)

Figure 9: *Undesired distortions can appear where competing constraints affect the same region. This is often caused by insufficient resolution of the deformation field. (a) Two neighboring patterns (orange) starting at different positions result in opposing constraints that are not handled well with a low frequency deformation field. (b) Undetected patterns and insufficient resolution in the deformation field can cause large distortions.*

BOKELOH, M., BERNER, A., WAND, M., SEIDEL, H.-P., AND SCHILLING, A. 2009. Symmetry detection using line features. *Computer Graphics Forum 28*, 2.

BOKELOH, M., WAND, M., AND SEIDEL, H.-P. 2010. A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph. 29* (July), 104:1–104:10.

BOTSCH, M., AND SORKINE, O. 2008. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1, 213–230.

COQUILLART, S. 1990. Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proc. Siggraph*.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. *ACM Trans. Graph. 23*, 3.

GAL, R., SORKINE, O., MITRA, N., AND COHEN-OR, D. 2009. iwires: An analyze-and-edit approach to shape manipulation. *ACM Trans. Graph. 28*, 3.

GELFAND, N., AND GUIBAS, L. 2004. Shape segmentation using local slippage analysis. In *Proc. Symp. Geometry Processing*.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. *ACM Trans. Graph. 25*, 3.

HUANG, Q., MECH, R., AND CARR, N. 2009. Optimizing structure preserving embedded deformation for resizing images and vector art. In *Pacific Graphics*.

JOSHI, P., MEYER, M., DEROSE, T., GREEN, B., AND SANOCKI, T. 2007. Harmonic coordinates for character articulation. *ACM Trans. Graph. 26* (July).

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. *ACM Trans. Graph. 24* (July), 561–566.

KRAEVOY, V., JULIUS, D., AND SHEFFER, A. 2007. Shuffler: Modeling with interchangeable parts. In *Pacific Graphics 2007*.

KRAEVOY, V., SHEFFER, A., SHAMIR, A., AND COHEN-OR, D. 2008. Non-homogeneous resizing of complex models. *ACM Trans. Graph. 27*, 5, 1–9.

LIPMAN, Y., LEVIN, D., AND COHEN-OR, D. 2008. Green coordinates. *ACM Trans. Graph. 27* (August).

LIU, L., ZHANG, L., XU, Y., GOTSMAN, C., AND GORTLER, S. 2008. A local/global approach to mesh parameterization. *Computer Graphics Forum 27*, 5, 1495–1504.

MITRA, N. J., AND PAULY, M. 2008. Symmetry for architectural design. In *Advances in Architectural Geometry*, 13–16.

MITRA, N. J., GUIBAS, L. J., AND PAULY, M. 2006. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph. 25*, 3, 560–568.

MÜLLER, M., DORSEY, J., MCMILLAN, L., JAGNOW, R., AND B., C. 2002. Stable real-time deformations. In *Proc. Symp. Computer Animation (SCA)*, 49–54.

PAULY, M., MITRA, N., GIESEN, J., GROSS, M., AND GUIBAS, L. J. 2005. Example-based 3d scan completion. In *Proc. Symp. Geometry Processing*.

PAULY, M., MITRA, N. J., WALLNER, J., POTTMANN, H., AND GUIBAS, L. 2008. Discovering structural regularity in 3D geometry. *ACM Trans. Graph. 27*, 3.

PODOLAK, J., SHILANE, P., GOLOVINSKIY, A., RUSINKIEWICZ, S., AND FUNKHOUSER, T. 2006. A planar-reflective symmetry transform for 3D shapes. *ACM Trans. Graph. 25*, 3.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Proc. Siggraph*, 151–160.

SIMARI, P., KALOGERAKIS, E., AND SINGH, K. 2006. Folding meshes: hierarchical mesh segmentation based on planar symmetry. In *Proc. Symp. Geometry Processing*, 111–119.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 109–116.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Symposium on Geometry processing*.

SUMNER, R. W., SCHMID, J., AND PAULY, M. 2007. Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3.

TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. 1987. Elastically deformable models. In *Proc. SIGGRAPH '87*, ACM, New York, NY, USA, 205–214.

TOLEDO, S., 2003. Taucs: A library of sparse linear solvers. Tel-Aviv University, http://www.tau.ac.il/ stoledo/taucs/.

VON FUNCK, W., THEISEL, H., AND SEIDEL, H.-P. 2006. Vector field based shape deformations. *ACM Trans. Graph. 25*, 3.

WANG, Y., XU, K., LI, J., ZHANG, H., SHAMIR, A., LIU, L., CHENG, Z., AND XIONG, Y. 2011. Symmetry hierarchy of man-made objects. In *Proc. Eurographics*.

WU, H., WANG, Y.-S., FENG, K.-C., WONG, T.-T., LEE, T.-Y., AND HENG, P.-A. 2010. Resizing by symmetry-summarization. *ACM Transactions on Graphics 29*, 6.

XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. 2009. Joint-aware manipulation of deformable models. *ACM Trans. Graph. 28*, 3, 1–9.

ZHENG, Y., FU, H., COHEN-OR, D., AU, O. K.-C., AND TAI, C.-L. 2011. Component-wise controllers for structure-preserving shape manipulation. In *Proc. Eurographics*.

ZHOU, K., HUANG, J., SNYDER, J., LIU, X., BAO, H., GUO, B., AND SHUM, H.-Y. 2005. Large mesh deformation using the volumetric graph laplacian. *ACM Trans. Graph. 24*, 3, 496–503.