Structure-aware content creation

Detection, retargeting and deformation

Thesis for obtaining the title of
Doctor of Natural Science of
Faculty of Mathematics and Computer Science of
Saarland University

Vorgelegt von

Xiaokun Wu
吴晓堃

am 25. August 2016 in Saarbrücken, Deutschland

Betreuender Hochschullehrer – Advisor
Prof. Dr. Hans-Peter Seidel

Gutachter – Reviewer
Prof. Dr. Hans-Peter Seidel
Prof. Dr. Michael Wand
Prof. Dr. Klaus Hildebrandt
Prof. Dr. Reinhard Klein

Dekan – Dean
Prof. Dr. Frank-Olaf Schreyer

Kolloquium – Examination

Datum – Date:
20. Jan. 2017

Vorsitzender – Chair:
Prof. Dr. Philipp Slusallek

Prüfer – Examiners:
Prof. Dr. Hans-Peter Seidel
Prof. Dr. Michael Wand
Prof. Dr. Klaus Hildebrandt

Protokoll – Reporter:
Dr. Qianru Sun

Abstract

Nowadays, access to digital information has become ubiquitous, while three-dimensional visual representation is becoming indispensable to knowledge understanding and information retrieval. Three-dimensional digitization plays a natural role in bridging connections between the real and virtual world, which prompt the huge demand for massive three-dimensional digital content. But reducing the effort required for three-dimensional modeling has been a practical problem, and long standing challenge in compute graphics and related fields.

In this thesis, we propose several techniques for lightening up the content creation process, which have the common theme of being structure-aware, i. e., maintaining global relations among the parts of shape. We are especially interested in formulating our algorithms such that they make use of symmetry structures, because of their concise yet highly abstract principles are universally applicable to most regular patterns.

We introduce our work from three different aspects in this thesis. First, we characterized spaces of symmetry preserving deformations, and developed a method to explore this space in real-time, which significantly simplified the generation of symmetry preserving shape variants. Second, we empirically studied three-dimensional offset statistics, and developed a fully automatic retargeting application, which is based on verified sparsity. Finally, we made step forward in solving the approximate three-dimensional partial symmetry detection problem, using a novel co-occurrence analysis method, which could serve as the foundation to high-level applications.

Kurzzusammenfassung

Jetzt hat die Zugang zu digitalen Informationen allgegenwärtig geworden. Dreidimensionale visuelle Darstellung wird immer zum Einsichtsverständnis und Informationswiedergewinnung unverzichtbar. Dreidimensionale Digitalisierung verbindet die reale und virtuelle Welt auf natürliche Weise, die prompt die große Nachfrage nach massiven dreidimensionale digitale Inhalte. Es ist immer noch ein praktisches Problem und langjährige Herausforderung in Computergrafik und verwandten Bereichen, die den Aufwand für die dreidimensionale Modellierung reduzieren.

In dieser Dissertation schlagen wir verschiedene Techniken zur Aufhellung der Erstellung von Inhalten auf, im Rahmen der gemeinsamen Thema der strukturbewusst zu sein, d. h. globalen Beziehungen zwischen den Teilen der Gestalt beibehalten wird. Besonders interessiert sind wir bei der Formulierung unserer Algorithmen, so dass sie den Einsatz von Symmetrische Strukturen machen, wegen ihrer knappen, aber sehr abstrakten Prinzipien für die meisten regelmäßigen Mustern universell einsetzbar sind.

Wir stellen unsere Arbei aus drei verschiedenen Aspekte in dieser Dissertation. Erstens befinden wir Räume der Verformungen, die Symmetrien zu erhalten, und entwickelten wir eine Methode, diesen Raum in Echtzeit zu erkunden, die deutlich die Erzeugung von Gestalten vereinfacht, die Symmetrien zu bewahren. Zweitens haben wir empirisch untersucht dreidimensionale Offset Statistiken und entwickelten eine vollautomatische Applikation für Retargeting, die auf den verifizierte Seltenheit basiert. Schließlich treten wir uns auf die ungefähre dreidimensionalen Teilsymmetrie Erkennungsproblem zu lösen, auf der Grundlage unserer neuen Kookkurrenz Analyseverfahren, die viele hochrangige Anwendungen dienen verwendet werden könnten.

Contents

Chapter 1

Introduction

In a nutshell, the field of *computer graphics* is concerned with the representation and rendering of 3-dimensional digital content, so one of the major focuses in this field is on technologies used to create and manipulate such content. If we look around, almost every modern artificial product would be designed on computer before final manufacturing. Taking figure 1.1 for example, we can clearly see that, we are living in a world designed by ourself.

Nowadays, our world is connected through an unified digital network, so people are becoming more and more interested in digitalizing the entire human world. This trend is rooted in the fact that modern people are becoming heavily dependent on digital information, either for everyday trivial tasks or for complicated decision-making processes. To meet people's strong demand for information, two necessary infrastructures have to be provided: a data warehouse with comprehensive descriptions of each subject, and an intuitive retrieval interface.

Traditionally, both data storage and retrieval are based on text, which clearly fails to satisfy our expectation. On the one hand, it is very hard to depict every details of a subject purely using text. On the other hand, it is also hard for people to imagine that subject's appearance from those words. Considering vision is the most informative human sense, it is inevitable that visual representation is becoming a popular information carrier of digitization, which also establishes the bridge between the real and the virtual world.

One of the most common information retrieval application in our daily lives is map service. Just taking a look at our vicinity, as shown in online maps [Google], we can claim that almost every single activity of our daily lives relates

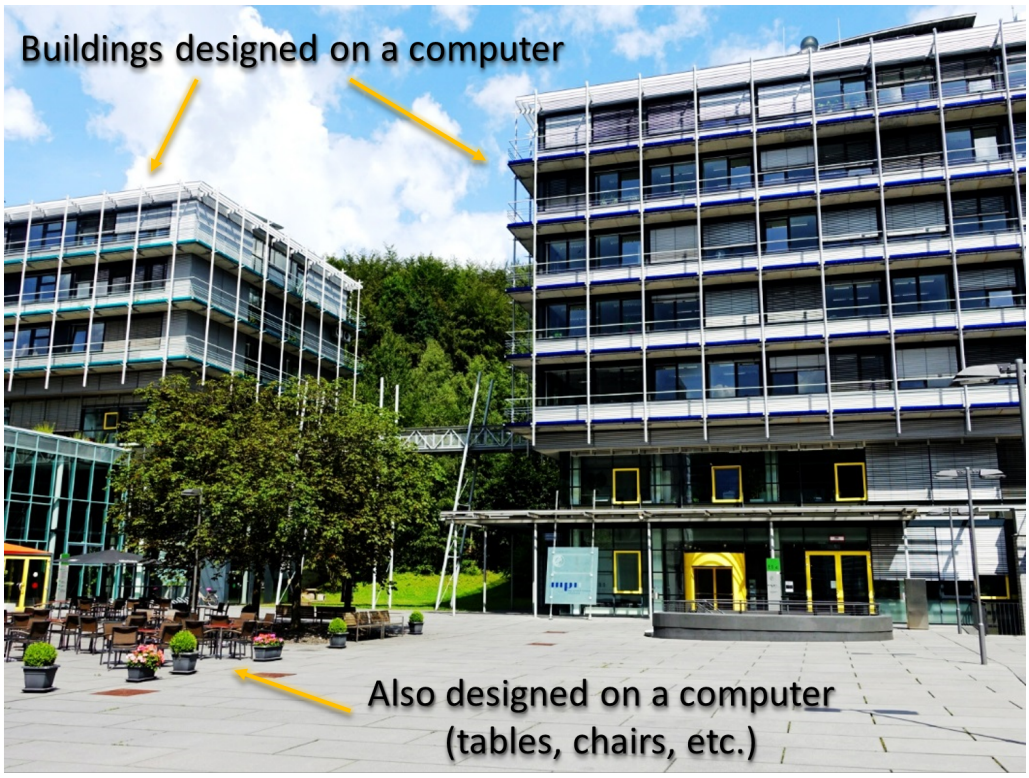


Figure 1.1: Almost every man-made objects are more or less related to computer aided geometric processing

to retrieving information from the map: such as looking for a shopping center, and also searching for the most economic transportation to reach it.

Satellite imagery becomes a very popular way of geographical information digitization, is largely due to the fact that this information acquisition method is relatively easily realizable, considering our current status of hardware technology. But on closer examination we might still be confused by it, because this 2D representation lacks stereoscopic look-and-feel. The extra details provided by high fidelity only add up to disturbing our attentions. In contrast, intuitive 3-dimensional representation could help to solve the orientation deficiency problem.

Introducing 3-dimensional content into the digital world is both interesting and challenging. Back to the map service example, another provider [Aladdin] illustrated the map content using a very stylish way of non-realistic modeling. One may find that this new appearance looks much cleaner: it exhibits streets

and buildings in a very concise manner, but conveys equally comprehensive information at the same time.

Obviously a properly large 3D digital content data bank is necessary for such a massive scene. So the first question that came into our mind is: how to economically generate those tremendous amount of 3D digital contents? Exploring effective methods for addressing that problem is also the main theme of this thesis.

1.1 3D digital content creation

The development of 3D modeling tools has progressed constantly since the beginning of computer graphics, but only in recent years people started studying fast massive model production techniques.

Direct modeling The most intuitive and naïve way to model digital shapes is simply creating everything from scratch, which is exactly the approach adopted by [Aladdin]. But one can imagine that even for a well-trained artist, this brute-force approach means extremely high amount of manual work. The modeling task can quickly becomes impossible, when the load incurred by scale of digital world surpass its undertaker's capacity.

Example based modeling In the latest years, researchers are trying to develop semi-automatic or even fully automatic methods to speed-up this modeling process [Parish and Müller, 2001; Bokeloh et al., 2010a; Mitra et al., 2013a]. Most of them belong to the *example based modeling* category, which assumes a given example geometry could be decomposed into a small set of elementary parts and generating rules. Those rules could be applied to reassemble input parts together, or even create new amplified complex. As a natural inference, the key to its success relies on the correctness of examples' structural analysis, and an effective way of utilizing the rules extracted from the input.

The input example is *reducible*, if it is larger than the union of its elementary parts. Typically, one possible prerequisite that could lead to a reducible example is geometric redundancy. Redundancy means if we broke the example model into any combination of smaller parts, we can always find at least two pieces that are exactly the same. On the other hand, we also assume those repeated observation of constituent parts conform to certain identifiable regular pattern, so even if a small portion of that model is missing, we can still recreate the

complete model using those rules. The assumption of *geometric redundancy* and *pattern regularity* is the basic motivation of our work.

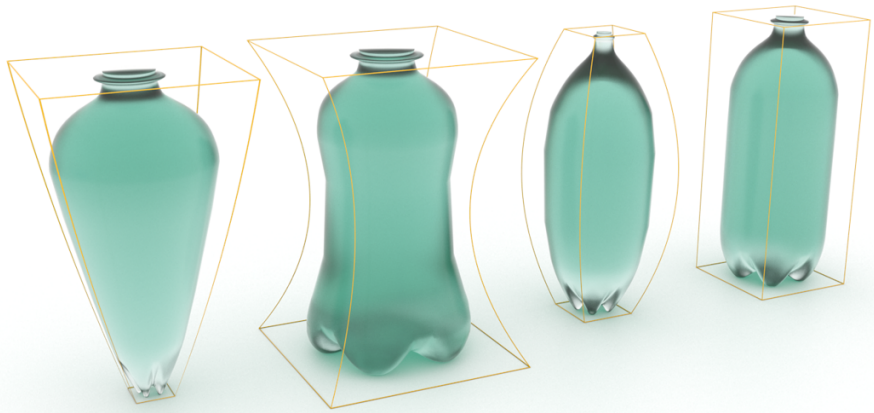


Figure 1.2: Maximize variance, while keeping shape structure.

For example, as shown in 1.2, those bottles looked different, but actually all of the left three are derived from the right most neutral template shape. This creating process relates to human's ability of abstracting structures from given input model, and then infer new variances within acceptable perceptual range. Specifically speaking, the structure entailed here is (rotational) symmetry, which is a mathematically well defined and actively studied tool in recent work. We will elaborate the foundations in Chapter 2, and also detailed applications thereafter.

Laser scan technique Example collection through direct modeling is a very tedious work, so people have been looking for alternative data acquisition methods. One of the most successfully applied technique is 3D scanning, which has been widely used in applications from world heritage protection to real-time city scanning. The data acquired is normally stored as 3-dimensional point array, which is also called *point cloud*, and its level of detail depends on scanning density (Figure 1.3).

The data acquired often exhibit two serious problems as shown in Figure 1.3. First, light always travel in a straight line, so the area behind cars or other occluding objects can not be imaged. Second, different materials have their own reflection and refraction property, and the scanner has its own mechanical errors, so noisy patches are expected amongst deviated data.

Reconstructing geometry from such messy data also benefit from the information redundancy and pattern regularity assumptions. To consolidate the data, recurring patches with matching features surrounding missing area are good completion hypothesis. However, inaccurate input data needs special treatment, so we will discuss it in detail when we come to structure detection problem in Chapter 5.

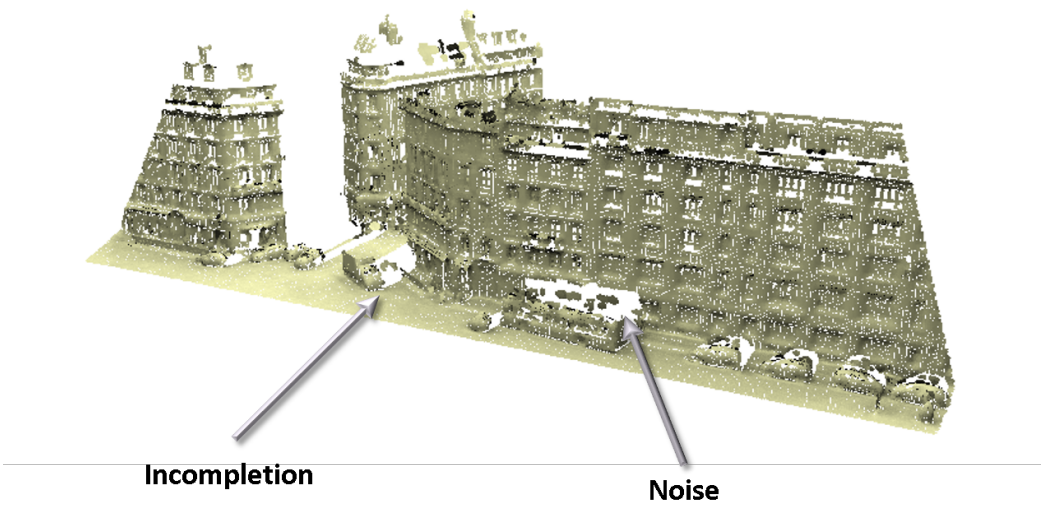


Figure 1.3: Even best quality scanning data available still exhibit serious incompletion and noise problems.

1.2 Contributions

We contribute some concrete methodologies of addressing several specific problems, which could be found in the following publications (this thesis is based on the last three):

- Kurz, C., Wu, X., Wand, M., Thormählen, T., Kohli, P., and Seidel, H.-P. (2014). Symmetry-aware template deformation and fitting. *Computer Graphics Forum*, 33(6):205–219

- Wu, X., Li, C., Wand, M., Hildebrandt, K., Jansen, S., and Seidel, H. (2014a). 3d model retargeting using offset statistics. *IEEE 3DV*
- Wu, X., Wand, M., Hildebrandt, K., Kohli, P., and Seidel, H.-P. (2014b). Real-time symmetry-preserving deformation. *Comput. Graph. Forum*, 33(7):229–238
- Li, C., Wand, M., Wu, X., and Seidel, H. P. (2015). Approximate 3d partial symmetry detection using co-occurrence analysis. In *3D Vision (3DV), 2015 International Conference on*, pages 425–433

In [Wu et al., 2014b], we focus on the problem of structure-aware shape deformation, which accommodates classical finite symmetry group properties into applications. The main contributes of this work are:

- characterizing the shape spaces of fixed symmetry as affine spaces.
- direct construction of basis of spaces of symmetry-preserving deformation, through an effective sampling strategy.
- simple and efficient shape editing algorithms, based on numeric computation methodology.

Then we turned our attention to the 3-dimensional model synthesis problem in [Wu et al., 2014a], by exploiting translational invariance. We made efforts to

- carry out an empirical study that confirms the sparsity of offset statistics in man-made 3D shapes.
- present an algorithm for detecting dominant offsets from single input models.
- propose a 3D model synthesis algorithm that utilizes offset statistics for automatic, high quality model retargeting at interactive speed.

We continued our exploration into the depth of low level shape analysis and the structure detection problem in [Li et al., 2015], and finally arrived at an algorithm for partial symmetric structure detection. Our main contributions can be summarized as:

- We introduce consistently co-occurring patterns as a novel structural invariance, which remarkably improved feature matching for 3D point clouds.
- We propose a new unsupervised partial symmetry detector, which outperforms previous methods in the difficult case of strong geometric variability.

1.3 Thesis outline

In the following text, we will discuss some background and review previous work in Chapter 2. Then from Chapter 3 to Chapter 5, our work are elaborated in details. More specifically, Chapter 3 is based on [Kurz et al., 2014] and [Wu et al., 2014b], Chapter 4 is based on [Wu et al., 2014a], while Chapter 5 is based on [Li et al., 2015] and some further extensions we have made to it. Finally, Chapter 6 concludes our discussion, and tries to pointing out some possible directions of future work.

Chapter 2

Background and previous work

In this chapter, we will briefly summarize relevant background knowledge, as well as selected state-of-the-art approaches. First, as pointed out in Chapter 1, symmetry has becoming widely adopted in shape analysis. The success of symmetry-based approaches in computer graphics is not only due to symmetry is a well-studied subject in mathematics, but also for the reason that symmetry is ubiquitous in both artificial and natural world.

So in the following we will discuss some symmetry related topics, which act as the main clue and will occur frequently throughout the rest of this thesis. After that, we will list interesting related work that had inspired us.

2.1 Symmetry

Throughout this thesis, we only consider problems in the 3-dimensional space $\mathbb{R}^3$. $\mathbb{R}^3$ is a metric space, where the norm is defined by Euclidean distance.

Our problem domain is usually a surface S in $\mathbb{R}^3$, which is given by an embedding $x : M \to \mathbb{R}^3$ that maps from a two-manifold M to $\mathbb{R}^3$. Depending on the practice of each different application, we could represent such surface in a discrete way either as triangle mesh or point cloud. In terms of discretization, x is simply the piecewise linear mapping that maps every vertex (or point) to its positions in $\mathbb{R}^3$. So unless otherwise stated, we will also denote S as its discretization, i. e., triangle mesh or point cloud.

In 3D, an *isometry* (or *Euclidean motion*) is a bijective map on $\mathbb{R}^3$ itself, which preserves standard distance. In our discussion, basic isometric mappings that frequently appear are *rigid motions* (translation or rotation) and reflection. These different classes of transformations can be arbitrarily combined using function composition, and they collectively form a subset of Euclidean motions.

The definition of symmetry is always associated with a specific domain. Let's denote $\Psi(S)$ as the set of Euclidean motions, which map S to itself $\varphi : S \to S$, then S is *symmetric* if $S = \varphi(S)$.

Note that here φ is surjective, which makes this definition to be *global* on S. Besides, the equation is exactly established on S (without approximation).

Informally speaking, symmetry in 3D geometry means self-congruence under Euclidean motion. Coincidently, isometric mappings are also called *congruence transformation* in geometry.

Symmetry group

Groups are a mathematical tool discussed in abstract algebra. To form a *group* $< G, \cdot >$ from a set G, specific binary operator $\cdot$ must be defined that needs to satisfy four fundamental *group axioms*:

- **Closure:** the product of any pair of two elements is still contained in G, i. e., $\forall a, b \in G, a \cdot b \in G$.
- **Associativity:** the multiplication order can be arbitrarily changed, as long as the operands are not commuted, i. e., $\forall a, b, c \in G, (a \cdot b) \cdot c = a \cdot (b \cdot c)$.
- **Identity:** there is an unique identity element e in G, such that $\forall a \in G, e \cdot a = a \cdot e = a$ holds.
- **Invertible:** the inverse of any element always exist, and is also contained in G, i. e., $\forall a \in G, \exists! a^{-1} \in G, a \cdot a^{-1} = a^{-1} \cdot a = e$.

In our discussion, groups are usually applied in function space, so by default the group operator $\cdot$ is just function composition $\circ$. For the sake of simplicity, we sometimes only use the underlying set G to denote group $< G, \cdot >$.

In practical applications, we always study finite groups, i. e., the number of elements (*group order*) is finite. Group can be factorized: if a subset H is closed under group operator $\cdot$, then H is a *subgroup* of G.

For example, $\Psi(S)$ under function composition $\circ$ is just the Euclidean symmetry group $E(3)$. The set of symmetries induced by rigid motions forms *special Euclidean group* $SE(3)$, which is a subgroup of $E(3)$.

Symmetry group examples

From application side, a reference list of well classified groups helps identifying repetitive elements, which further contributes to discover hidden geometric patterns. In three dimensions, the list of symmetry groups has been completely collected as *space group symmetries* [Hahn, 2002]. This illustration of spacial configuration is instructive to many natural science branches, especially in crystallography which studies the arrangement of atoms in the crystal solids.

The general formulation of a space group is the combination of *point groups* which keep at least one point fix, and *lattice systems* which are generated by translations. In more details, the elements of point group include involution (identity and inversion), rotation and rotary reflection.

A detailed discussion of space groups in 3D geometry can be found in [Tevs et al., 2014a], where a symmetry detector was developed for exact geometries.

Generalization

In the previous discussion, we have assumed the range of symmetry mapping is the whole input domain, but in many real scenes that is not true. The common solution to this *partial symmetry* case is subdividing the input domain into non-overlapping symmetric regions, and then apply global symmetry algorithms onto each of those regions. It is often helpful to make sure each region is self-contained, i. e., no superset could contain larger number of symmetries. So we also aggregate small regions into larger ones when possible, and the classification of symmetry groups can be a very intuitive clue for this process. In Chapter 3 we will deal with examples with partial symmetry in our application.

As mentioned in Chapter 1, we often encounter cases where input data is noisy. In this case, symmetric parts can not be congruous under any hypothetical mapping. Close-to-exact symmetry also happens in the natural world, as biological growth almost always shows deviations to perfection. This kind of *fuzzy* (or *approximate*) symmetry is discussed in [Mitra et al., 2013b], where some error estimation methods have been described. We will also discuss our treatment in Chapter 5.

2.2 Shape deformation

Shape deformation has become an indispensable tool in shape editing, which is one of the most important way of creating new variants of example geometry.

In Chapter 3 we will consider a special class of deformations that preserve symmetries during shape editing. This idea leads to an efficient interactive algorithm for generating structurally similar shapes. But before describing our approach we would like to survey previous work, and point out their limitations that we are attempting to improve.

Early methods used low-dimensional spaces of low-frequency deformations spanned by spline-bases that were explicitly controlled by the user [Sederberg and Parry, 1986; Coquillart, 1990]. With increasing level-of-detail, these spaces become too large to be navigated explicitly by the human modeler, which led to a wide-spread adaptation of variational deformation models that are usually based on mimicking physical processes such as elastic or plastic deformations [Terzopoulos et al., 1987; Welch and Witkin, 1992], or rely on smoothness assumptions for more general shape deformation [Alexa et al., 2003; Brown and Rusinkiewicz, 2007]. Geometric approximation such as Laplace surface modeling or as-rigid-as-possible deformations have recently become particularly popular in this context [Sorkine et al., 2004; Au et al., 2006; Sorkine and Alexa, 2007; Botsch and Sorkine, 2008]. Computational costs can be reduced by subspace methods [Huang et al., 2006; Hildebrandt et al., 2011; Jacobson et al., 2012; von Tycowicz et al., 2013] that restrict the set of feasible deformations to a low-dimensional subspace.

Structure-aware deformation Structure-aware deformation has been introduced in image processing through seam-carving [Avidan and Shamir, 2007] for image retargeting. A similar idea for shape-resizing has been introduced by Kraevoy et al. [2008], using differential properties of shapes to detect suitable deformation directions and a vulnerability score to protect complex structures. However, this method only permits axis-aligned stretching of models and only protects salient area from deformation, rather than maintaining non-local symmetry relations. Retargeting has also been extended to noncontinuous operations [Lin et al., 2011a; Bokeloh et al., 2011a], which is beyond the scope of our discussion. Other deformation constraints include slippability for joint-detection [Xu et al., 2009a].

The “iWires” system by Gal et al. [2009] is based on the observation that geometric relations such as parallel lines and planes, right angles, and symmetric and regular parts are characteristic for man-made-shapes. This method uses a list of Euclidean invariants and a greedy propagation algorithm to maintain these shape properties. Most of these invariants arise from the more concise assumption of preserving the *symmetry structure* of the input model, i. e., the

algebraic relation between transformations that form regular correspondences within a shape [Kurz et al., 2014].

Symmetry-aware deformation Symmetry has been utilized as a tool for structuring and guiding shape editing in a number of recent shape editing approaches: Zheng et al. [2011] enforce symmetry of object-aligned proxies for intuitive shape manipulation. Wang et al. [2011] use hierarchical propagation of attributes in objects with complex symmetry patterns for structure-preserving shape editing.

Bokeloh et al. [2011a; 2012a] use translational regularity as invariant for continuous shape deformation (we do not consider their additional option to model topological changes here). The main drawback of the first method is that it is a least-squares approach. Computational costs are still considerable despite the involved numerical treatment in their work. Symmetries are maintained only approximately and traded-off against user constraints. In practice, this leads to the problem that the user needs to choose constraints that are roughly satisfiable; unsatisfiable constraint will lead to bending artifacts. Increasing the penalties for structure constraints reduces these but a large spread in penalties soon leads to ill-conditioned optimization problems. The second method [Bokeloh et al., 2012a] avoids these by fixing the null-space of the deformation energy through a singular-value-decomposition (SVD). However, the SVD creates a dense basis, which limits the size of models that can be handled due to the $\Omega(n^2)$ costs involved for n vertices.

Our new method in Chapter 3 overcomes this problem by directly constructing a sparse basis. Further, our technique handles general Euclidean motions, not being limited to translational symmetry. The work of Kurz et al. [2014] is targeting at shape matching rather than editing. A least-squares formulation that is inaccurate and multiple orders of magnitude slower than our new approach.

Symmetric bases Symmetry-invariant scalar functions have been studied by Lipman et al. [2010]. Symmetry-invariant functions obviously form a linear space (linear combinations of symmetric functions maintain the symmetry property), and they propose a spectral method for determining these spaces. Ovjanikov et al. [2013] use the subspace property to factor out distracting variability in the context of shape matching. The concept has various further applications, such as symmetry-based shape descriptors [Kazhdan et al., 2004a], and robust intrinsic symmetry detection [Wang et al., 2014].

In Chapter 3, our work extend these previous ideas towards *shape deformation*. We will show that symmetry-preserving deformation functions form affine spaces (more specifically, the displacement fields added to the input form a linear space). Unlike the scalar case, local frames induced by the group action need to be taken into account to preserve symmetries. Assuming knowledge of the symmetry groups, our approach can directly construct a suitable basis without any need for expensive eigenvalue decompositions. This construction provides a user-defined level-of-detail and, unlike spectral methods, can preserve the sparseness of traditional spatial bases for deformation fields.

2.3 Model retargeting

Deformation based shape editing can not cover every aspects of 3D modeling. Sometimes we want to add or remove elements, change model topology, etc. In this section we would like to introduce model retargeting based content creation.

Retargeting means given an example described in predefined domain, synthesizing output with similar content in a different sized domain (usually up-scaling). Retargeting arise from a very practical 2D problem: we often need to display images without distortion or important information lost, on screens of various sizes or aspect ratio. So the image would not looks much different either on a cellphone with portrait orientation, or on a projector screen with landscape orientation. Seam carving [Avidan and Shamir, 2007] is a representative algorithm for image resizing, which is optimized to keep content while cropping the image.

We are interested in extending this idea to the problem of 3D content creation. The difficulties and our solution in a large shape category will be explained in Chapter 4, while related work is listed in the following.

Model assembly This category of methods decompose an input model into parts and recombine them from different sources into new shapes. Early work from [Funkhouser et al., 2004] allowed parts to be assembled interactively to form new shapes. Shapes can also be created by interpolating parts from two exemplar shapes, based on parts matching [Jain et al., 2012] or hand coded substructure [Zheng et al., 2013]. More recently, large collection of shapes have been taken into consideration. In [Xu et al., 2012a] novel shapes are produced from the evolution of an initial population of 3D models. Averkiou et al. [2014] analyze unorganized collections of 3D models to facilitate exploratory shape synthesis using high-level feedback. Kalogerakis et al. [2012] use a generative probabilistic model of shape structure trained on a set of compatibly segmented shapes to learn the structural variability.

Rertarget based methods People have made initial attempt to apply MRF-based texture synthesis to 3D modeling. For example, Merell et al. [2011] stitch geometry using adjacency constraint to preserve local consistency. However, MRF-based 3D model synthesis suffers from the problem of lacking knowledge of high level structure, and the computational cost of generating detailed models is high [Merrell, 2009]. In the past decade, great efforts have been made to explore shape structures for high quality 3D synthesis. For example, global symmetries [Mitra et al., 2006a; Pauly et al., 2008b] and partial symmetries [Bokeloh et al., 2010b, 2011b] are extracted from input models, which are used to regularize synthesis model. This forms the basis for inverse procedural modeling, which represents families of shapes by automatically inferred context-free grammars [Bokeloh et al., 2010b]. Bokeloh et al. [2012b] propose an algebraic model that describes shapes in terms of regular patterns, interlinked with fixed topology. The shape space is parameterized by a small set of parameters. The limitation is that their method requires exact regularity (grids of at least 3×3 identical copies) to be applicable. More complicated (architectural) models can be handled using manual user annotations [Lin et al., 2011b].

Texture synthesis 3D texture synthesis has been boosted from fruitful progress recently made in the 2D case [Wei et al., 2009]. Here, various forms of guidance have been employed to overcome the limitations of the MRF models. This can be achieved using an additional image layer with user annotations [Hertzmann et al., 2001] or on-the-fly user interaction [Barnes et al., 2009]. For image retargeting, results can be improved by better metrics for comparing MRFs. For example, [Simakov et al., 2008] uses bidirectional matching and a gradual resizing procedure for retaining the image structure.

We are particularly inspired by the recent success of applying statistics in image editing [Pritch et al., 2009] and in-painting [He and Sun, 2012]. When using this idea for 3D synthesis, we can obtain plausible shape variations that previous 3D texture synthesis algorithms fail to produce without user-guided analysis. This conceptual novelty leads to a simple but robust implementation in Chapter 4 for content-aware 3D shape synthesis.

2.4 Symmetry detection

Since its introduction to the graphics community by a series of seminal papers [Podolak et al., 2006; Mitra et al., 2006c; Simari et al., 2006; Gal and Cohen-Or, 2006], symmetry detection and its applications has attracted many attentions [Mitra et al., 2013b]. We focus on key ideas to deal with geometric variability,

which guided us in Chapter 5 towards an algorithm for detecting approximate symmetries.

Transformation voting Methods in this category equip every data points with a transformation-invariant local descriptor, then count transformation votes casted from matching feature pairs [Mitra et al., 2006b; Podolak et al., 2006; Loy and Eklundh, 2006]. Voting methods excel at recognizing approximate symmetry, which is shown clearly in [Mitra et al., 2007]. However, the use of a single transformation space (marginalizing out location information) limits their detection to coarse scale structures. Details and/or larger numbers of instances can not be easily identified out of structured background noise in the transformation space. This problem can be alleviated by a more restrictive hierarchical decomposition [Mitra et al., 2006b] or additional regularity priors [Pauly et al., 2008a].

Feature-graph matching This line of work avoids the prerequisite of common transformation space [Berner et al., 2008; Bokeloh et al., 2009]. Their methods can be understand as brute-force direct alignment schemes that uses constellations of local features as filters, i. e., rapidly reject clearly non-matching geometry. Final validation is always obtained via ICP alignment [Chen and Medioni, 1992]. With a careful design, feature-graph matching can providing near-linear scaling behavior [Kerber et al., 2013]. As these methods rely on direct rigid alignment, they are able to yield good detection results on exact data (even in the presence of substantial noise), but direct alignment prohibits systematic shape deformation. Attempts have been made to handle deformable feature graphs [Berner et al., 2009, 2011]. However, results on real-world data were rather limited because they lack an *efficient* way of validation, when conjectured patterns have large supporting areas in the data.

Spectral clustering Lipman et al. [2010] recognize symmetric features as clique members in a matching graph, or equivalently, estimate a transitively (cycle-) consistent equivalence relation between features [Huang et al., 2012]. Spectral embedding is used to recover the latent equivalence relation from noisy data. The method does not yet address the problem of how to find the support region of the match, i. e., how to determine the area of the match during the optimization. Kalojanov et al. [2012] show that building blocks (coined "microtiles" in their paper) have consistent constellations of matches under the transformations that relate them. However, their model cannot handle approximate matching (it even yields artifacts on exact data due to numerical noise). Spectral clustering has been extended by Kim et al. [2012] towards shape collections. They apply diffusion in a sparse matching graph, which avoids unreliable long-distance matches. Mattausch et al. [2014] cluster feature patches and incor-

porate co-occurrence by adding a pairwise geometric consistency term, which is in line with [Leordeanu and Hebert, 2005]. This model does not try to find consistent co-occurrence patterns that characterize specific building blocks, i. e., different arrangements of elementary feature constellations would be treated the same. Their feature design is optimized for indoor architecture (planar patches), and requires a global upward direction. A similar approach is employed by Tam et al. [2014] in an intrinsic setting, where individual point-orbits are assembled using a greedy algorithm.

Isometric matching Invariance to near-isometric bending can be obtained by an intrinsic formulation. Ovsjanikov et al. pioneered this approach using a global Laplacian embedding [2008], then partial matchings have been considered afterwards [Xu et al., 2009b; Mitra et al., 2010; Kim et al., 2011; Xu et al., 2012b]. Building block area can be optimized using boundary length regularization [Raviv et al., 2010].Huang et al. [2014a] detect near-regular intrinsic patterns in shapes with strong geometric variability. To our knowledge, this is the only method available to date that can automatically detect subtle repetitive features, such as the scales on the Stanford dragon. However, it requires near-regular placement of instances (a deformed grid), and only works on manifold geometry.

Co-segmentation Co-segment model collections has received a lot of attention recently. [Kalogerakis et al., 2010] learn segmentation from examples, and [Golovinskiy and Funkhouser, 2009; Huang et al., 2011; Sidi et al., 2011; Hu et al., 2012] extend this idea to unsupervised optimizations. Laga et al. [2013] add geometric relations in a conditional random field model, which combines local feature matching with pairwise neighbor consistency of discrete elements. Demir et al. [2015] step further to couple segmentation and similarity detection together, which can reveal shape similarities for architectural models.

Other approaches Geometric moments can be used to detect symmetry [Kazhdan et al., 2004b; Martinet et al., 2006], but these methods are limited to global symmetry. Recursive folding [Simari et al., 2006] can be used for hierarchical bilateral similarity, with similar restrictions to other direct alignment methods. Continuous symmetry can be directly detected using differentials of alignment energies [Gelfand and Guibas, 2004], which is beyond our focus. [Xu et al., 2013; Liu et al., 2014; Chen et al., 2014; Zheng et al., 2014] reverse engineer the high-level organization of scenes through geometric invariants or shape grammars. Fish et al. [2014] characterize shape families by marginal distributions of geometric properties, which is related to generalized co-occurrence model that also leverages pairwise marginal distributions of relating geometry. Functional maps [Ovsjanikov et al., 2012] consider first order marginals of correspondences, which form a convex Euclidean space. Various matching cues (date terms) can be

integrated through simple linear algebra tools. Regularization is more involved and requires a careful design of basis functions and additional constraints. For example, [Huang et al., 2014b] shows how cycle consistency is linked to low-rank maps.

Chapter 3

Real-Time Symmetry-Preserving Deformation

In this chapter, we specifically studied 3D geometric shapes of fixed symmetry structure, and address the problem of structure-aware shape deformations. We also made efforts to alleviate the running time issues due to numerically expensive and poorly conditioned optimization, which normally encountered in alternative approaches for symmetry-preserving shape editing.

3.1 Introduction

As explained in Chapter 1, content creation is one of the remaining open challenges in the field of computer graphics, mainly because the creation of digital 3D models is still tedious, technical, and expensive. In this work we stick to the *structure-aware* approach, and specifically contributes to the area of structure-aware shape *deformation*.

Structure-aware methods try to reduce modeling costs and lift interaction with digital 3D models to a higher semantic level by an analysis-and-synthesis approach: A shape (or a collection of shapes) is first analyzed in order to detect important structural invariants. Afterwards, these constrains are used to narrow down the space of possible shapes considered in editing and shape synthesis, thus permitting the user to obtain plausible shape variations more quickly.

This chapter contributes to the area of structure-aware shape *deformation*. The task here is to find and apply structural constraints during shape editing, thereby making the space of possible deformations more easily navigable.

Several such structure-aware deformation methods have been proposed in the last few years (for example, see [Kraevoy et al., 2008; Xu et al., 2009a; Zheng et al., 2011]). A significant idea in this domain has been introduced by Gal et al. [2009]: Their "iWires" system detects relations of Euclidean geometry (parallelity, orthogonality, symmetry) within shapes, and uses them as invariants during shape deformation. In particular for man-made shapes, this provides a very useful tool to fully automatically determine a large class of plausible shape variations. In more recent work [Bokeloh et al., 2011a; Zheng et al., 2011; Bokeloh et al., 2012a; Kurz et al., 2014], the model has been simplified towards preserving the *symmetry structure* of the model, which reduces the approach to a simple, abstract principle. While such a formulation is formally appealing, it is limited by a complex implementation, approximation artifacts (in least-squares formulations), and computational costs.

In this chapter, we study classes of shapes of fixed symmetry structure. We consider shapes with partial symmetries, i. e., there are one or more groups of affine transformations that are symmetry groups of subset of a shape, i. e., leave these subsets invariant. In particular, this includes the important class of symmetry under Euclidean motions. Extending previous work on symmetry-invariant functions on shapes [Kazhdan et al., 2004a; Lipman et al., 2010; Ovsjanikov et al., 2013; Wang et al., 2014], we show that the set of all deformations that preserve the symmetries of a shape forms an affine subspace of the set of all continuous spatial deformations.

We use this insight to devise an efficient and simple algorithm for constructing a basis for these deformation spaces: We perform Poisson-disc sampling of the domain while replicating points by the group action of the present symmetry groups. Further, we associate each point with the Jacobian of the transformation to handle general Euclidean symmetry, and equip it with a Gaussian radial basis function to span spaces that are band-limiting to the user's preference.

We apply this construction to create a symmetry-preserving subspace deformation method based on Laplace surface editing (in a non-linear variant with co-rotation of the local frame, to account for large deformations). Our technique preserves symmetries *exactly* while nonetheless being able to represent low-frequency deformations with a very small set of basis functions. The basis is redundancy-free in the sense of having symmetry backed-in already; no explicit constraints need to be solved for to maintain symmetry. This, along with the subspace approach, leads to a deformation method that is significantly faster

than previous approaches and at the same time is numerically robust and very easy to implement. We also believe that the idea of a direct construction of a redundancy-free basis in symmetric domains might be useful beyond the application area of shape deformation that we explore as motivating application in this work.

In summary, we make two main contributions: First, we characterize shape spaces of fixed symmetry as affine spaces. Second, we use this observation to directly construct a basis for the space of symmetry-preserving deformations, which leads to very simple and efficient shape editing algorithms.

3.2 Partial Symmetries

In this section, we revisit notions of (partial) symmetry (see Chapter 2 for a more detailed exposition of basic concepts), and introduce notations that we need for deformation problems.

Let us consider a surface in $\mathbb{R}^3$ given by a two-manifold M and an embedding $x : M \to \mathbb{R}^3$. An automorphism of M is a map $\varphi : M \to M$ that is a homeomorphism, i. e., is continuous, bijective and has a continuous inverse. Under the composition of maps, the set of automorphisms of M forms a group that we denote by $\Psi(M)$. Symmetries of M relate to subgroups of $\Psi(M)$.

For our experiments, we are using triangle meshes in $\mathbb{R}^3$. In this case, M is the surface mesh itself (or any isomorphic simplicial manifold) and x is the continuous and piecewise linear map that maps every vertex to its positions in $\mathbb{R}^3$.

Symmetry Any Euclidean motion $g \in E(3)$ can be composed with the embedding x, which results in a new map $g \circ x : M \to \mathbb{R}^3$. Loosely speaking, a (Euclidean) symmetry of the embedded surface (M, x) is subgroup G of $E(3)$ such that every $g \in G$ maps $x(M)$ to itself. More formally, we say that a subgroup $G \leq E(3)$ is a symmetry of (M, x) if there is a subgroup $\Phi \leq \Psi(M)$ such that for every $g \in G$ there is a $\varphi \in \Phi$ such that

$$g \circ x = x \circ \varphi \tag{3.1}$$

and the map $G \to \Phi$ induced by this relation is a group isomorphism. The equality in (3.1) means that $g \circ x$ and $x \circ \varphi$ are the same map from M to $\mathbb{R}^3$.

Remark 1. *Note that this concept of symmetry also works if we relax the assumption that the surface is embedded. It suffices that x is locally an embedding, e. g., an immersion.*

Partial symmetry In addition to symmetries of the whole object, we consider symmetries of parts of the object and call them partial symmetries. This makes the concept more powerful as objects often exhibit only partial symmetries. To define partial symmetries, we consider a submanifold N of M, which need not be connected. Then, the restriction $x\restriction_N$ of x to N is an embedding of N in $\mathbb{R}^3$. A symmetry G of $(N, x\restriction_N)$ is a partial symmetry of (M, x).

Symmetry detection Various symmetry detection methods have been proposed in literature (see for example [Mitra et al., 2013b] for a recent survey). Detection is not a focus of this work; we use both manual annotation as well as automatic detection based on Tevs et al.'s algorithm [Tevs et al., 2014b] for our experiments. The resulting symmetry information is encoded as an annotation of the surface M with regions $S_i \subseteq M, i = 1 \dots m$, in which the geometry $x(M)$ is symmetric with respect to a symmetry group $G_i \leq E(3)$. Each group G_i and region S_i is maximal with respect to set inclusion, in that order (first maximizing the group size, then the area covered). The prioritized maximization implies that the regions S_i can be overlapping or even hierarchically nested; see Figure 3.3. Our current implementation does not handle continuous symmetries, which we leave for future work.

3.3 Symmetry-Preserving Deformation

We describe deformations of the surface by variations of x. For this we use a displacement map $u : M \to \mathbb{R}^3$. Then, the sum $x + u$ describes the deformed surface. Self-intersections are permitted, as it is common for surface deformation methods. The resulting set of displacements forms a vector space. For triangle meshes, deformations are described by the displacements of the vertices. Then, the space of displacements equals $\mathbb{R}^{3n}$, where n is the number of vertices.

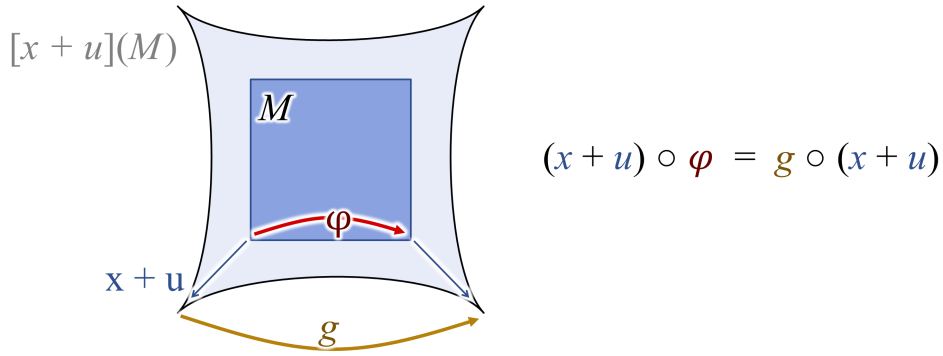


Figure 3.4: If the symmetry transformation commutes with the deformation $x+u$ (the automorphism φ in the input domain turns into the extrinsic map g here), the deformed shape $[x+u](M)$ will have the same symmetry as $x(M)$.

If we have a group g that describes symmetries of the surface, then a displacement u preserves the symmetry if

$$g \circ (x+u) = (x+u) \circ \varphi, \tag{3.2}$$

where φ is the automorphism induced by g. Figure 3.4 illustrates how this condition induces a symmetry-preserving deformation field; see Kurz et al. [2014] for more details.

The basis of our surface modeling scheme is the observation that the set of all symmetry-preserving displacements forms a subspace of the vector space of all displacements (and thus the deformations themselves form an affine space).

Lemma 1. *Given a symmetry group G of a surface. The set of symmetry-preserving displacements forms a subspace of the vector space of all displacements.*

Proof. We have to show that for two arbitrary symmetry-preserving displacements u, v and any scalar $\lambda \in \mathbb{R}$, the displacement $\lambda u + v$ preserves the symmetry as well. We proceed in two steps: first we show that $u+v$ is symmetry preserving. Consider an arbitrary $g \in G$, and let φ denote the corresponding automorphism. Then, we will show that $g \circ (x+u+v) = (x+u+v) \circ \varphi$ holds. Since g is a Euclidean motion, there is an orthogonal matrix O and a translation t such that $g(p) = O(p) + t$ for all $p \in \mathbb{R}^3$.

$$\begin{aligned}
& g \circ (x+u+v) = (x+u+v) \circ \varphi \\
\Leftrightarrow \quad & g \circ (x+u+v) + g \circ x = (x+u+v) \circ \varphi + x \circ \varphi \\
\Leftrightarrow \quad & O(x+u+v) + t + O(x) + t = (x+u) \circ \varphi + v \circ \varphi + x \circ \varphi \\
\Leftrightarrow \quad & O(x+u) + t + O(x+v) + t = (x+u) \circ \varphi + (x+v) \circ \varphi \\
\Leftrightarrow \quad & g \circ (x+u) + g \circ (x+v) = (x+u) \circ \varphi + (x+v) \circ \varphi
\end{aligned}$$

In the first step, we used the definition of the symmetry (3.1). The last equation holds by our assumption that u and v are symmetry-preserving. A similar argument shows that λu is symmetry-preserving. □

3.4 Direct Subspace Construction

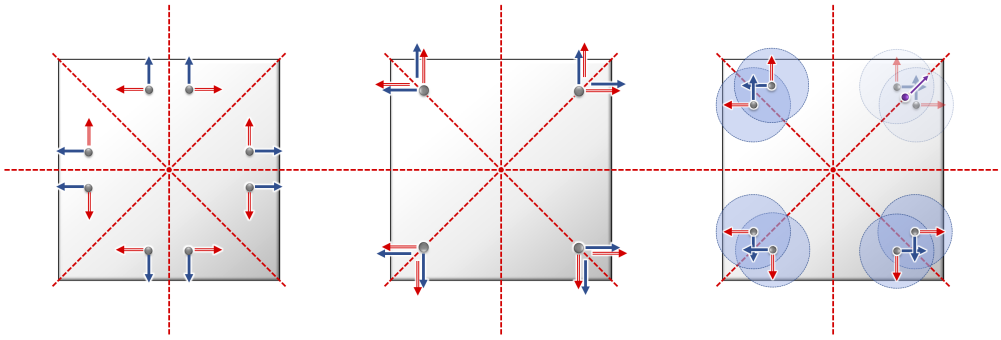
In this section, we introduce an explicit construction of subspaces of the space of symmetry-preserving displacements. For this, we first generate a sampling of the surface that respects the symmetries (including partial symmetries) of the surface. This means every symmetry of the surface is a symmetry of the sampling as well. In addition, the sampling is an r-sampling of the surface, which means that for every point of the surface there is a point in the sampling at distance less than r. As a second step, we construct the space of symmetry-preserving displacements of the point sampling. Finally the displacements of the sampling are propagated to symmetry-preserving displacements of the surface. This construction offers two benefits. Most importantly, we obtain a low-dimensional subspace of the space symmetry-preserving deformations. Even if the underlying mesh is highly resolved, we can control the size of the subspace using the parameter r. This is important for obtaining an editing system that runs in real-time. A second benefit is that even if the input is a set of approximate symmetries, we can create a symmetric sampling. In this case, the sample points are not on the surface, but only close to it. To compute the deformations, we only need to preserve the exact symmetries of the sampling.

Symmetric sampling We propose a symmetry-aware Poisson disc sampling scheme that scatters points sparsely in the domain such that they conform to the underlying symmetry group structures. As input, we are given the surface M, annotated with potentially multiple, and potentially overlapping regions $S_i \subseteq M$ with symmetry groups $G_i \leq E(3)$, as discussed above. We now randomly start with a seed point, chosen with uniform probability from M. We then search for all annotations S_i this point is contained in. Then, for each symmetry group G_i, whose domain S_i contains this point, we add all the corresponding points transformed by its group action into sample set. To handle overlapping and nested symmetries correctly, we form the transitive closure: If the transformed sample point lies within a previously unseen area S_j, we recursively apply all transformations from G_j to this point. This process is continued until no new, previously non-sampled points are discovered anymore (a simple threshold of $10^{-4}\times$ the scene size is used to recognize doublet samples). All of those sample points will later be coupled, moving coherently and not providing more than

three degrees of freedom altogether. For each of those newly added samples, we mark all points with a small radius ϕ_{sam} as invalid, which means they can not be picked later. We redo this process on the remaining point set, and iterate until all the points are either picked into the sample set, or marked as invalid.

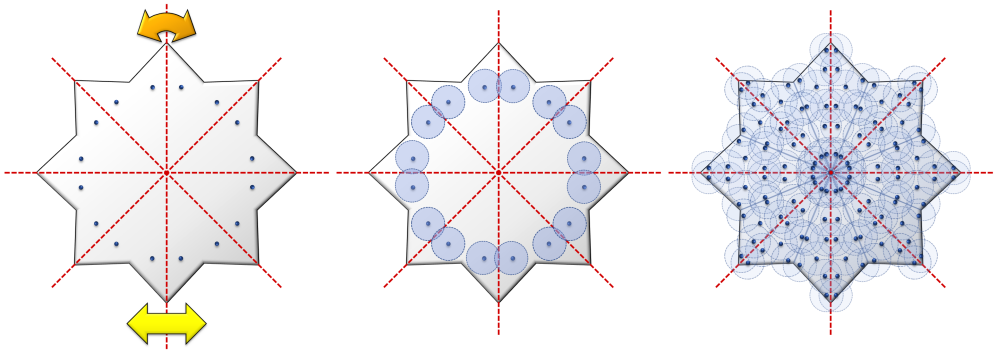
Symmetry-preserving displacements of the sampling Let us first assume that the shape has only one constant symmetry group: During the sampling, we collect the transformations that map every seed point to its whole orbit. These can be used to construct the space of symmetry-preserving displacements of the sampling. For this, we use the following fact: whenever a point p is transformed by a Euclidean motion $g(p) = O(p) + t$, a displacement u of the point is transformed only by the orthogonal matrix O. Hence, we obtain a symmetry-preserving displacement of the sampling by displacing one vertex and propagating the displacement to the orbit of the point using only the orthogonal parts O of the Euclidean motions g (Figure 3.7(a)). The orbit of any sample point p has exactly three degrees of freedom that we obtain by applying this procedure to the unit displacements of p into each of the three coordinate directions. To generate a basis of the space of symmetry-preserving displacements, we construct the three basis vectors for every seed point we placed during sampling.

Partial, overlapping, and nested symmetry The same construction also works for nested and overlapping symmetry groups, where the transitive closure of the orbits is considered (Figure 3.6). The sampling algorithm generates these points by following and concatenating the local transformations during sampling. Hence, the local frames O are given by concatenations of the orthogonal mappings involved.



(a) local frames (b) double frames (c) near-doublets

Figure 3.7: Local frames. (a) Each sample point is associated with a local frame O. (b) If a point lies within a transformation-invariant set, it can have more than one frame $O_1, O_2, \ldots$. (c) The problem can be ignored for points in general position as the contributions of the radially-symmetric basis functions cancel out and the low-pass kernel maintains the band-limitation.



(a) random sample (b) excluded discs (c) final sampling

Figure 3.8: Symmetric sampling. (a) We pick a random point and apply all group transformations to it. (b) An exclusion disc is centered at each point for futher sampling. (c) New points are sampled from non-excluded area until the object is fully covered.

Degenerate samples A special case occurs if a sampling point is visited more than once but with different local frames O_i. This can happen on transformation-invariant sets, such as the diagonals in Figure 3.7(b): Here, we have orbits with four points from eight transformations, and each point has two different frames, differing by a reflection. The correct solution is obtained by reducing the dimension of the basis to those vectors v for which $O_i v = O_j v$ for all i, j, which

yields is a linear system of equations. Due to the random sampling, this is rarely encountered in practice. In relevant cases, we can perform an SVD reduction of the null space to remove spurious degrees of freedom. If points do not perfectly overlap but only come close (which is still common close to transformation-invariant sets, see Figure 3.8(c)), we do not need to take special measures — the contributions of the basis functions cancel out exactly; we only obtain some overhead due to too dense sampling. The overhead is small as it only occurs at transformation-invariant sets of measure zero (reflection planes, rotation centers, Figure 3.7(c)).

Lifting the displacements To propagate a displacement of the sampling to a displacement of the surface, we compute for each basis vector $\bar{b}_i$ of the space of symmetry preserving displacements of the sampling a corresponding displacement vector b_i of the mesh. Then, the displacement $\bar{u} = \sum_i q_i \bar{b}_i$ of the sampling is lifted to the displacement $u = \sum_i q_i b_i$ of the mesh. The basis vectors $\bar{b}_i$ and b_i are a vector fields specifying a three dimensional vector for each sample point and mesh vertex. We denote these three dimensional vectors by $\bar{b}_i(\bar{v}_l)$ and $b_i(v_k)$. A displacement of a sampling point should only affects the displacement of the mesh vertices in a local neighborhood. We use Gaussian functions with standard deviation equal to the sampling density around every sample point to a assign influence weights to the mesh vertices. The Gaussian functions are cut-off (set to zero) for function values below 0.001. For each pair of a point $\bar{v}_l$ of the sampling and a vertex v_k of the mesh, we obtain a weight w_{kl}. Due to the compact support property this weight matrix is sparse. The basis vectors b_i are given by a partition-of-unity:

$$b_i(v_k) = \frac{1}{\sum_l w_{kl}} \sum_l w_{kl} \bar{b}_i(\bar{v}_l).$$

The basis b_i can be precomputed such that the Gaussians need not be evaluated in the interactive editing phase.

3.5 Symmetry-Preserving Editing

Once the subspace of symmetry-preserving displacements has been constructed, any deformation-based editing scheme could be used to produce symmetry-preserving deformations. Only the set of feasible displacements needs to be restricted to the subspace. However, as the meshes can be highly resolved, the computation of a deformation can be expensive. To be able to compute deformations of the surface in real-time, we restrict to low-frequency deformations

that are liftings of displacements of the sampling. To compute the displacements of the sampling, we use an iterative co-rotated Laplace editing inspired by the approach proposed in [Au et al., 2006]. The reasons for choosing this are approach are that on the one hand, we obtain a non-linear editing scheme that allows for large deformations, and, on the other hand, we only need minimal additional structure to compute the deformations. Namely, we need a Laplace matrix for the sampling and a list of neighbors for each vertex.

There are different ways to get a Laplace matrix for the sampling. One is to compute the Laplace matrix of the original mesh and to restrict this matrix to the subspace generated by the sampling, see [Huang et al., 2006]. A second way is to compute a point-cloud Laplacian, see [Liu et al., 2012], of the sampling. In our implementation, we specify a graph structure on the sampling: any pair of distinct points closer than $2r$ is connected by an edge. Then, we use the discrete Laplace (or Laplace–Kirchhoff) matrix of the graph, which is given as the difference of the degree matrix and the adjacency matrix. For Laplace editing, the Laplace matrix is applied not just to one function, but to the three component functions of the displacement vector field of sample points. Therefore, the $n \times n$ Laplace matrix is extended to a $3n \times 3n$ matrix by replacing every entry with the 3×3 matrix that is the entry times the 3×3 identity matrix. We denote this extended Laplace matrix by L.

Laplace editing The basis of the non-linear iterative co-rotated Laplace editing is the linear Laplace editing. For a thorough discussion of linear surface editing methods, we refer to [Botsch and Sorkine, 2008]. Here we briefly review the variant of Laplace editing our scheme is based on. The deformation is computed by solving a quadratic minimization problem. The objective functional combines two quadratic functionals: one measures the deviation of the so-called Laplace coordinate and the other measures the deviation (in a least-squares sense) from user-specified constraints. We denote by $\bar{x}$ the vector listing the coordinates of the sample points and by $\bar{u}$ the displacement of the sampling points. We use the bar to distinguish the coordinates and displacements of the sampling from those of the surface mesh. The vector of Laplace coordinates is $\delta = L\bar{x}$ and the first quadratic functional is

$$E_L(\bar{u}) = \|L(\bar{x} + \bar{u}) - \delta\|^2 .$$

In our implementation the user can select handle regions in the sampling and assign desired positions to the selected sample points by rotating and translating the handles in space. The deformed sampling will approximate these constraints. The corresponding least-squares functional is

$$E_C(\bar{u}) = \|A(\bar{u}) - a\|^2 ,$$

where a lists the desired displacements of all vertices in the handle regions. The matrix A is rectangular and has only one non-zero entry per row, which takes the value 1. The resulting deformation is given by the displacement that minimizes

$$E(\bar{u}) = E_L(\bar{u}) + \alpha E_C(\bar{u}) \tag{3.3}$$

among all symmetry-preserving displacements. The parameter $\alpha \in \mathbb{R}_+$ controls how strongly the surface is pulled towards the user-specified handle positions.

To solve the quadratic program, we use the *null-space method*; see [Nocedal and Wright, 2006], Chapter 16.2. Let U be the rectangular matrix whose columns are the basis vectors of the space of symmetry-preserving displacements of the sampling, and let q be the vector of coordinates with respect to the basis. To compute the minimizer of (3.3) in the space spanned by U, we have to solve the linear system

$$U^T(L^T L + \alpha A^T A)Uq = U^T(\alpha A^T a + L^T(\delta - L\bar{x})) \tag{3.4}$$

for q. Then, Uq is the solution in the space of symmetry-preserving displacements of the sampling. Since the symmetric, positive definite matrix $U^T(L^T L + \alpha A^T A)U$ only changes when new handle regions are selected or the weight α is modified, it is efficient to compute a Cholesky factorization of this matrix and to re-use it for solving the minimization problems. In addition, using a factorization speeds up the iterative co-rotated Laplace editing.

Iterative co-rotated Laplace editing A limitation of Laplace editing is that deformations that include larger rotational components lead to visually observable artifacts in the deformed surface, see [Botsch and Sorkine, 2008].

Therefore, we use a non-linear variant obtained by iteratively applying Laplace editing. The Laplace coordinate can be interpreted as a vector field on the sampling specifying a 3-dimensional vector δ_i for every vertex. The Laplace coordinate is related to the discrete mean curvature normal and every δ_i should point into the direction of the surface normal at the corresponding sample point. The co-rotated Laplace editing process iterates a two-step procedure. First, the linear system (3.4) is solved. Then, the Laplace coordinates are rotated to point in normal direction of the deformed sampling. For rotating the Laplace coordinate of each vertex, we consider the undeformed and deformed local neighbors of the sample point and compute the rotation such that the rotated undeformed neighborhood best matches the deformed neighborhood. The neighborhoods are defined by the graph structure on the sample points. The rotation matrix can be computed using SVD, as described in (the additional

material of) [Sorkine and Alexa, 2007]. We stop the process when the maximum number of iterations (usually 5-10) has been reached. Once a displacement of the sampling is computed, we use the lifting process described in Section 3.3 to propagate the displacement of the sampling to a deformation of the surface mesh.

3.6 Experiments and Discussion

We tested our implementation of the proposed method on a set of shapes with varying complexity. Some shape have only a few symmetries, other have more complex nested symmetries. The mesh sizes vary from 4k to almost 200k vertices. Details are listed in Table 3.1. We tested the method with automatically detected and with manually specified symmetries. For example, the 61 symmetries of the car model (Figure 3.11) have been detected using the method proposed in [Tevs et al., 2014b] and the 9 symmetries of the center piece (Figure 3.1) have be specified manually. For every example, the symmetries are illustrated in the corresponding figure. We tested with various sampling densities resulting in 72 to 690 sample points. We used random sampling and feature-sensitive sampling, where sharp features are selected first. We found the random sampling to produce the same quality of results as the feature-sensitive approach. Still, we show one example (the bar, Figure 3.5) produced with feature-sensitve sampling (here, we use the SVD-reduction described in Section 5 to remove doublet frames; this is not required/performed elsewhere). The dimensions of the spaces of symmetry-preserving displacements vary from 18 to 234. For many examples, we show large deformations, which the iterative co-rotated Laplace editing nicely supports.

Timings Our implementation of the symmetry-preserving modeling runs at 13-60 fps in our experiments. Table 3.1 lists details for the shown examples, including timings for pre-factoring the Laplace matrix, 5 iterations of the co-rotated Laplace editing, lifting the deformation from the sampling to the surface mesh, and rendering. The timings were generated using a single-threaded C++ implementation running on an Intel® Xeon® E5-1620 processor at 3.60GHz with 16GB of RAM. The timings could be improved using parallelization, in particular, the time required for lifting the displacement from the sampling to the surface mesh. The shown run times demonstrate that, except for lifting the solution from the sampling to the surface and rendering the surface, the time needed to compute a deformation is independent of the resolution of the surface. It mainly depends on the dimension of the subspace of deformations, which in

turn depends on the symmetry structure of the model and the resolution level of the desired editing operations. For example, though the car and the yard tool examples have a comparable number of vertices, there is a large difference in the time needed to compute the deformations, which is due to different sampling densities.

Comparison We compare our method to the recent scheme of Kurz et al. [2014]. They propose to use symmetry-preserving deformation for template fitting to incomplete scan data. A comparison of results is shown in 3.12. A difference between the schemes is that their method enforces the symmetry in a least-squares sense, whereas our approach exactly preserves the symmetry. The resulting difference in visual quality can be seen in the figure. For example, our method better preserves the symmetry of the blades of the fan and better fits the desired positions of the legs of the armchair. To obtain the deformation that matches the given incomplete data, they combine handle-guided deformation and ICP. Let us compare the timings for the fan example. The timings listed in [Kurz et al., 2014] are 2 seconds and 2 minutes per iteration of the optimization, depending on whether linear or smooth basis functions (with wider support) are used. Their main bottleneck is the setup of the transfinite least-squares-constraints that involve numerical integration and fine discretization (disabling them reduces the computation times in their approach from 2 seconds to 0.4 seconds, still including ICP). Our results were produced using only handles. After 43ms for pre-factoring the Laplace matrix, the time per iteration of our optimization is less than 1ms. While the numerical implementation of Kurz et al. arguably leaves room for improvement, a performance gain of at least three orders of magnitude indicates clear advantages of our approach.

In the following, we discuss relations and differences of the proposed approach to the other previous work. Similar to our approach, the " iWires" framework [Gal et al., 2009] aims at preserving geometric relations of a shape to be edited. Our symmetry-preservation model (based on [Kurz et al., 2014]) captures most of their constraints (equal area, equal length, etc.). In addition to conceptual simplicity, this also provides a navigable shape space that can be utilized as prior in various applications. Some details are different: For example, our method does not explicitly parallel lines. However, as linear constraints, they could be added as special cases to our approach, too. Further, unlike our approach, we use dense relations on symmetric area rather than feature lines ("wires"), thereby no relying on the existence of such sharp features. As solver, iWires uses a greedy propagation algorithm that, unlike our method, does not guarantee to satisfy all constraints. Nonetheless, our method is still faster (Gal et al. report response times of 2-4 seconds).

Wang et al. [2011] use hierarchical propagation of attributes in objects with complex symmetry patterns for structure-preserving shape editing. The work differs from ours in that it builds a hierarchical representation, breaking cycles in symmetry-constraints by perceptual reasoning. In contrast, we utilize all symmetry information, including arbitrary cycles. By understanding the feasible set as a linear shape space, efficient navigation can still be guaranteed (whereas the method of Wang et al. utilizes direct edit propagation in the hierarchy).

3.7 Summary

In this chapter, we specifically consider structure-aware deformations that preserve symmetries of the shape being edited. While this is an elegant approach for obtaining plausible shape variations from minimal assumptions, a straightforward optimization is numerically expensive and poorly conditioned. Our work introduces an convenient method to explicitly construct linear spaces bases of shape deformations, which exactly preserve symmetries for any user-defined level of detail. This permits the construction of a low-dimensional spaces, which allow only low-frequency deformations that preserve the symmetries. We obtain substantial speed-ups over alternative approaches for symmetry-preserving shape editing due to (i) the sub-space approach, which permits low-res editing, (ii) the removal of redundant, symmetric information, and (iii) the simplification of the numerical formulation due to hard-coded symmetry preservation. We demonstrate the utility in practice by applying our framework to symmetry-preserving co-rotated iterative Laplace surface editing, which can easily handle models with complex symmetry structure, including partial and nested symmetry.

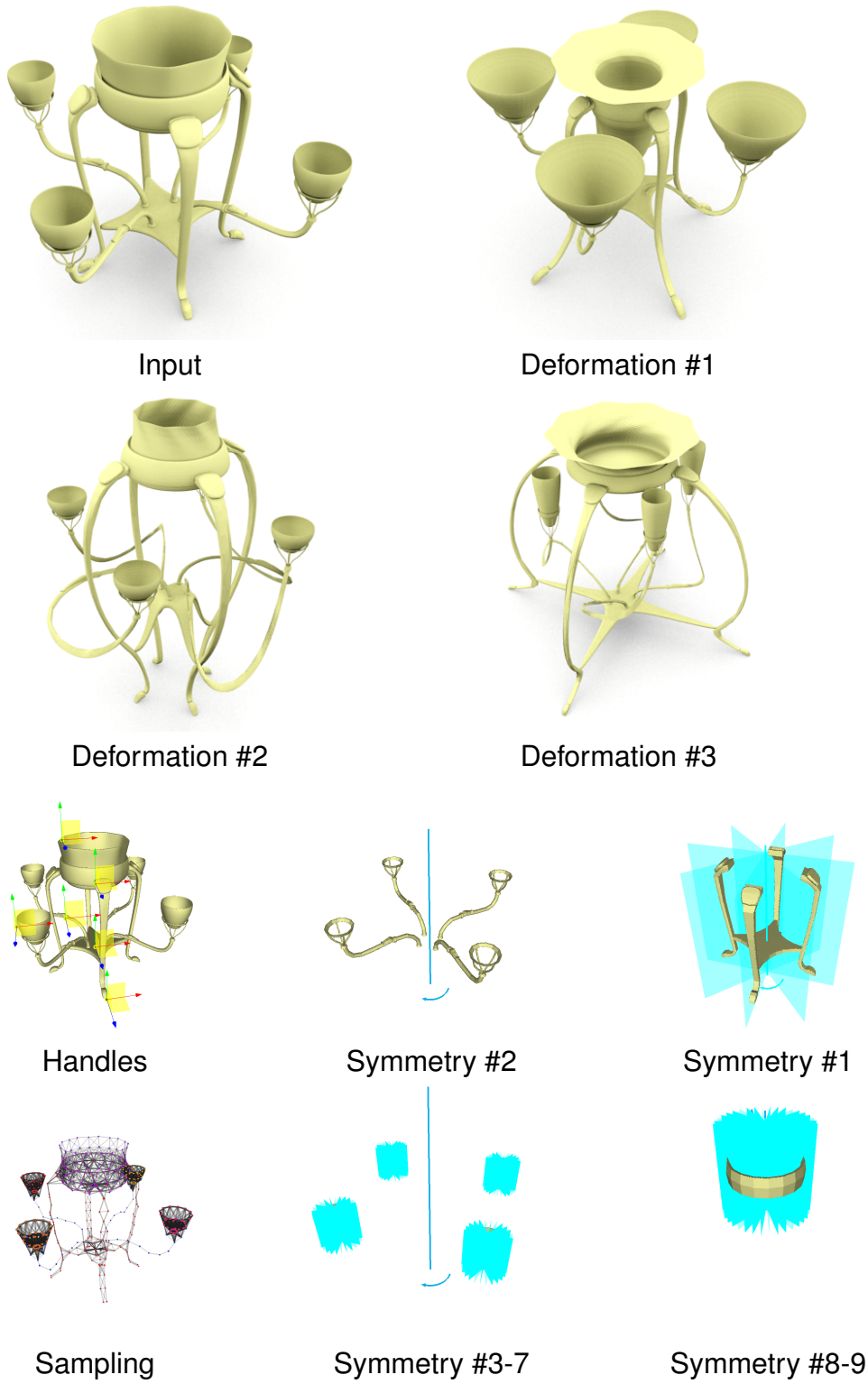


Figure 3.1: Shapes generated with our symmetry-preserving modeling system are shown. These example show a combination of complex symmetries and large deformations. The symmetries are illustrated in the bottom row.

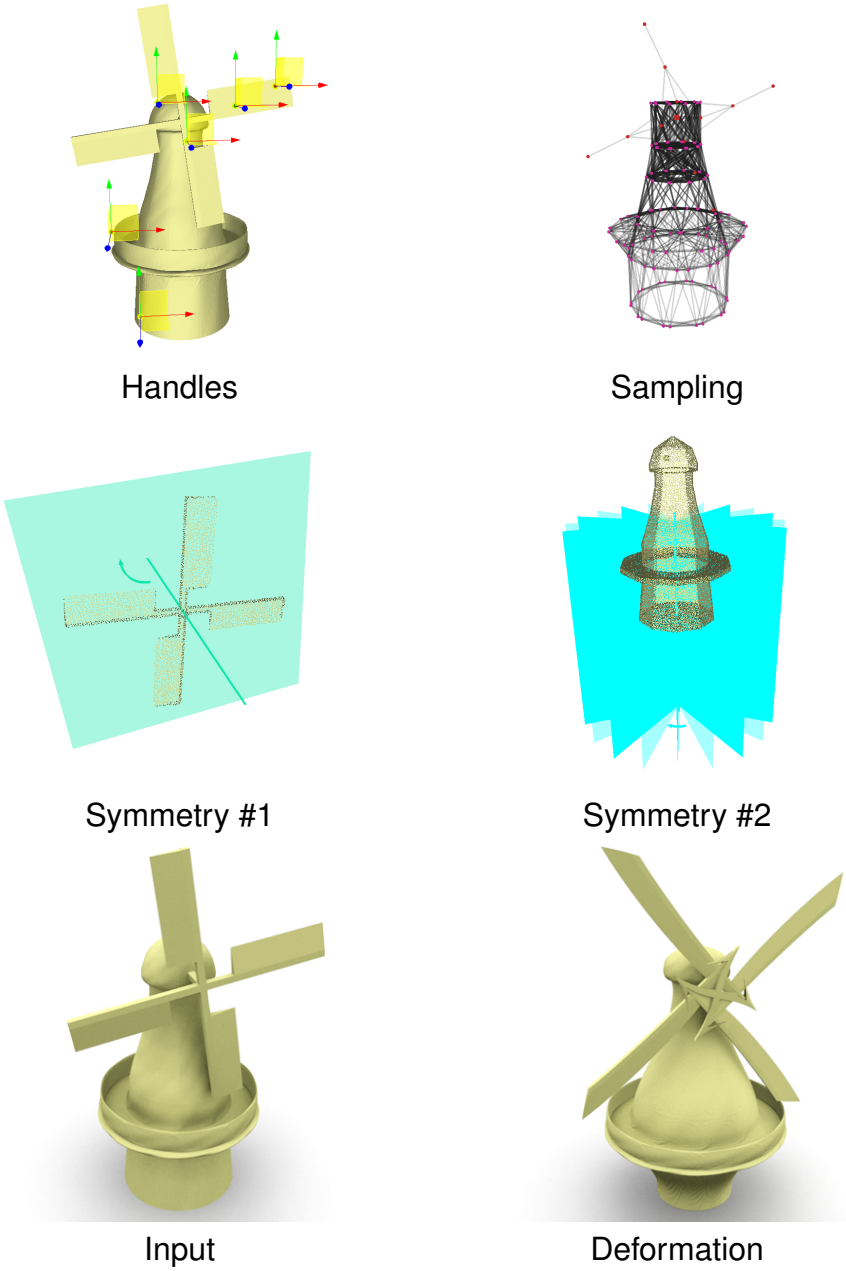


Figure 3.2: A deformation of a wind mill model generated with out symmetry-preserving modeling system is shown.

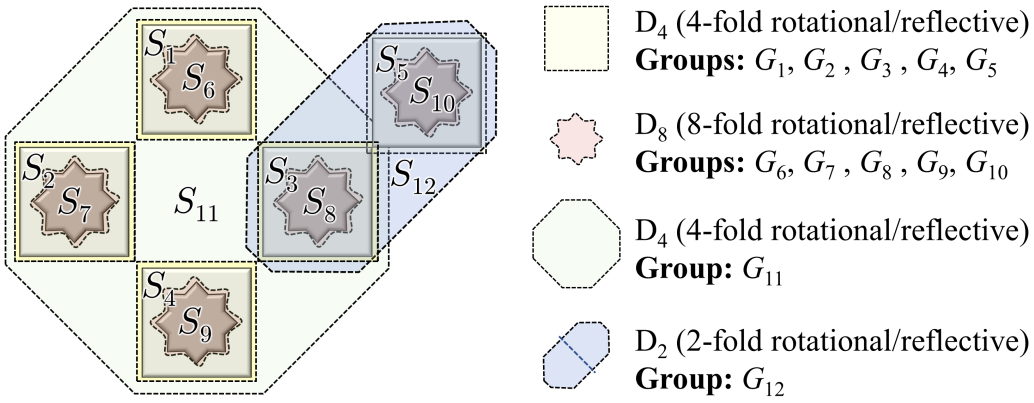


Figure 3.3: Nested and overlapping symmetries. Regions with different symmetries overlap. For example, region S_1 is four-fold symmetric, and the subset S_6 is 8-fold symmetric. The 2-fold symmetry of S_{12} overlaps partially with S_{11}.

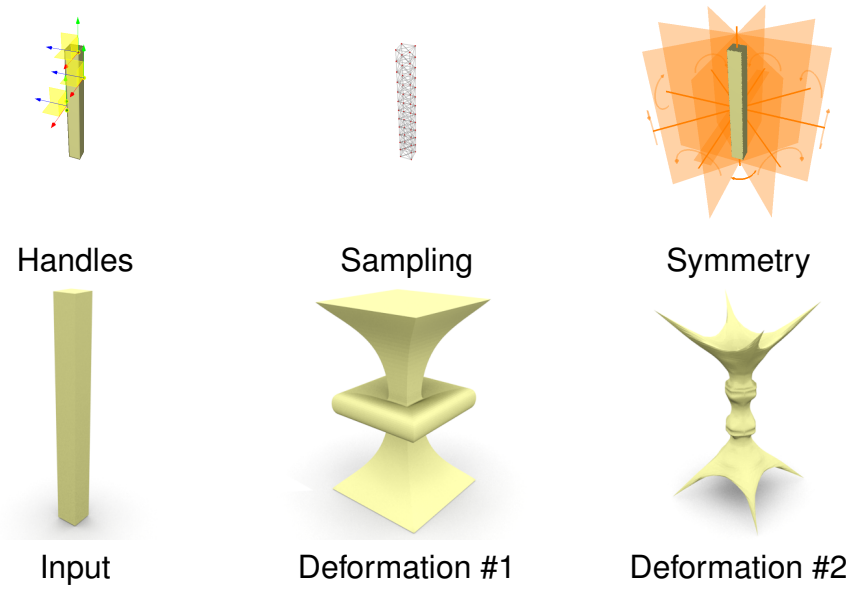


Figure 3.5: Symmetry-preserving deformations of a bar are shown. The deformation are generated using three handles.

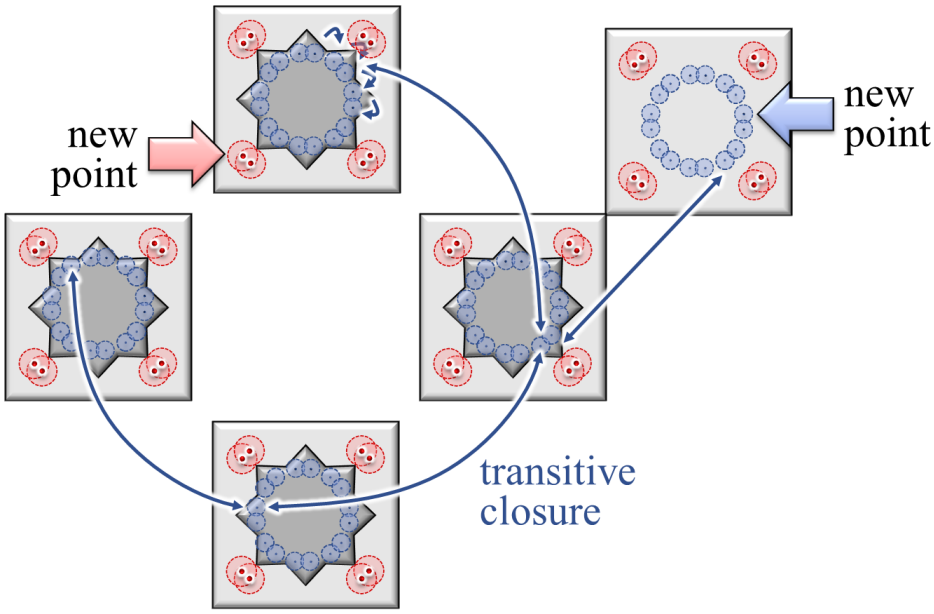


Figure 3.6: Nested and overlapping symmetries are treated by propagating samples along transformations.

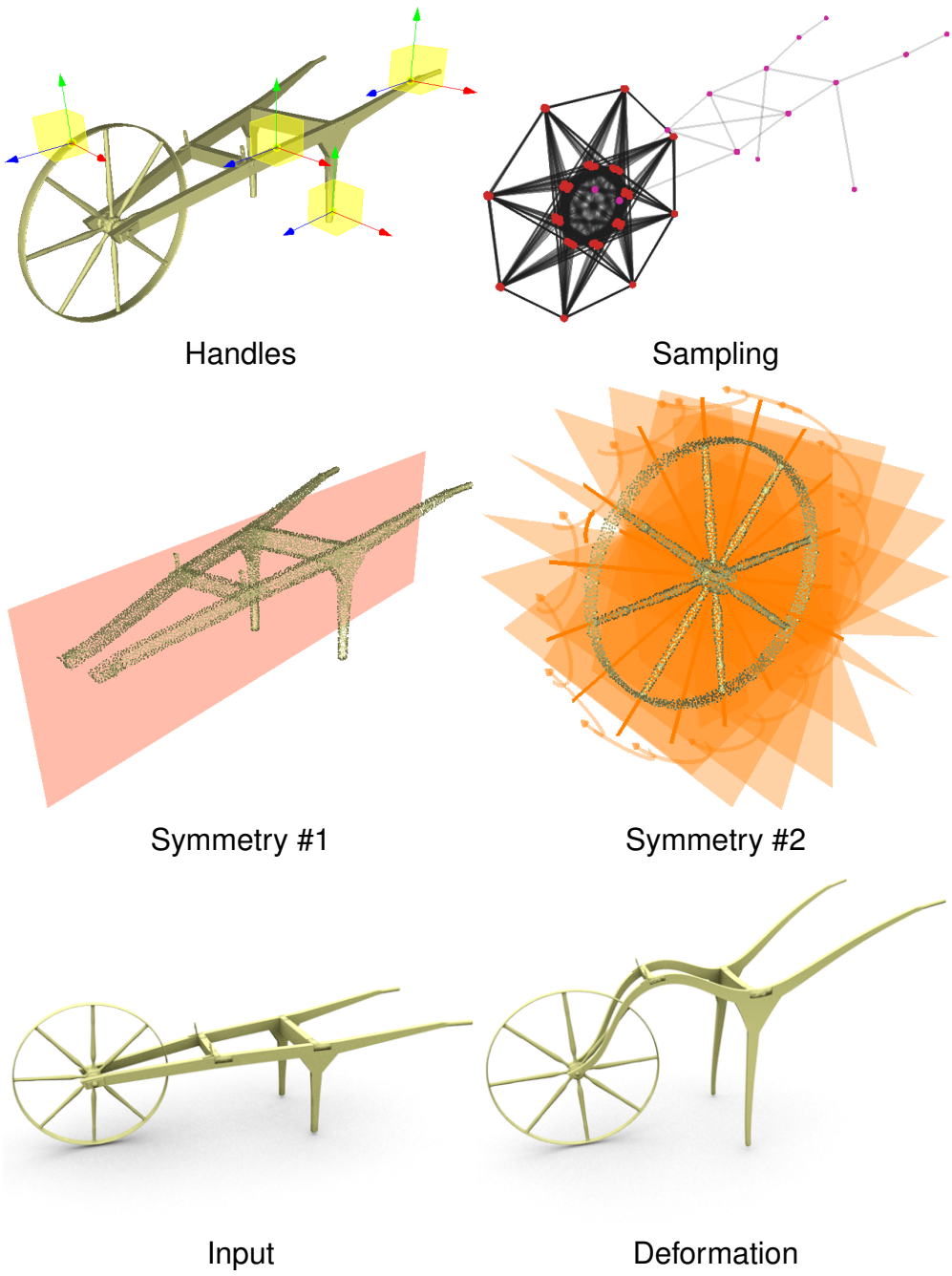


Figure 3.9: A larger deformation of a yard tool generated with our symmetry-preserving modeling system is shown.

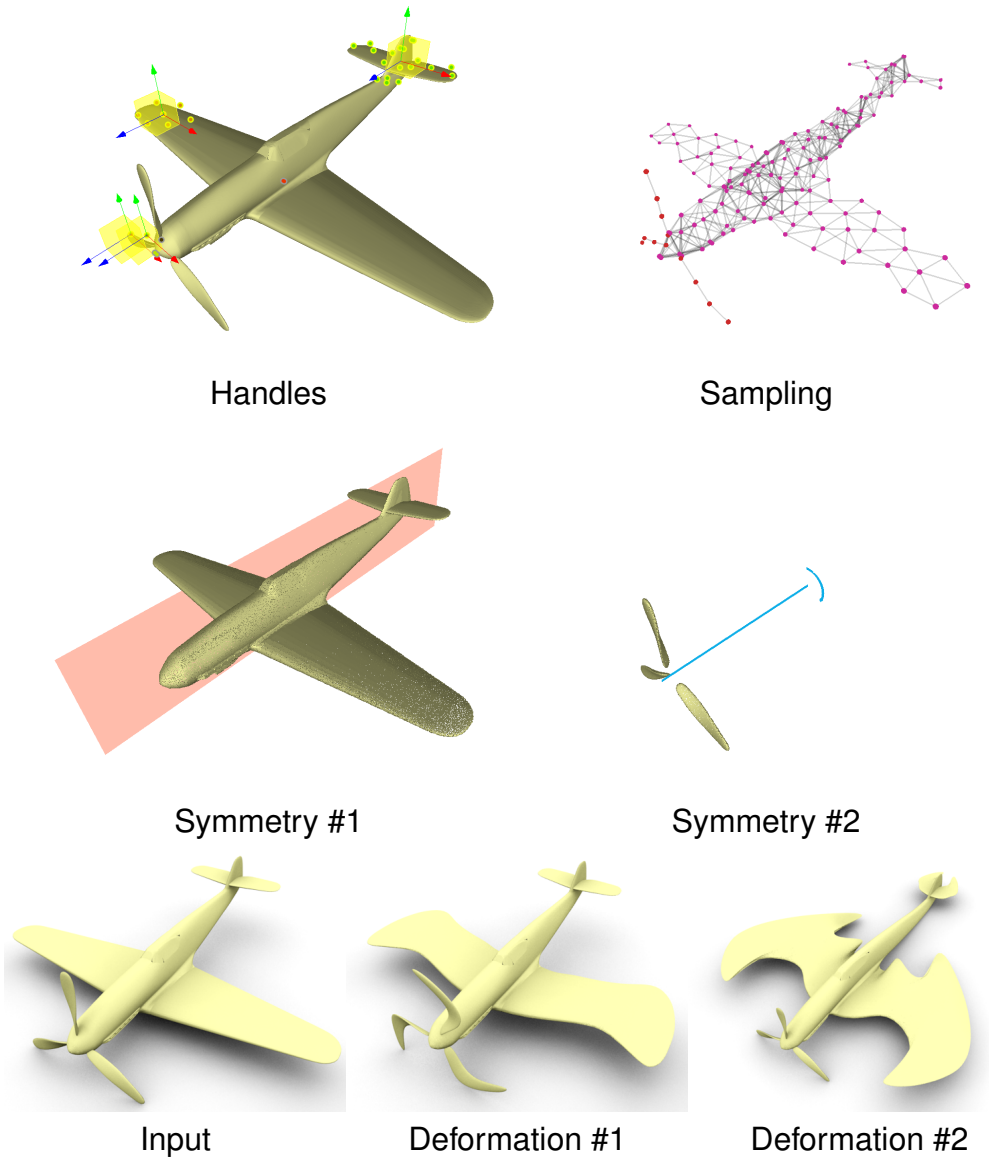


Figure 3.10: Deformation generated with our symmetry-preserving modeling system are shown.

Model		Mesh Info.		Sampling Info.			Timing (ms)				Frame rate
Name	Figure	Vertex (k)	Triangles (k)	Groups	Samples	DoF	Prefactor	Solve (5 Iter.)	Lift	Render	Frames per s
Bar	3.5	4	8	1	96	18	13	2	3	12	58.8
Center piece	3.1	83	165	9	690	150	1300	21	32	12	15.4
Plane	3.10	181	359	2	160	234	75	4	51	18	13.7
Yard tool	3.9	50	97	2	78	27	55	3	22	5	33.3
Car	3.11	47	95	61	675	156	4160	35	3	10	20.8
Wind mill	3.2	20	40	2	156	33	97	5	7	6	55.6
Fan	3.12	2	4	3	72	27	43	3	1	13	58.8
Chair	3.12	15	58	2	32	39	4	2	1	16	52.6

Table 3.1: Statistics and timings of the shown examples. From left to right: number of vertices, number of triangles, number of symmetry groups, number of sampling points, dimension of the space of symmetry-preserving deformations, timings for pre-factoring the Laplace matrix, five iterations of co-rotated Laplace editing, lifting the solution from the samples to the surface mesh, and rendering the mesh, and the total number of frames per second.

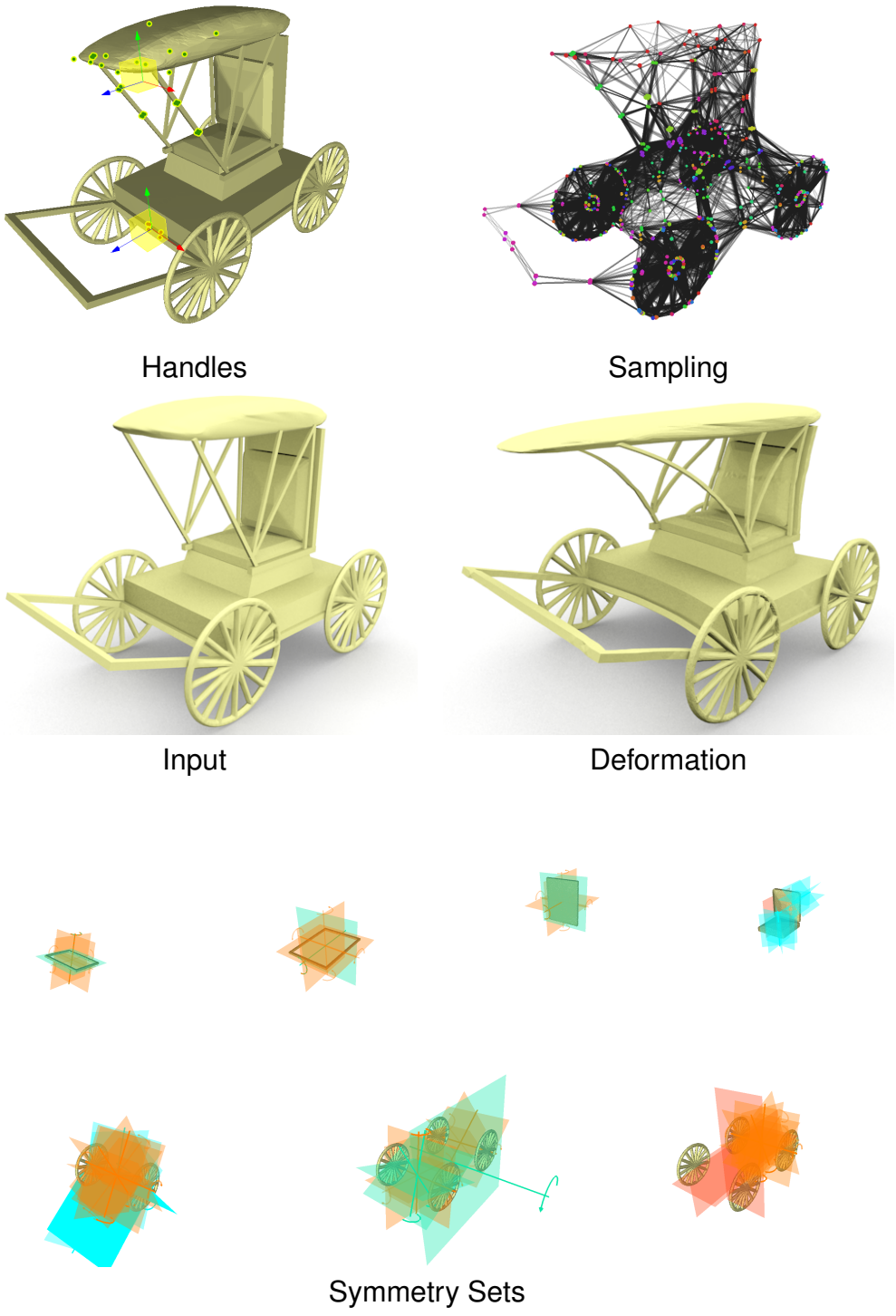


Figure 3.11: A deformation that preserves 61 symmetries that were automatically detected is shown. Despite of the complex symmetry structures, the modeling system runs at real-time. Some of the detected symmetries are shown in the bottom row.

Figure 3.12: We compare results of our method with results produced by the scheme proposed by Kurz et al. [2014].

Chapter 4

3D Model Retargeting Using Offset Statistics

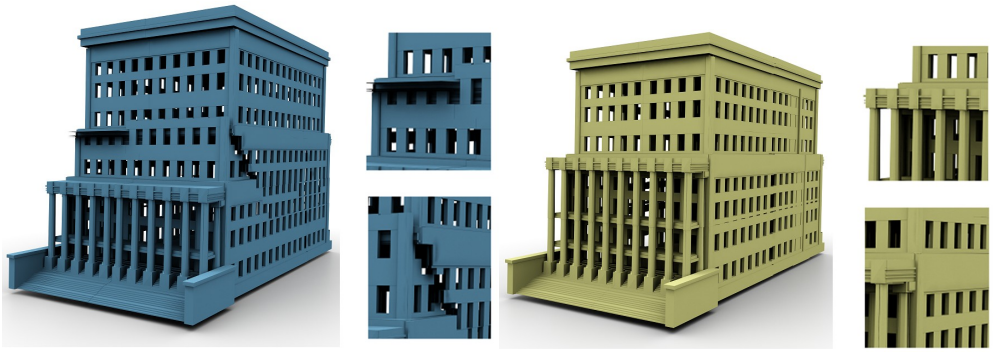
In this chapter we consider discrete translational symmetry, so statistics along each translational line can be easily summarized. Through the study of offset statistics, we have observed sparse distribution, and used this property to simplify retargeting process. We also conducted empirical experiments to verify our sparsity hypothesis, which further serves as a guidance to our algorithm.

4.1 Introduction

As introduced in Chapter 2, retargeting is a very practical method for creating 2D contents. Texture synthesis is one of such applications, which is a versatile example-based generative model for creating and editing 2D images. However, generalize its principles to 3D is difficult, due to the higher demand of model accuracy, and the larger search space that also contains many implausible shapes.

Synthesizing shapes from examples involves two key challenges: First, we have to characterize the shape space of plausible variants of the input data. Second, we also need an efficient algorithm to explore that shape space.

A necessary requirement for plausibility is local consistency: Shapes should form closed surfaces that resemble the input data at a local scale. This notion is captured by Markovian texture synthesis methods [Wei et al., 2009]. However, local consistency is not sufficient to characterize meaningful shapes (figure



(a) 4992 nodes, 64 labels, 8.035 sec. (b) 378 nodes, 8 labels, 0.195 sec.

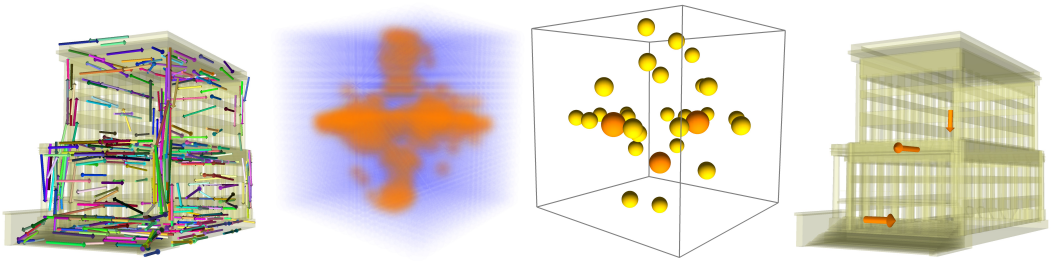
Figure 4.1: 3D retargeting without (a) and with (b) offset statistics. Close inspections are shown on the right. The complexity of the the graph-cut problem and the corresponding optimization time are reported for each result.

4.1-a); synthesizing complex models (beyond "texture" with stationary statistics) require additional structure to be maintained.

In this chapter we propose a simple but effective method to retarget 3D models using the statistics of low-level translational symmetries. Our method is inspired by the recent success of offset statistics in image in-painting [He and Sun, 2012] and editing [Pritch et al., 2009]. Intuitively, it uses an MRF model to stitch translational copies of the input model, where the optimal stitching can be found using graph-cut based optimization. However, the method only deals with those shapes that are aligned with salient, frequently observed offsets between matching features, instead of arbitrary translations. This additional regularization helps enforcing a more plausible global structure.

The key insight of our work is the sparsity of offset statistics, which reveals important structural redundancy, and can be used for generating good retargeted shapes. Frequently appearing offsets represent dominate repetitive structures in the model, and also indicate promising directions and spacings for retargeting the model. To utilize this information, we set up a transformation space that is the linear span of the dominant offset vectors, then use transformations inside this space to translate the input model. In this way, the structural regularities within the input model influence the distribution of the translational copies, so are more likely to be preserved in the retargeted model (figure 4.1-b). In the meantime, since the number of dominant offsets is relatively small (also due to the sparsity), retargeting is significantly more efficient (c.f. figure 4.1).

A major advantage of our method is being low-level, which made our algorithm simple and robust. In contrast, a lot of recent works use much stronger regu-



(a) Feature Matching (b) 3D histogram (c) Histogram peaks (d) Principle offsets

Figure 4.2: Our analysis step a) automatically matches features, b-c) find dominates offsets, and d) extracts up to three principle offset vectors.

larities, such as explicit algebraic models [Bokeloh et al., 2012b] or symmetry hierarchies [Wang et al., 2011; Jain et al., 2012]. However, these methods require either clean data with exact regularity, or human assistance for pre-segmentation [Lin et al., 2011b; Zheng et al., 2013], which limit their applicability in practice. While our method is only limited by a simpler structure representation, texture synthesis can be easily accommodated for imperfect regularity and other data imperfections, thereby completing previous methods.

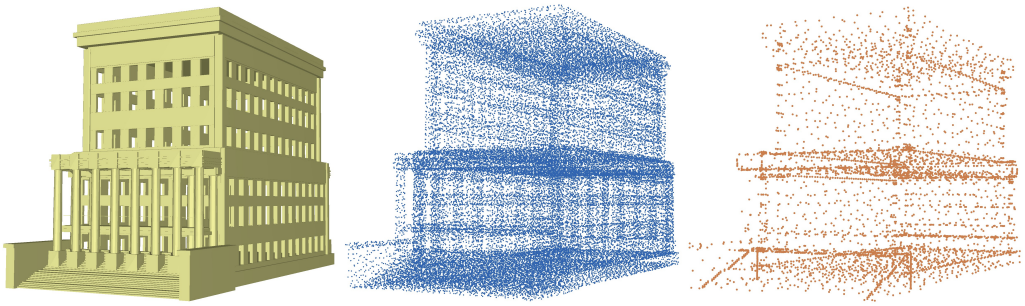
In summary, we made the following main contributions:

- An empirical study that confirms the sparsity of offset statistics in 3D man-made shapes.
- An algorithm that automatically detects dominant offsets from single input models.
- A 3D texture synthesis algorithm that utilizes offset statistics for automatic, high quality model retargeting at interactive speed.

The rest of this chapter is organized as following: section 4.2 provides an overview of our method. In Section 4.3 we give empirical evaluation for the sparsity of 3D offset statistics, which is the theoretical foundation of our method. In Section 4.4 we explain implementation details and in Section 4.5 we demonstrate our results.

4.2 Method

Our method has two main steps: Offset statistics detection and model retargeting (see figure 4.2 and 4.4).



(a) Input model (b) Dense samples (c) Features

Figure 4.3: Sampling from input model.

4.2.1 Offset Statistics Detection

In this step we compute offset statistics from the input triangle mesh. To calculate features and their descriptors, we first perform Poisson disc sampling at two different scales (figure 4.3). Low density samples are stored as features, and the high density samples are used for computing feature descriptors. We also include all mesh vertexes as features to improve the matching. For each feature $\mathbf{x}$, we compute its SHOT [Tombari et al., 2010] descriptor, denoted by $S(\mathbf{x})$.

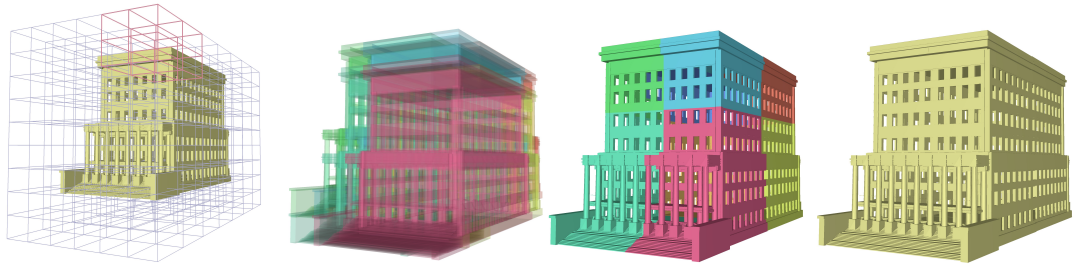
Next, we match similar features in the input model based on their descriptors. For each feature $\mathbf{x}$, we compute the offset vector $\mathbf{v}$ as the distance to its best match:

$$\mathbf{v} = \arg\min_{\mathbf{v}} |S(\mathbf{x}+\mathbf{v}) - S(\mathbf{x})| \quad s.t \;\; |\mathbf{v}| > \tau > 0 \tag{4.1}$$

Here $\mathbf{v} = (u, v, w)$ is the translation vector from feature $\mathbf{x}$ to its closet match. Figure 4.2-a illustrates some example mappings (only 10% of the them are shown for clarity). The similarity between two SHOT feature descriptors is measured by the L2 norm $|\cdot|$. The threshold τ (set to 0.15 in our experiments) avoids matchings between features that are too close. As noted by [He and Sun, 2012], such a minimum distance constraint is useful for avoiding trivial statistics.

Next, we compute offset statistics from the matched features. To do so we bin all offsets into a 3D histogram, as shown in figure 4.2-b. Bins with strong magnitude are rendered in warmer color. We then detect peaks in the histogram using standard non-maximum suppression (figure 4.2-c) with a radius of 0.15. The detected peaks (yellow) are the dominate offsets.

In practice we usually have about 20 dominate offsets detected from a model. Although it is possible to keep all of them for later retargeting, this is not necessary as many of them could be formed from combination of very few gener-



(a) Space discretization (b) Offset montage (c) Graph cut (d) Retargeting result

Figure 4.4: Offset based retargeting.

ator vectors, which we call *principle offsets*. (orange spheres / arrows in figure 4.2-c, 4.2-d). Principle offsets span a regular grid of vectors (a translational k-parameter group [Pauly et al., 2008b; Bokeloh et al., 2012b]), which provides candidate offsets for model stitching. As our method aims at retargeting (i. e., resizing in principal directions), we keep up to three principle offsets for a single model (which is also the maximum number of independent generators of a single 3D lattice [Pauly et al., 2008b]). In consequence, this approach more or less guarantees correct results if a 3D model actually contains a dominant grid pattern (figure 4). However, graph-cut-based stitching, as a global optimization method, still can find plausible solutions for models of much more complex structure (such as the top three examples in figure 9).

In the next section we continue with the core retargeting algorithm, assuming that the principle offsets are known. In Section 4.4 we provide details about how to extract the principle offsets.

4.2.2 Model Retargeting

The basic idea is to retarget models by stitching together translational copies of the input model. Intuitively, one can discretize the retargeting space into regular voxels. Each voxel can be assigned with a label, which indicates the source geometry that used for copying into that voxel. The assignment of labels can be solved using the standard multi-label graph-cut-based optimization [Boykov et al., 2001a]. However, stitching shapes together is non-trivial in practice, because graph-cut-based geometry stitching can produce implausible shapes, when the transformation that used to combine two copies is not chosen carefully.

In our context, it is impossible to stitch shapes into a good model if the boundary of translational copies are not aligned with grid border. In theory, one can use larger number of copies to increase the chance of getting a good synthesized

model (such strategy would also involve a yet to be defined criterion for rejecting bad offsets). However, this is not practical as the optimization cost quickly increases with the resolution of discretization (number of nodes and labels in the graph). In the meantime, even with a very fine discretization, graph-cut geometry synthesis can still produce invalid shapes if the structure of the input model is ignored.

Our key insight is to use offset statistics to set up a proper transformation space that preserves the structure of the input model. To do so, we define the *offset space* Φ as the span of pre-detected principle offset vectors:

$$\Phi = \left\{ \sum_{i=1}^{K} \lambda_i \mathbf{v}_i | \lambda_i \in \mathbb{Z} \right\} \tag{4.2}$$

As previously discussed, the principle offsets $\{\mathbf{v}_i\}_{i=1}^{K}$ are generators for resizing a model in different canonical directions. $\mathbb{Z}$ in Eq. 4.2 is the set of integer numbers, so λ_i is the number of steps to resize a model using $\mathbf{v}_i$. We restrict λ to be integer values for discrete optimization. In this work we use three principle offsets ($K = 3$) so the offset space contains $M = U \times V \times W$ translational copies of the input model, where $\{U, V, W\}$ are the steps for different principle offsets. Each copy has a unique offset label $\{l_i\}_{i=1}^{M} = (u_i, v_i, w_i)$, indicating the translational mapping between the copy and the original model, represented as number of steps that principle offsets to be applied. For example, $l = (1, 0, 0)$ indicates the copy is shifted by one step in the first principle direction. It is clear that M is also the number of labels for the later multi-label graph-cut optimization.

We now define *retargeting space* Ω as the volume covering synthesized model. We discretize Ω into a regular subdivided volume, once again, using the principle offsets as the generators so it is consistent with the offset space Φ. Specifically, the retargeting space has $\Omega = (U_{in} + U) \times (V_{in} + V) \times (W_{in} + W)$ voxels, where U_{in}, V_{in}, W_{in} are the respective dimensions of the input model (in units of principle offsets). Figure 4.4-a shows an input model, imposed by a $2 \times 2 \times 2$ offset space (red) and the resulting retargeting space (gray). Figure 4.4-b shows a montage of all translational copies rendered with different colors. These copies collectively fill up the retargeting space.

To stitch all copies together, we assign each voxel in Ω an offset label: $L(\mathbf{x}) = l_i$. Intuitively, it means that voxel $\mathbf{x}$ in the synthesized model has its geometry copied from voxel $\mathbf{x} - (v_i, u_i, w_i)$ in the input model. We find the optimal labeling assignment by minimizing the following MRF energy function:

$$E(L) = \sum_{\mathbf{x} \in \Omega} E_u(L(\mathbf{x})) + \sum_{(\mathbf{x}, \mathbf{x}') | \mathbf{x}, \mathbf{x}' \in \Omega} E_p(L(\mathbf{x}), L(\mathbf{x}')) \tag{4.3}$$

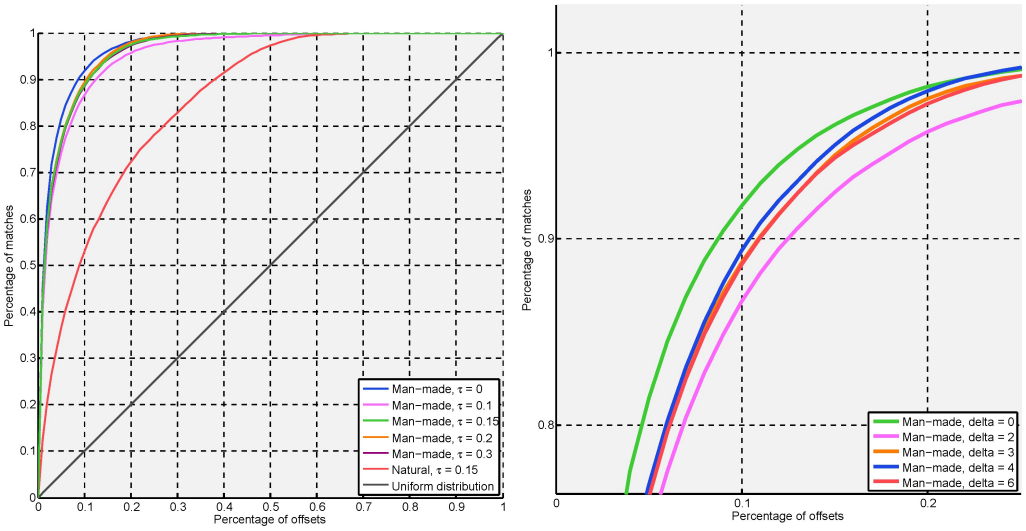
The unary term E_u is 0 if the label is valid (that is, $\mathbf{x} - (v_i, u_i, w_i)$ is a known voxel in the input model). Otherwise the term will receive infinite cost. The pairwise term E_p encourage smoothness between adjacent voxels in the 26-connected neighborhood. For each pair of neighboring voxels $(\mathbf{x}, \mathbf{x}')$ with label assignment $L(\mathbf{x}) = l_i$ and $L(\mathbf{x}') = l_j$, we define E_p as:

$$E_s(l_i, l_j) = d_H(G(\mathbf{x} + l_i), G(\mathbf{x} + l_j)) + d_H(G(\mathbf{x}' + l_i), G(\mathbf{x}' + l_j)) \tag{4.4}$$

This energy function penalizes stitching together voxels that have different geometries. Here $G(\mathbf{x})$ is the shape descriptor for voxel $\mathbf{x}$. We simply use the set of SHOT descriptors contained in the voxel. Then similarity between two voxels are computed as the Hausdorff distance d_H between those two sets, while similarity between individual SHOT features is measured using the L_2 norm. Notice the pairwise feature distances have already been computed in the previous step (Section 4.2.1) and can be reused here. In practice the target volume often contains empty voxels. In this case we assign zero distance if both voxels are empty, or infinite distance if only one of them is empty. It is easy to see that E_s is sub-modular, so we can optimize Equation 4.3 using the alpha-expansion algorithm [Boykov et al., 2001a], which produces optimal label assignments for the retargeting space N. Figure 4.4-c, d show the synthesized model output by the graph-cut optimization, where colors in figure 4.4-c indicate the offset labels assigned to different parts of the model. In Section 4.5 we will show more results and compare our method to an alternative method.

4.3 Sparsity of 3D Offset Statistics

The sparsity of offset statistics is the theoretical foundation for our method. In this section, we give some empirical evidence for the validity of this assumption using a study similar to [He and Sun, 2012]. We collect two datasets of 3D shapes, one for man-made objects and the other for natural objects. The man-made dataset (figure 4.5-c, top row) contains 200 models, including architectures, furniture, vehicle etc. The natural dataset (figure 4.5-c, bottom row) contains 120 models, such as human, animals and foliage. All models are pre-normalized and centered so the maximal dimension of each model is scaled to $[-1, 1]$. As previously described (figure 4.3), a dense point cloud is sampled for each model using Poisson sampling with 0.02 as the sampling space, from which a feature point cloud is re-sampled using 0.06 as the sampling space. Then 352 dimensional SHOT descriptor is computed for each feature. We use a fairly large radius (set to 0.2 in our experiment) to encode sufficient geometric information in the SHOT descriptors [Tombari et al., 2010]. For each feature, we find its closest match as



(a) Cumulative distribution (b) Zoom-in

(c) Datasets: man-made (top) and natural (bottom) subjects

Figure 4.5: Offset sparsity.

defined in Equation 4.1. We then bin the offsets between matched features and sort the bins in descending order of their magnitude (number of offsets in a bin). We use bin of size 0.05 in each dimension, and test different minimum distance thresholds τ (figure 4.5-a, b) for precluding trivial infinitesimal similarity case.

In order to examine the sparsity of offset statistics, we accumulate the bins and plot the cumulative distribution (averaged over all models) in figure 4.5-a. The diagonal line (gray) is the cumulative distribution for a uniform statistics,

indicating the most non-sparse distribution. In comparison, man-made objects have a clearly sparse distribution as the curve is highly non-linear. For example, when $\tau = 0.15$ (green curve), 79.9% of offsets are distributed in only 7% of bins, 97.1% of offsets are distributed in 20% of bins. We also observe that τ has little influence on the distribution – as the curves for different τ are very similar. This can be better seen in the enlarged plot (figure 4.5-b). Notice although the curve of $\tau = 0$ (blue) is relatively more distanced to the others, the difference is not as distinct as being observed in 2D image case [He and Sun, 2012]. This is because 3D models have higher degree of accuracy, thus are less likely to have false positive matchings. Nonetheless in practice we still use $\tau = 0.15$ to eliminate the possibility of getting trivial peaks around $(0,0,0)$ in the 3D histogram. Figure 4.2 shows an exemplar model where three principle offsets are detected. Figure 4.6 shows cases where only one or two principle offsets can be found in a model.

It is worth mentioning that the sparsity is not equally notable for natural objects, as the red curve shows. For example, only 40.4% (compare to 79.9% of the man-made objects) of offsets are distributed in 7% of bins. Although there is still some sparsity, as the cumulative curve is above the diagonal line of an uniform distribution, we find it gives little advantage for model retargeting.

4.4 Implementation

In this section we provide implementation details of our algorithm. We use the popular SHOT descriptor for matching features. But any other descriptors, such as FPFH [Rusu et al., 2009], can also be used. Since we only consider translational offsets, we do not need rotationally invariant feature matching. We therefore fix the local reference frame of each feature to be aligned with the world coordinate frame. Once features are matched, we bin them into a 3D histogram, where each bin is a cube of 0.05 on each side. We detect peaks using non-maximum suppression with a local window size of 0.15 (consistent with the value of τ in Equation 4.1). Each peak represents a dominant offset of the model. Since those offsets are not very accurate at this stage, for the reason that they are defined at the resolution of bins, we increase their precision using a global iterative closest point algorithm on the entire point cloud.

Next, we find up to three principle offsets (c.f. Section 4.2.1) using a greedy selection strategy on a list of dominant offsets sorted in decreasing order of their histogram magnitudes. We select the first dominant offset in the list as the first principle offset $\mathbf{v}_1$, since it has the maximal number of matched features. The second principle offset $\mathbf{v}_2$ is the first remaining dominant offset that satisfies

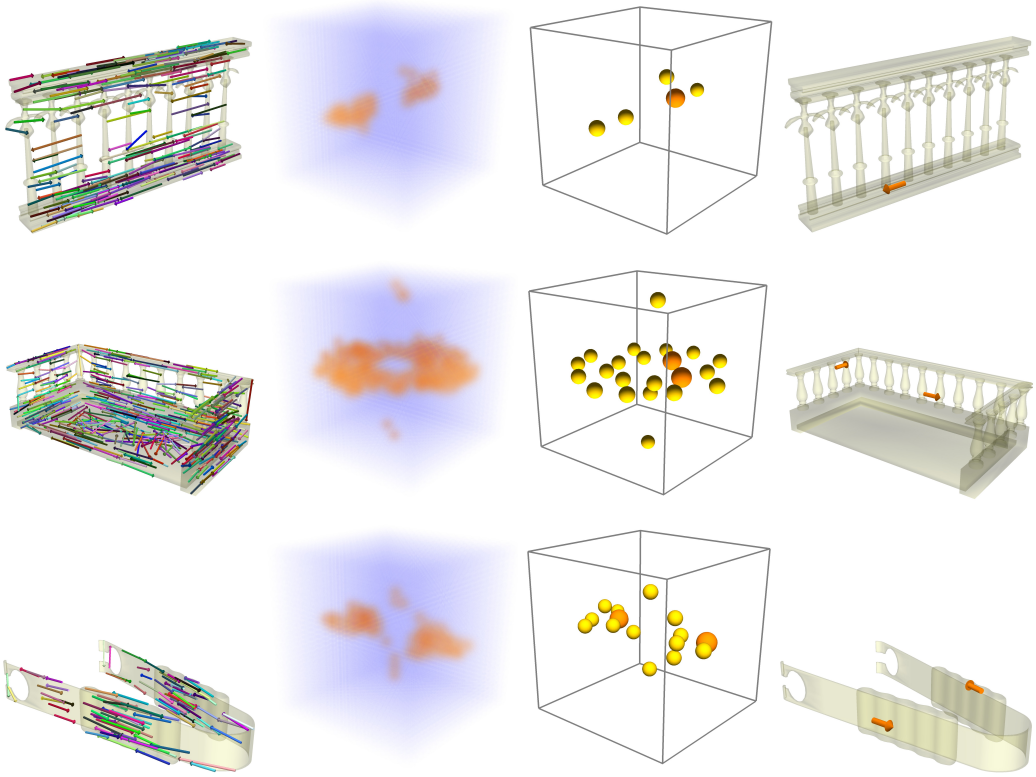


Figure 4.6: Offset statistics for different structures. Top: 1D grid, Middle: 2D grid, Bottom: sheared 2D grid.

$-\cos\theta \leq \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|} \leq \cos\theta$, which specifies a minimum angle between those two principle offsets. In practice we set $\theta = \pi/3$. The third principle offset, if there is any, is set to be the first remaining dominant offset that satisfies the minimum angle check with both of the first two principle offsets. Notice the number of detected principle offsets determines the degree of freedom for model retargeting. For example, if only one principle offset is detected, the model can only be resized in one direction.

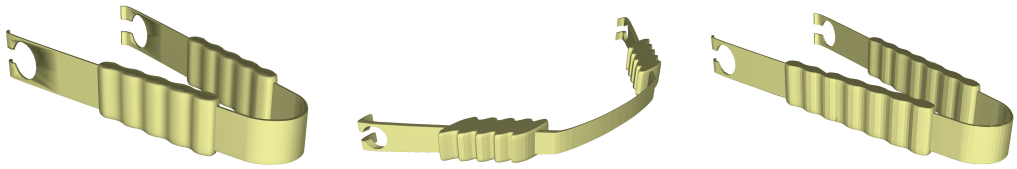
For the graph-cut optimization, we discretize the embedding space of retargeted model into a regular volume Ω as used in Eq. 4.3. The discretization uses principle offsets as unit length, so each voxel is of size $\{|\mathbf{v}_1|, |\mathbf{v}_2|, |\mathbf{v}_3|\}$, where $|\cdot|$ denotes the L2 norm. In cases where there are less than three principle offsets, we create up to two orthogonal dummy directions with length 0.2. These auxiliary offsets are only used to discretize the model, no actual computation will be performed along these directions.

To generate new models, we stitch together translational copies of the input model, as described in section 4.2.2. The translations are restricted to multiples of voxel size for two important reasons: First, to keep those translations consistent with the repetitive structure of input model. Second, to keep the number of graph-cut labels low for computational efficiency. In fact, the optimization is usually run on a graph of less than 1000 nodes with 30 labels, so retargeting can be performed at interactive speed. Despite using such relatively low complexity graph, our algorithm is still able to produce high quality results and preserve the exemplar structure in the synthesized models.

In practice we map the input model into a canonical space (i. e., 3D Cartesian coordinate system) if the detected principle offsets are not orthogonal to each other. The mapping is constructed by collecting all 3 normalized principle offset vectors into a 3D transformation matrix, which is then applied to all model vertexes. Figure 4.6 (bottom) shows an example where the object has two non-orthogonal principle offsets. Doing so allows the discretization of the model to be axis-aligned for fast geometry computation. Figure 4.7-a and 4.7-b show the same model in ordinary space and in canonical space. The retargeting is performed in the canonical space, and the result is mapped back to the ordinary space (figure 4.7-c).

4.5 Results

In this section we show our retargeting results. Figure 4.8 shows side-by-side comparisons between our results (yellow) and conventional MRF based results



(a) Input model (b) Canonical space (c) Final output

Figure 4.7: A model with sheared 2D grid (a) is mapped into a canonical space (b) where the two principle offsets are orthogonalized. The retargeting is performed in the canonical space and mapped back to the normal space (c).

without offset statistics (blue). The gray models are the input examples. The exact number of graph nodes, number of labels, and time for graph-cut optimization are also provided for each result. To generate the blue models, regular offset (set to 0.1) is used. Despite using much more complex graphs, conventional MRF produces shapes that do not preserve the structure of the input models, as highlighted in the close inspections. In contrast, our method uses simpler graphs but eliminated such artifacts. Our optimization (yellow models) is fast enough for real time modeling.

From figure 4.9 to 4.11 we show a gallery of our results. Notice that our result for the book shelf model (the first example) would be difficult to accomplish by previous methods, such as [Bokeloh et al., 2012b]. Figure 4.12 shows the detected degrees of freedom (DoF) for resizing this model using Bokeloh et al.'s method [Bokeloh et al., 2012b]. As it is shown there, the detection only covers a small part of the model due to irregularities in the shape. This lack of DoF can be an obstacle to synthesizing plausible new shapes. In particular, it often locks the entire shape out of being changed. In contrast, a MRF based method such as ours is less rigid and can still produce convincing results by synthesizing the model as a "texture" and preserve fuzzy repetitions using offset statistics. Compare to other retargeting methods that explicitly model shape structure [Talton et al., 2011; Lin et al., 2011c; Merrell and Manocha, 2011], our method is more flexible in different ways: it does not require prescribed procedural rules [Talton et al., 2011] nor manual decomposition [Lin et al., 2011c], and performs less sensitively to the discretization of shape space [Merrell and Manocha, 2011]. Compare to low level texture synthesis methods such as [Turk, 2001; Bhat et al., 2004; Zhou et al., 2006], our method handles man-made structures better, due to the capture of offset statistics.

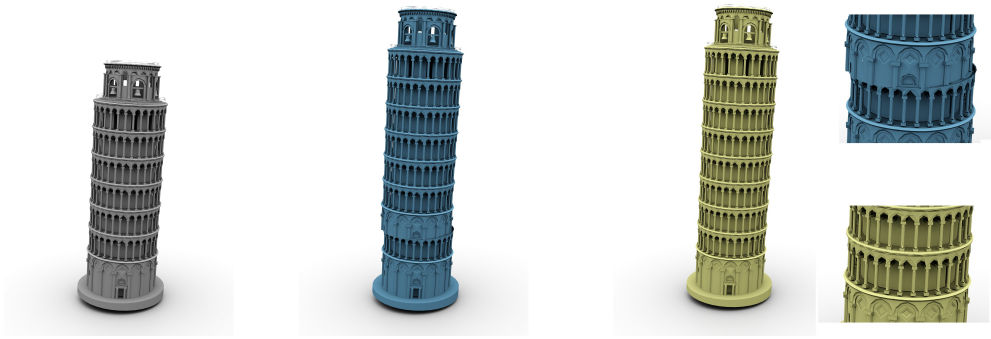
However, our current implementation has limitations for retargeting more complex structures. Figure 4.13 shows an example. In this case, the offset statistics is less sparse (top row, middle) and there are more than three dominant offsets

(top row, right) that influence the structure of the model. We believe such a shape needs to be modeled by multiple grids, otherwise the retargeting produces artifacts as shown in figure 4.13-b[1]. It is an interesting future direction of extending our method to work with such models of higher structure complexity.

4.6 Summary

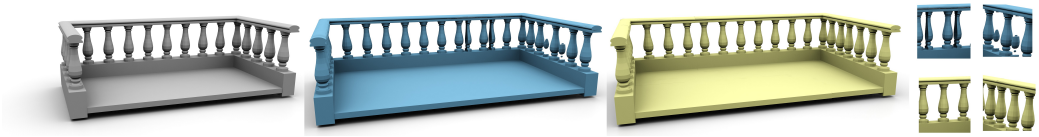
In this chapter, we have explored offset statistics for 3D shape retargeting. Our main argument is that the offset histograms between similar 3D features are sparse, in particular for man-made objects such as buildings and furniture. This observation is verified by our empirical studies, and act as a guiding principle in our pipeline. We employ sparse offset statistics to improve 3D shape retargeting (i. e., rescaling in different directions). We employ a graph-cut method that similar to texture synthesis, which iteratively stitches model fragments shifted by the detected sparse offsets. The offsets help to reveal important structural redundancies, which further leads to more plausible results and more efficient optimization. Our method is fully automatic, while intuitive user control can be incorporated for interactive modeling in real-time. We empirically evaluate the sparsity of offset statistics across a wide range of subjects, and show our statistics based retargeting significantly improves quality and efficiency over conventional MRF models.

[1] This result uses the top three dominant offsets for discretizing the retargeting space into a single regular volume, and the top seven dominant offsets for generating translational copies of the input model.

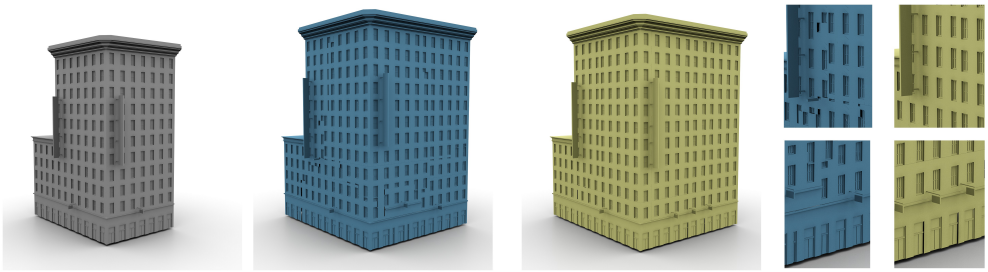


(a) Blue: 1600 nodes, 5 labels, 0.139 sec. Yellow: 360 nodes, 3 labels, 0.012 secs.

(b) Blue: 924 nodes, 8 labels, 0.082 sec. Yellow: 32 nodes, 6 labels, 0.008



(c) Blue: 2625 nodes, 25 labels, 0.956 sec. Yellow: 104 nodes, 9 labels, 0.013



(d) Blue: 7429 nodes, 54 labels, 5.151 sec. Yellow: 810 nodes, 8 labels, 0.072 secs.

Figure 4.8: Qualitative comparison between retargeting without (blue) and with (yellow) offset statistics.

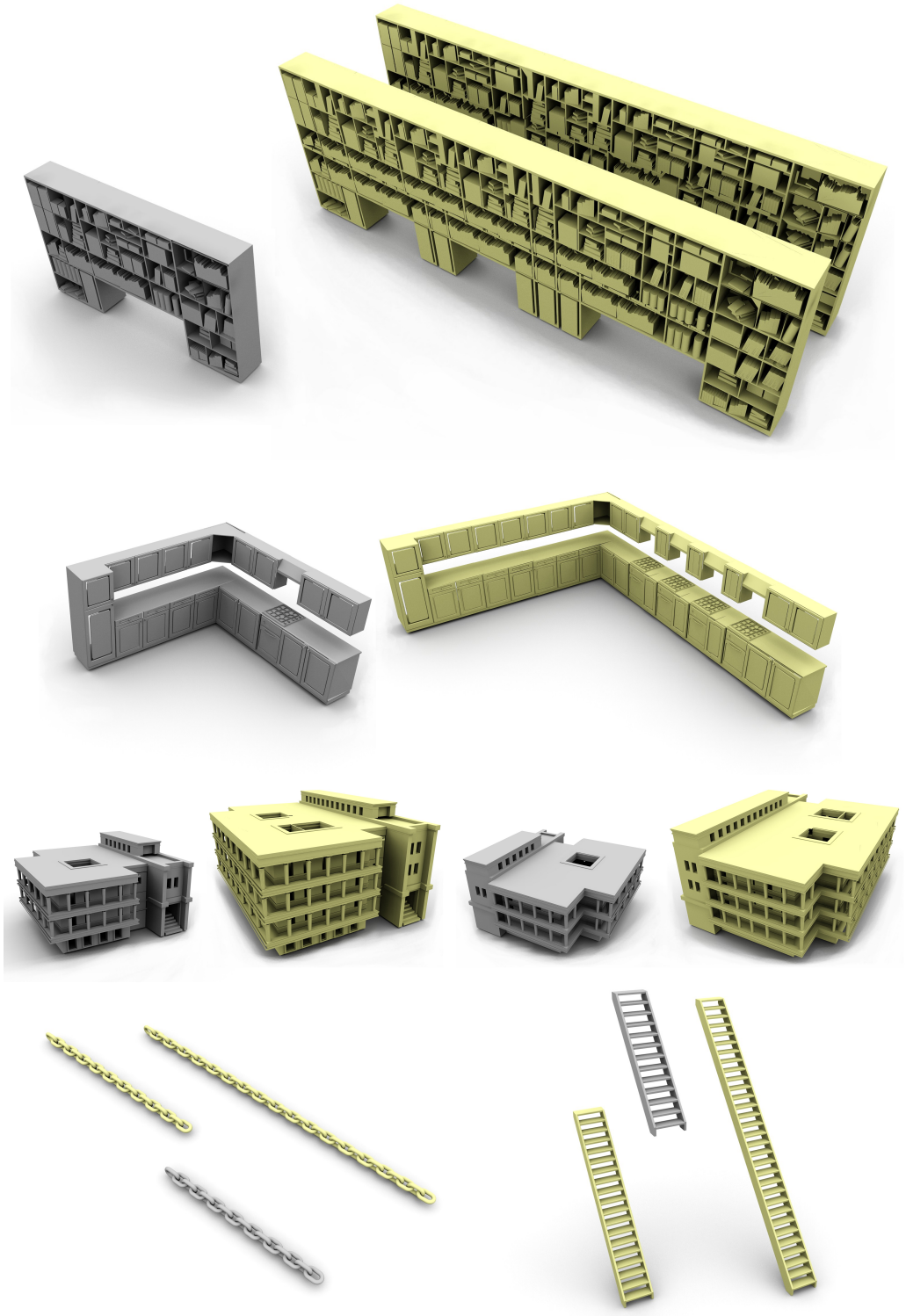


Figure 4.9: Retargeting results.

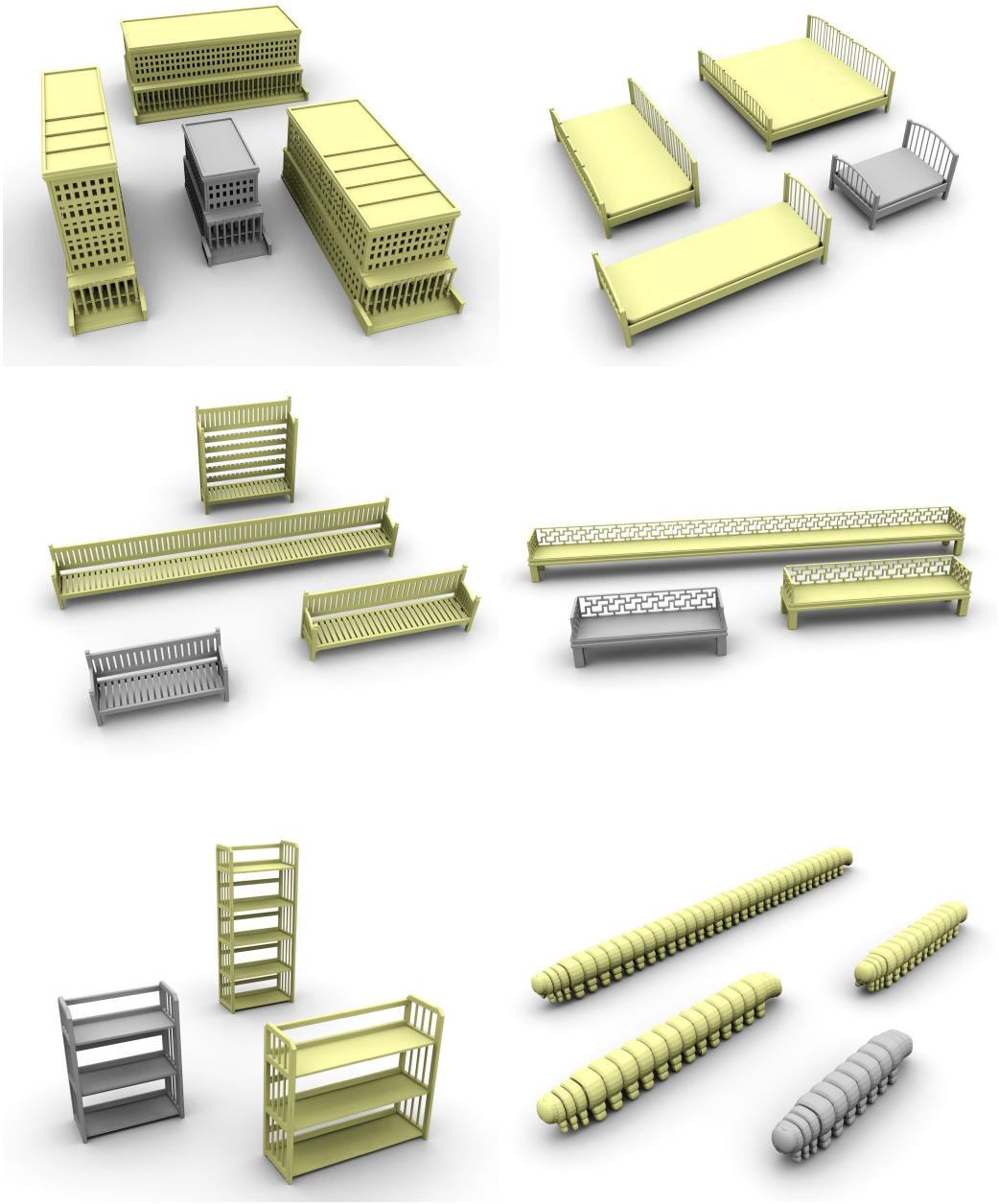


Figure 4.10: Retargeting results, continued.

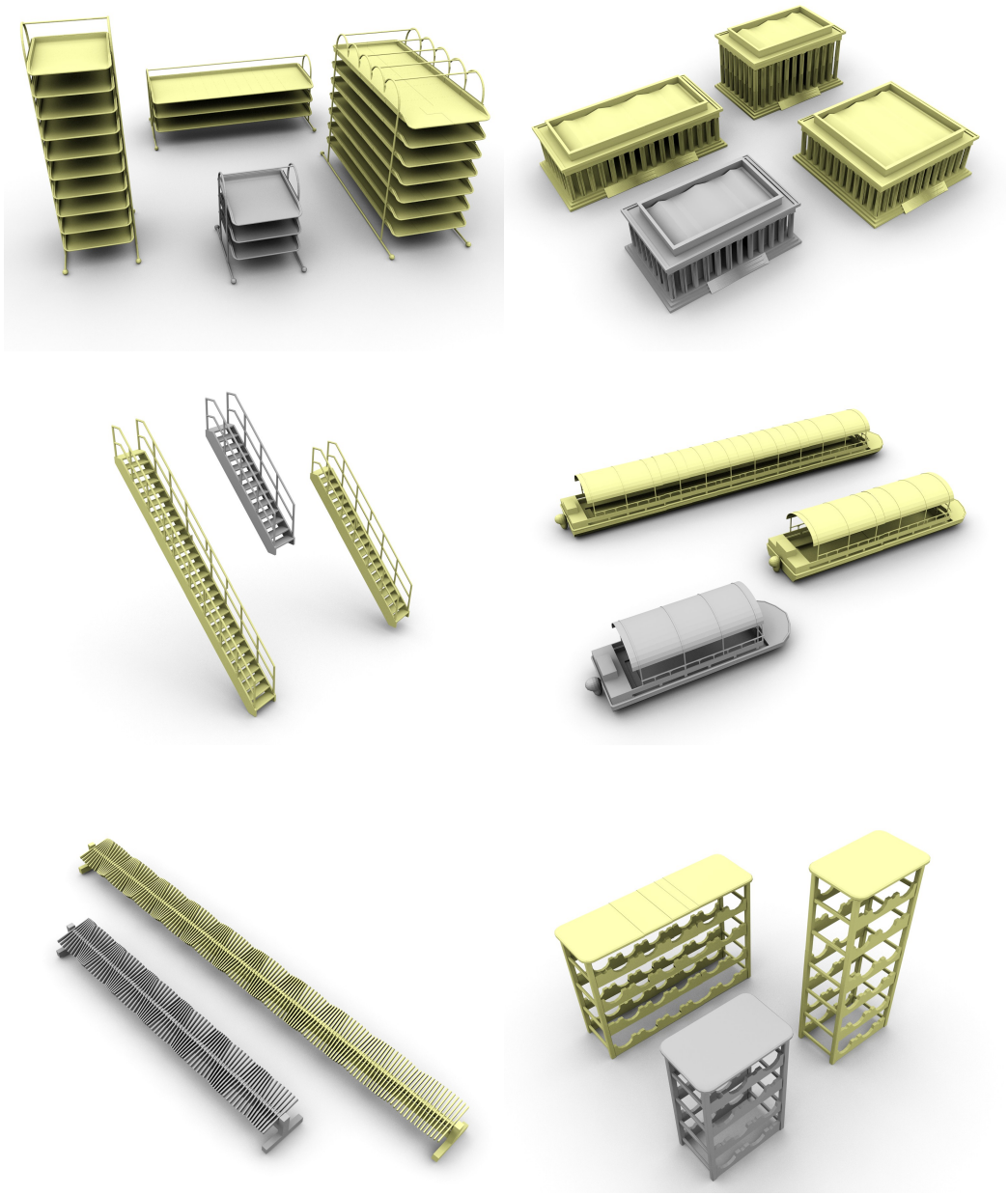


Figure 4.11: Retargeting results, continued.

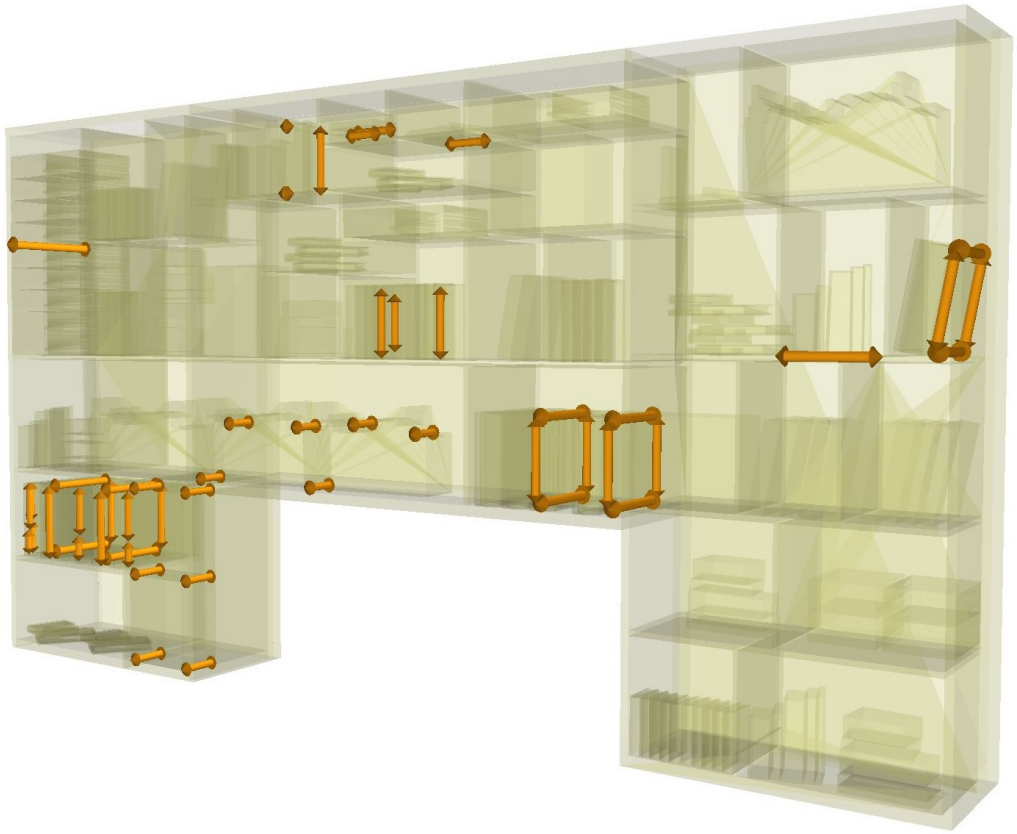
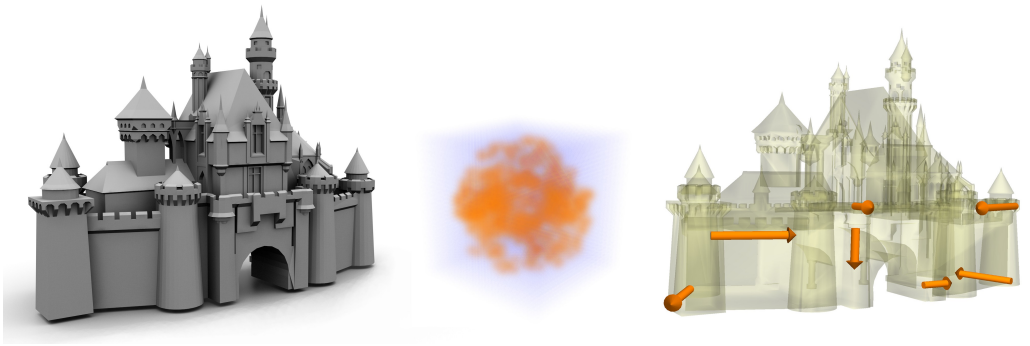
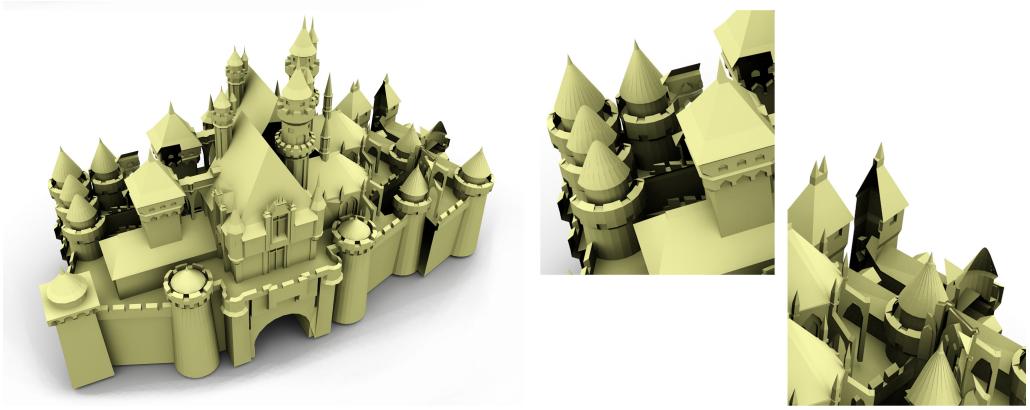


Figure 4.12: Linear constraints detected by [Bokeloh et al., 2012b].



(a) Input model, 3D histogram and dominant offsets.



(b) Synthesized model with artifacts in close view.

Figure 4.13: Limitation of our method.

Chapter 5

Approximate 3D Partial Symmetry Detection Using Co-Occurrence Analysis

As we have seen the importance of symmetry and its applications in previous chapters, now we take a step further down to the low level symmetry detection problem. The algorithm in this chapter could serve as a pre-processing step in many applications, including our previous work introduced in Chapters 3 and 4. We will focus on approximate partial symmetry detection, which applies to noisy data in 3D point clouds.

5.1 Introduction

One of the prime problems in contemporary computer graphics is understanding and organizing 3D models: while manual 3D modeling is still demanding and expensive, we have nowadays unprecedented data repositories at our fingertips that provide example data of synthetic and real-world geometry, ranging from fictional space ships to accurate city scans. This has spawned a lot of recent research in *structure-aware* modeling category [Mitra et al., 2013a], which base the synthesis of novel content on the analysis of existing example data.

One of the fundamental tools for analyzing shapes is *symmetry detection* [Mitra et al., 2013b]: Structured data differs from maximum-entropy random noise by some form of characteristic redundancy. Symmetry offers a model that captures

such redundancy: A shape is *globally symmetric* if it is invariant under a group of transformations acting upon it. For example, a gear is invariant under a small group of rotations, and this property is important for its functionality.

While global symmetry is useful for analyzing isolated objects and small parts (such as a single gear), the notion of *partial symmetry* is required to understand more sophisticated scenes (such as a gear-box). In the broadest sense (as employed in the graphics literature), a scene is partially symmetric if its non-trivial parts can be mapped to each other under isometric mappings [Mitra et al., 2006c; Gal and Cohen-Or, 2006; Bokeloh et al., 2009]. In addition, symmetry group structure can be imposed to restrict the focus specifically to regularly placed instances [Pauly et al., 2008a]. We use general model of unrestricted instance placement, and refer to such instances as *building blocks*.

Symmetry detection in the case of exact geometry is very well understood. As we will see in Section 5.2, many algorithms are available to detect both global and partial symmetries (with or without strict regularity). Unsystematic data noise and partial acquisition (which often occurs in 3D scanner data) can also be handled robustly [Mitra et al., 2006c; Bokeloh et al., 2009; Pauly et al., 2008a]. The reason is that transformation matrices have very few parameters, which can be reliably estimated as long as larger number of data points are available, even if they are noisy and/or incomplete. The main open issue is dealing with shape variability: it is still very difficult to detect geometric correlations that are similar in terms of semantics or functionality, but deviate in local details. Isometric matching models [Ovsjanikov et al., 2008; Xu et al., 2012c] can still handle varying embeddings of a fixed Riemannian manifold into 3D space, but it is challenging to deal with non-isometric deformations.

Robustness can be gained from transformation voting [Mitra et al., 2006c; Podolak et al., 2006], but their approaches are limited to coarse scale geometric structures due to the common voting space. Scenes with many instances can only be handled under further assumptions, such as regularity or hierarchy. As an alternative, feature-graph matching heuristics have been applied in [Berner et al., 2009, 2011], which achieved limited success on real-world data. Spectral diffusion in matching networks [Lipman et al., 2010] helps to cope with noisy and ambiguous data, but they are limited by failing to observe the notion of spatially contiguous building blocks.

In this chapter we introduce a novel unsupervised method for detecting approximate partial symmetry. It takes a 3D point cloud as input and detects its partial symmetries even when there is considerable geometric variability, and/or irregular distribution of repetitive instances.

We start by building a feature abstraction of characteristic high-curvature regions [Bokeloh et al., 2009]. As many recent proposals [Kerber et al., 2013; Herzog et al., 2015; Song and Xiao, 2014; Rahmani et al., 2014], we use spatial and orientational pooling to gain invariance against local shape deformation. However, such invariance comes at the price of increased false positives. We therefore need to integrate information over larger quantities of data to improve the signal-to-noise level. Our key idea is to extract each building block instance as a collection of consistently co-occurring features. While matching individual features is unreliable, aggregating many feature matches improves the detection. There are two sources of information to integrate: (i) the mappings between instances, and (ii) the area covered by each instance. Knowing either the mappings or the instances simplifies the search of the other. However, in practice we know neither initially, so this is a chicken-and-egg dilemma.

To solve this problem, we employ an *Expectation-Maximization* scheme, which is followed by spectral clustering to find optimal co-occurring *constellations* of features. We first build a dictionary of relevant features by k-means clustering, pruned by a sliding-window filter. Then the pipeline iterates between (i) the *expectation* step, which finds pairwise co-occurring features using current dictionary entries (words); and (ii) the *maximization* step, which updates the words so that the affinity between co-occurring features is maximized. This creates a dictionary of reliable features (Figure 5.1, feature detection). At last the spectral clustering step optimizes location and area coverage of the building blocks (Figure 5.1, building block detection).

In comparison to transformation voting [Mitra et al., 2006c; Podolak et al., 2006; Pauly et al., 2008a], our new approach has no restriction in cardinality and placement of the symmetric elements. Comparing to feature-graph matching [Berner et al., 2009, 2011], it is able to handle much stronger geometric deformation. Comparing to symmetry factored embedding [Lipman et al., 2010], our spectral clustering embeds the distance between *spatial patterns* of matched features rather than single correspondences. We demonstrate that our method comes with significant performance benefits. Unlike most of the previous methods, our approach optimizes for coherent constellations, therefore outputs building block instances of different classes rather than only pairwise correspondences (which are sometimes subject to greedy segmentation). In summary, this chapter makes two main contributions:

- We introduced consistent co-occurrence patterns as a novel invariant for improving feature matching in 3D point clouds.

- Based on this, we developed a novel unsupervised partial symmetry detector, which outperforms know methods in case of strong geometric variability.

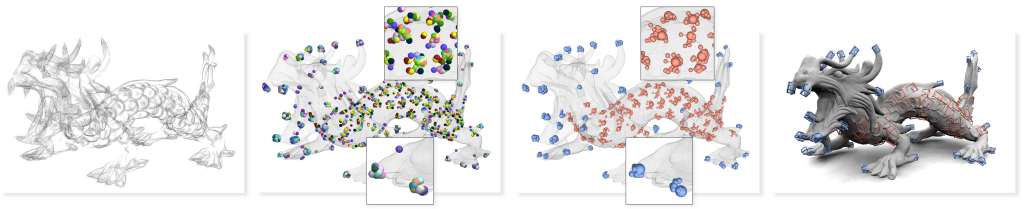


Figure 5.1: System overview: Input (preprocessed) — Feature Detection (distribution on model and close up look) — Building blocks detection (constellation visualization) — Detection results (bounding box visualization)

5.2 Overview

Building blocks are local maximal pieces of geometry that can be coherently mapped, as shown in Figure 5.2 (left). This section explains the concept of building block and its relation to coherent co-occurrence patterns.

Notation: Let $\mathbf{Q} = \{\mathbf{Q}^i\}_{i=1}^M$ denote the set of all M different *classes* of detected building blocks. Each class $\mathbf{Q}^i = \{\mathbf{q}_j^i\}_{j=1}^N$ consists of N *instances*. The instances $\mathbf{q}_j^i \subset \mathbb{R}^3$ are pieces of geometry that are rigid copies of each other and *approximately* match the input data. Hence, the j-th instance generates all other instances by $\mathbf{Q}^i = \{\mathcal{T}_{j\to k}^i \cdot \mathbf{q}_j^i\}_{k=1}^N, \mathcal{T}_{j\to k}^i \in E(3)$ (Figure 5.2, left). Without loss of generality, we consider the first instance $\mathbf{q}_1^i$ as template instance, with transformations $\mathcal{T}^i = \{\mathcal{T}_{1\to k}^i\}_{k=1}^N$.

Approximate building blocks are detected out of coherently co-occurring features that have the same spacial constellation, which appear repeatedly in all instances of the same class. Such *consistent co-occurrence pattern* is an invariant of its containing building block class. Let $\mathbf{D} = \{\mathbf{D}^i\}_{i=1}^K$ denote a feature dictionary, where each word $\mathbf{D}^i = \{\mathbf{d}_k^i\}_{k=1}^N$ is associated with a set of matched features $(\mathbf{d}_k^i) \in \mathbb{R}^3$. A pair of features *co-occur* if their lists of matches, i. e., $\mathbf{D}^i$ and $\mathbf{D}^j$, are identical, after applying the same *local* transformation $\mathbf{T}_{i\to j} \in E(3)$ to every member of $\mathbf{D}^i$ (Figure 5.2, right). In another word, $\mathbf{D}^j = \{\mathbf{T}_{i\to j} \cdot \mathbf{d}_k^i\}_{k=1}^N$. Using local coordinates at this point achieves rotationally invariant matching.

During detection, we identify co-occurring feature pairs across the data (in the EM step) and then assemble the pairs into instances (using spectral clus-

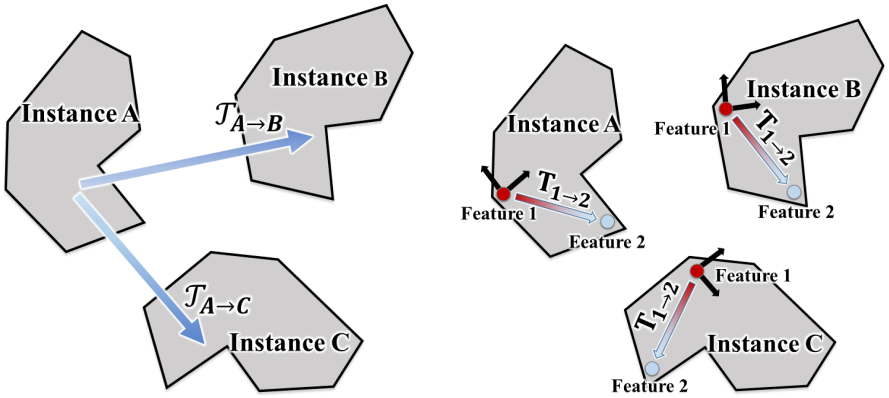


Figure 5.2: Left: $\mathcal{T}_{source\to target}$ maps instances of the same building block class. Right: $\mathbf{T}_{source\to target}$ maps co-occurring features in the same instance. The black arrows indicate the local frame of the source feature.

tering). In this way we aggregate co-occurrence information over the set of transformations $\mathcal{T}^i$ and target geometries $\mathbf{Q}^i$.

5.3 Method

This section explains our core algorithm. We will briefly introduce the input data and the necessary pre-processing step in Section 5.3.1. The two main steps of our algorithm will be explained in Section 5.3.2 and Section 5.3.3.

5.3.1 Pre-processing

Our input data is a point cloud with oriented normals. We re-sample the input point cloud using uniform sampling with spacing ϵ, which controls the processing resolution and consequently the scale of detection (see Section 5.4 for details about parameter settings). We estimate the maximum principal curvature on the surface (Figure 5.1, preprocessed input), from which we sample points with strong curvature as salient features. We denote this feature point cloud by Ω, and the original input point cloud where the features are sampled by Υ. Then we set up a local coordinate frame at each feature's location for computing its descriptor.

Our descriptor is a 3D variant of the Histogram of Oriented Gradient (HOG) descriptor [Dalal and Triggs, 2005]. We use curvatures as the analog of 2D-image

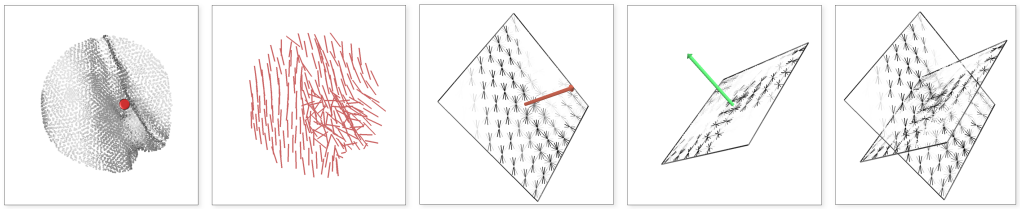
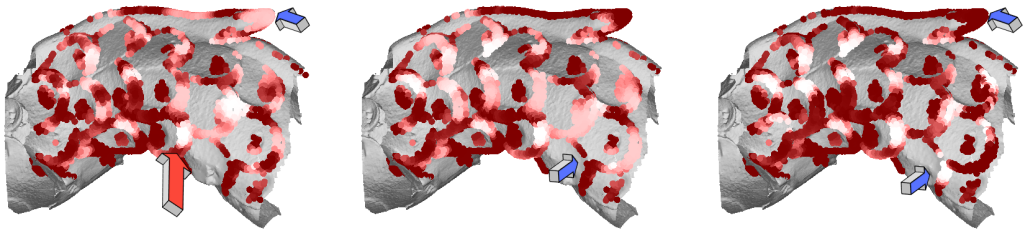


Figure 5.3: HOC descriptor is created from the curvature statistics of a local piece of geometry. From left to right: local geometry, curvature, HOC projected to the tangent plane, HOC projected to a secondary plane, the final descriptor.

gradients for 3D-surfaces, which produce a *histogram of oriented curvature* (HOC) descriptor [Kerber et al., 2013; Herzog et al., 2015]. Figure 5.3 shows an example of this type of descriptor, and Section 5.4 provides implementation details. Notice our core detection algorithm is independent of the descriptor design. Prior work also achieves good results from pooling various types of differential information (position distributions, normals, curvature, etc.) [Song and Xiao, 2014; Rahmani et al., 2014; Herzog et al., 2015].

Next, we build an initial dictionary $\mathbf{D}$ of frequently appearing features using K-means clustering. Each word $\mathbf{D}^i = \{\mathbf{d}^i_k\}_{k=1}^N$ consists a list of similar features. W.l.o.g., we mark the first feature as the template $\mathbf{t}^i = \mathbf{d}^i_1$, which has smallest distance to all other cluster members. We purify each word using a sliding window filter: features that are too close (within a minimum distance threshold set to half of the value that we used to define HOC descriptor) are excluded.

Now, each word $\mathbf{D}^i$ represents a list of matched features, which is a transformation set $\mathcal{T}^i$ that maps the template $\mathbf{t}^i$ to all other features in the same word (we use the terms "word" and "list of matches" interchangeably, following the formal definition of $\mathbf{D}^i$). Figure 5.4-a shows the sliding window response of a single feature. The response map is imposed onto the entire input geometry. Notice that the resulting feature response is noisy (a red arrow points at the query feature, and blue arrow points at a false positive). This is because strong appearance variation of the dragon scales is beyond the invariance of a single HOC descriptor. Simply increasing size of the descriptor does not help, since the feature response will be blurred. This observation shows that a naive feature dictionary is not sufficient for finding such complex building blocks.



a) Initial response b) 1st iteration c) 5th iteration

Figure 5.4: Feature matching can be improved by the proposed EM algorithm. The heat map encodes the probability of finding a match (white indicates high probability). a) single feature based detection (red arrow points at the query feature) is unreliable. For example there is false positives (blue arrow). b) The output after one iteration. The false positive is removed. But missing detection still exists (blue arrow). c) The output after five iteration.

5.3.2 Robust Co-occurring Feature Detection

The first contribution of our work is to use pairwise co-occurrence constraint to improve the low level feature detection. Specifically (and unlike [Li and Wand, 2015]), we use an iterative EM scheme to enforce pairwise co-occurrence constraints in the search of reliable features. We prefer pairwise relation instead of higher order co-occurrence for avoiding over-fitting in the unsupervised setting. The input of our algorithm is an improved feature dictionary $\mathbf{D}$. At each iteration, we first estimate the co-occurrence relation using exist words, then update the words so the affinity between the distributions of co-occurring features is maximized. A summary of our algorithm is listed in Algorithm 1.

Estimation. The aim is to estimate the probability of co-occurring features. Given the current feature dictionary, we first compute the best local transformation $\mathbf{T}_{i\to j}$ that maps $\mathbf{D}^i$ to $\mathbf{D}^j$, which essentially matches these two words' spatial pattern. A successful match is found for each mapped feature $\mathbf{T}_{i\to j}\cdot\mathbf{d}^i_k$ that has a close-by neighboring feature in $\mathbf{D}^j$. We exhaustively test all feature pairs in $\langle\mathbf{D}^i,\mathbf{D}^j\rangle$, and choose $\mathbf{T}_{i\to j}$ as the one that gives the maximum number of matches. We denote by $\Psi^{i,j}$ the union of source features from the matches.

Next, we compute $\mathbf{p}(\mathbf{d}^i,\mathbf{d}^j)$, which is the probability density function (PDF) for a pair of features $(\mathbf{d}^i=\mathbf{x},\mathbf{d}^j=\mathbf{y})\in\mathbb{R}^{3\times 2}$ to co-occur in Ω. We first use a voting based method to compute $\mathbf{p}(\mathbf{d}^j|\mathbf{d}^i)$: For each $\mathbf{x}\in\Omega$, we predict its counterpart using $\mathbf{T}_{i\to j}\cdot\mathbf{x}$, then add the prediction confidence to PDF. Specifically, we find all $\mathbf{y}\in\Omega$ that satisfy $||\mathbf{y}-\mathbf{T}_{i\to j}\cdot\mathbf{x}||<\sigma_s$, and vote for each of them with the following

probability:

$$\mathbf{p}(\mathbf{d}^j{=}\mathbf{y}|\mathbf{d}^i{=}\mathbf{x}){=}\exp\left(-\frac{||\mathbf{y}-\mathbf{T}_{i\to j}\cdot\mathbf{x}||}{\sigma_s^2}m(\mathbf{t}^j,\mathbf{y})\right) \tag{5.1}$$

Here σ_s controls the tolerance of mis-match, and $m(\mathbf{t}^j,\mathbf{y})$ computes the distance between the template and the prediction in descriptor space. Finally we compute the joint probability: $\mathbf{p}(\mathbf{d}^i,\mathbf{d}^j) = \mathbf{p}(\mathbf{d}^j|\mathbf{d}^i)\mathbf{p}(\mathbf{d}^i)$. Here $\mathbf{p}(\mathbf{d}^i)$ is the prior of seeing $\mathbf{d}^i$ in Ω. It is approximated as the spatial distribution of the word:

$$\mathbf{p}(\mathbf{d}^i=\mathbf{x}) \approx \begin{cases} 1 & \text{if } \mathbf{x}\in\mathbf{D}^i, \\ 0 & \text{if } \mathbf{x}\notin\mathbf{D}^i. \end{cases} \tag{5.2}$$

Notice this approximation significantly reduces the computational cost, as we only need to compute $\mathbf{p}(\mathbf{d}^j|\mathbf{d}^i)$ for $\mathbf{x}\in\mathbf{D}^i$ instead of $\mathbf{x}\in\Omega$.

Maximization. In this step we maximize the co-occurrence affinity by updating words. To do so, we recompute $\mathbf{p}(\mathbf{d}_i)$ as the integration of all hypothetical pairwise outcomes: $\mathbf{p}(\mathbf{d}^i) = \sum_{j=1,j\neq i}^{K}\mathbf{p}(\mathbf{d}^i,\mathbf{d}^j)$, where K is the size of the dictionary. We then update $\mathbf{D}^i$ as a new set of features detected as the local peaks in $\mathbf{p}(\mathbf{d}_i)$. Again, non-maximum suppression is used for detecting the peaks. Finally, we update the template $\mathbf{t}^i$ using new median feature of the word. The output of this EM process is a set of more reliable features. Figure 5.4 compares the feature response before and after the EM process: one iteration (Figure 5.4-b) already improves the feature response, and the result is further improved after five iterations (Figure 5.4-c).

Algorithm 1: Detection of Pairwise Co-occurrence Features

Data: Input feature dictionary $\mathbf{D}$, size of dictionary K, number of iteration M

for $t \leftarrow 1$ *to* M **do**
  for $i \leftarrow 1$ *to* K **do**
    for $j \leftarrow 1$ *to* K **do Estimation**:
      $\mathbf{T}_{i\to j} = \text{Match}(\mathbf{D}_t^i, \mathbf{D}_t^j)$
    for $j \leftarrow 1$ *to* K **do**
      $\mathbf{p}(\mathbf{d}_t^j|\mathbf{d}_t^i) = \text{Vote}(\mathbf{t}_t^j, \mathbf{T}_{i\to j})$ $\mathbf{p}(\mathbf{d}_t^i,\mathbf{d}_t^j) = \mathbf{p}(\mathbf{d}_t^j|\mathbf{d}_t^i)\mathbf{p}(\mathbf{d}_t^i)$
  for $i \leftarrow 1$ *to* K **do Maximization**:
    $\mathbf{p}(\mathbf{d}_{t+1}^i) = \sum_{j=1,j\neq i}^{K}\mathbf{p}(\mathbf{d}_t^i,\mathbf{d}_t^j)$ $\mathbf{d}_{t+1}^i = \text{FindPeak}(\mathbf{p}(\mathbf{d}_t^i))$ $\mathbf{t}_{t+1}^i =$ UpdateTemplate$(\mathbf{d}_{t+1}^i)$

Figure 5.5: Left: Co-occurrence features in the embedded space. Right: The median of each cluster (blue and red dot) is imposed onto the input model.

5.3.3 Instance Detection

Having found reliable features, our next task is to detect building blocks instances, which is the second contribution of our work. From a global perspective, we identify building blocks using spectral clustering. First we perform spectral embedding using co-occurrence measurement of pairwise features, then we use unsupervised clustering to extract different classes of building blocks and their supporting regions. Figure 5.5 shows an example of this process.

We define a co-occurrence matrix $\mathcal{M}$, where each entry is the affinity between spatial distributions of a pair of words ($|\cdot|$ denotes cardinality):

$$m_{i,j} = \frac{|\Psi(i,j)|}{\max(|\mathbf{D}^i|, |\mathbf{D}^j|)} \tag{5.3}$$

From $\mathcal{M}$ we create a low dimensional embedding using classical multidimensional scaling, where co-occurring features are close to each other. We extract modes from this embedding using mean-shift. This results in M different clusters ($M \leq K$), each representing a unique building block class $\mathbf{Q}^i, i = 1...M$. We use the median of each cluster $\overline{\mathbf{Q}^i}$ as an outlier-robust representation of the mode. The set of admissible transformations operate on $\overline{\mathbf{Q}^i}$ is the $\mathcal{T}^i$ that operates on the instances of this class.

Finally, we determine the supporting region of each instance. We build a "star model", which uses $\overline{\mathbf{Q}}^i$ as the centers of the instances, and links co-occurring features to their center. To do so, we add an edge for each matched feature pair from $\overline{\mathbf{Q}^i}$ and $\mathbf{D}^j$, where $\mathbf{D}^j \in \mathbf{Q}^i \backslash \overline{\mathbf{Q}^i}$. The size of each instance is approximated by the bounding box of its internal linked features. Figure 5.8 shows examples of our detection. The bounding boxes of all instances from the same class are averaged, which produces a more robust estimation.

5.4 Implementation Details

We now discuss additional implementation details. We re-sample the input point cloud using a sample space of ϵ, which controls the processing resolution and consequently the scale of the detection. In practice, we use different ϵ for data from different sources. It is a relative value to the diagonal length of each scene: 0.00005 for the *Qmulus&TerraMobilita* city scan dataset, and 0.0005 for the Stanford 3D scanning repository and the Aimatshape repository. We estimate surface curvature by applying quadratic moving-least-square [Alexa et al., 2003] algorithm to local neighborhoods of radius 4ϵ around each point, and then estimate the curvature tensor [Cazals and Pouget, 2005] for extracting the direction of maximum principal curvature $\mathbf{t}_1$ and its magnitude κ_1.

Feature point extraction. We sample points with strong curvature as salient features. A non-maximum suppression process is performed with a local search radius of 2ϵ. We use a threshold on the magnitude of the maximum principle curvature $\kappa_1 > 0.15$ to preclude specious local maxima.

Local feature frame. To set up a local frame for each feature, we use the normal of the query feature as the local x-axis, and the maximum principal curvature as the local y-axis. The local z-axis is the cross-product of previous two. To resolve sign ambiguity of the maximum principal curvature, we set up two different local frames as one is the 180 degree rotation of the other.

Descriptor. We compute curvature statistics around each query feature: each nearby sample point is projected onto local tangent plane. We collect the projected curvature into 8 orientation bins and 8×8 spatial bins, where each spatial bin has an edge length of 8ϵ. We also project curvature onto a second plane that uses local y-axis as the normal. Such dual projection increases the discriminative performance with just a modest increase of computational cost. In total our descriptor has 1024 dimensions. We normalize each descriptor so that its L^1 norm is one.

Feature Dictionary. The initial dictionary is built using K-means clustering(K set to 100). During sliding window detection, we compute matching score from the L^1 distance between two descriptors: $m(\mathbf{x},\mathbf{y}) = \exp\left(-\frac{|HOC(\mathbf{x})-HOC(\mathbf{y})|}{\sigma^2}\right)$. Here $\sigma = 0.1$ is a standard deviation parameter that models matching noise. The same matching function is used in Equation 5.1. Matches with score below 0.5 are removed during the non-maximum suppression to avoid spurious local maximum. The threshold is intentionally set to a relative low value to permit high noise scenarios and significant geometric variability.

Co-occurring Feature Detection. While matching the distribution of two features, we restrict the maximal spatial distance between the prediction $\mathbf{T}_{i\to j}\cdot\mathbf{d}_k^i$ and its nearest neighboring feature in $\mathbf{D}^j$ to be 8ϵ (one HOC cell). The matching process can be sped up by precluding features that are too far away in the input data: we require the spatial distance between $\mathbf{d}^i$ and $\mathbf{d}^j$ to be no larger than 64ϵ. However, it is still possible to detect instances that are larger than this size, due to the diffusion effect of spectral clustering. The σ_s in Equation 5.1 is set to 16ϵ. In the maximization step, we need to ensure that only reliable pairwise relations are kept in the EM process: First, we discard weakly co-occurring feature pairs via a threshold on cardinality of $\Psi(i,j)$. This threshold depends on the number of building blocks expected from the shape: 5 is used for individual shapes such as the Stanford dragon, and 10 is used for city scans. Second, we filter out weak feature responses during the updating of words (maximization step). In practice we normalize $\mathbf{p}(\mathbf{d}_j)$ so the highest scored feature has confidence of 1, and weak detections with confidence below 0.25 are excluded. For efficiency, we use up to 5 iterations for EM, and terminate the process as long as the dictionary does not have significant change (feature changes are less than 1% of total number).

Spectral Clustering. We perform a 5-dimensional embedding for the affinity matrix $\mathcal{M}$ using classical multidimensional scaling [Seber, 1984]. For unsupervised clustering, we use mean-shift with bandwidth of 0.5.

In practice we observe small building block classes (with fewer instances) can drift towards bigger building classes due to template updating in Algorithm 1. To solve this problem, we use an incremental detection scheme motivated by a minimal description length (MDL) argument. The task is to find the minimal number of building blocks classes that can describe most of the input data. To do so we run a sequence of building block detections. In each iteration, only the largest building block class in the result is kept. We remove features that are covered by this class from the feature point cloud Ω, so they would have no influence in the later iterations. In this way, the input model is iteratively decomposed into building blocks of cardinality in decreasing order. The result in Figure 5.6 is produced so. In total we found five different building blocks that can explain the input model in an efficient way.

Platform. We have implemented our algorithm in C++, and run on a 2.5Ghz quad-core Intel Core-i7 processor. The biggest computational bottle neck is the iterative EM process. Generally speaking, more complex symmetry has larger word, which leads to higher computational cost. In practice our implementation takes less than a minute to process each model in Figure 5.8, 62 seconds to process "dragon" and 132 seconds to process the model in Figure 5.6.

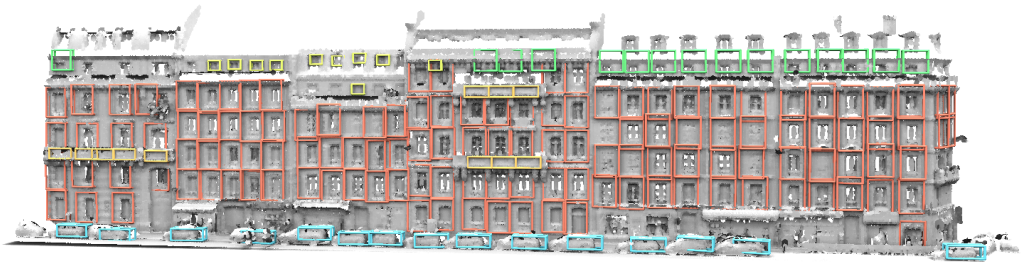


Figure 5.6: Here we find windows (red) and cars (blue) in the initial detection. We also find balconies (dark yellow) and two classes of smaller windows (with and without the dormer) as the green and the bright yellow building blocks in the late MDL iteration.

5.5 Evaluation

We conduct a quantitative evaluation on the *Qmulus&TerraMobilita* city scan dataset [Vallet et al., 2015] and on a set of non-man-made shapes from the Stanford 3D scanning repository and the AimAtShape repository. These data sets contain scan noise, irregular structure and/or strong geometric variability. We compare our method (EM + SC) with rigid ICP (RigidICP), symmetry factor embedding (Symmetry Embedding) [Lipman et al., 2010], and our implementation of [Liu and Liu, 2013] in 3D (Grasp). To show the importance of the proposed EM scheme, we also compare to spectral clustering without the EM optimization (SC), and to spectral clustering using [Li and Wand, 2015]'s discriminatingly learned features (DL + SC). We provide precision-recall curves for both datasets (Figure 5.9 a-b). Like [Liu and Liu, 2013; Li and Wand, 2015], we conduct a pressure test by varying the minimum overlapping ratio for instance matching and report the results in Figure 5.9 c-d.

5.5.1 Dataset

Previous work on partial symmetry detection mainly use qualitative evaluation, since very few dataset provides ground truth annotations. We use our own interactive tool to create ground truth annotation: a user first paints a piece of geometry as the template for query, then finds other instances using rigid ICP. Since rigid ICP does not work with strong geometric variability, the user often first over-decomposes the data into rigid pieces and then merges them by semantics. For example, windows of different sizes and styles will first be labeled as different classes, then will be merged into a single "window" class. We uniformly partition the entire *Qmulus&TerraMobilita* dataset into 96 scenes,

and perform annotation for each scene. There are three common building block classes across the entire dataset: window, car and balcony. Other smaller classes represents unique street furniture and building decorations. For the non-man-made shapes we annotate each individual model, and often build unique set of classes for them. For example the model "Dragon" has classes "scale" and "craws"; and the model "Buddha" has a single class for the decorative pattern on the cloth.

5.5.2 Methodology

We quantify the results using the standard precision and recall (PR) analysis and an additional stress test described in [Li and Wand, 2015]. Like [Song and Xiao, 2014], we match 3D bounding boxes of the instances from the detection and the ground truth annotation. Let $|\mathbf{D}|$ and $|\mathbf{G}|$ denote the number of different building block classes detected/annotated. We define a symmetric matching score between two classes $\mathbf{D}^j \in \mathbf{D}$ and $\mathbf{G}^i \in \mathbf{G}$:

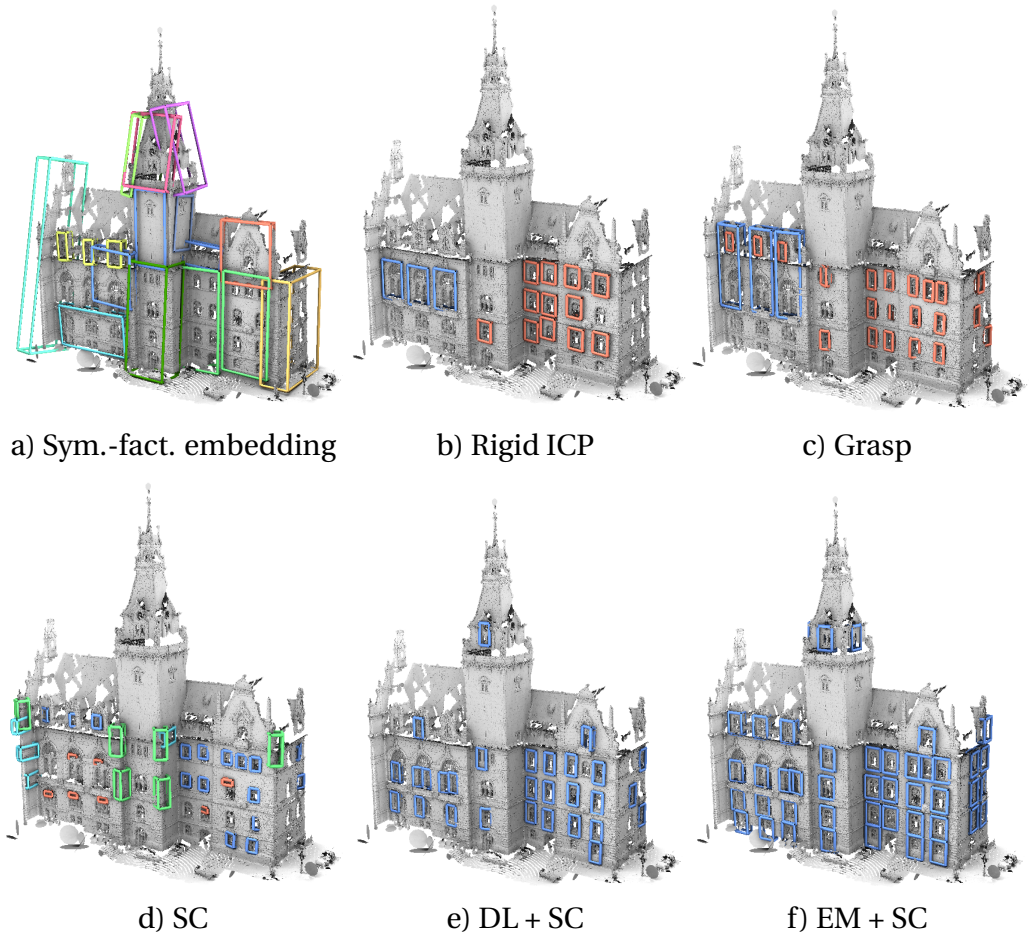
$$\text{score}(\mathbf{D}^j, \mathbf{G}^i) = \min(|\mathbf{D}^j \cap \mathbf{G}^i|, |\mathbf{G}^i \cap \mathbf{D}^j|) \tag{5.4}$$

where $\mathbf{X} \cap \mathbf{Y}$ denotes the set of bounding boxes in $\mathbf{D}^j$ that overlap with $\mathbf{G}^i$. Taking the minimum makes the score symmetric and avoids over-counting objects that have multiple intersections. This matching score can be computed for all possible assignments between the detected classes and the annotated classes. We define precision as the number of matched instances in $\mathbf{D}$ divided by the total number of instances in $\mathbf{D}$:

$$\text{precision} = \frac{\sum_{j=1}^{|\mathbf{D}|} \max_{i=1..|\mathbf{G}|}(\text{score}(\mathbf{D}^j, \mathbf{G}^i))}{\sum_{j=1}^{|\mathbf{D}|} |\mathbf{D}^j|} \tag{5.5}$$

Notice that each class $\mathbf{G}^i$ has its own matching score for $\mathbf{D}^j$, and only the highest score, $\max_{i=1..|\mathbf{G}|}(\text{score}(\mathbf{D}^j, \mathbf{G}^i))$, is kept for $\mathbf{D}^j$. We compute recall in a symmetric way by swapping the $\mathbf{D}$ and $\mathbf{G}$ terms in Equation 5.5. Like [Li and Wand, 2015], we generate a PR curve by successively removing building block classes from the detection, and simultaneously updating the recall and precision. Starting from the largest class, we remove one class at a time until no class remains. The final PR curve in Figure 5.9 is the average of all scenes.

We perform a stress test by incorporating a threshold of minimal overlapping ratio in Equation 5.4. Figure 5.9 a-b are generated using threshold 0.125, meaning there is on average half overlapping for each bounding box dimension. Increasing the threshold from 0 to 0.5 creates the F-measurement v.s. pressure curves as shown in Figure 5.9 c-d.



a) Sym.-fact. embedding b) Rigid ICP c) Grasp

d) SC e) DL + SC f) EM + SC

Figure 5.7: Detection results of different methods.

5.5.3 Results

We first discuss the PR curves of different algorithms (Figure 5.9 a-b). The baseline is a supervised detection using rigid ICP (purple). For each class in the ground truth, we perform a sliding window detection using the template from user annotation. The resulting low F-scores (0.55 and 0.48 on different datasets) indicate strong geometric variability can not be captured by rigid matching.

Symmetry factored embedding (pink) [Lipman et al., 2010] computes the symmetry factored distance between the input data and its transformed copy. Like [Lipman et al., 2010], we used a fairly large fraction of the entire shape (set to 20%) to compute the symmetry factored distance. We then use this distance to embed features and apply spectral clustering (with 10 clusters). Doing so

produces an feature point cloud segmentation. We "cut out" individual instance using a 26-connected 3D region growing. Doing so produces very low F-scores (0.48 and 0.45 on different datasets), indicating local diffusion can fail in detecting complex partial symmetry. We also experimented with replacing [Lipman et al., 2010]'s shape matching function by matching local feature descriptors computed from the same fraction of geometry (20%). However, no clear improvement could be observed in our experiment.

Next we test our 3D implementation of [Liu and Liu, 2013] (Orange), which uses stochastic search to find correlated features. We implement their search algorithm with our 3D HOC features, and observe a significant improvement (F-score 0.63 and 0.60) over the baseline. However, there are still many false detections. The reasons are three folds: first, unsupervised clustering usually produces inaccurate low level features; second, inaccurate features can not be traced due to the lack of iterative refinement; third, their search algorithm is greedy and the use of *average affinity* as the optimizing objective biases the result towards false negatives.

In contrast, our algorithm with EM optimization and less greedy spectral clustering (red) is able to significantly improve the performance. It achieves F-score of 0.74 on the Qmulus&TerraMobilita dataset and 0.68 on the non-man-made dataset. We also test intermediate results of our algorithm by dropping the EM optimization (SC, blue). Doing so significantly drops the performance (0.58 and 0.54). This indicates spectral clustering indeed requires reliable input features. We also test the discriminative learning scheme from [Li and Wand, 2015] (DL + SC, green), which trains a one-vs-all linear SVM for each feature. However, in our experiment this only gives marginal improvement over [Liu and Liu, 2013], due to the additional frame noise for computing the feature descriptors. Figure 5.7 show a qualitative comparison between different methods.

The same conclusion can be seen from the pressure tests (Figure 5.9 c-d). This again proofs aggregating consistent co-occurrence information from data improves the signal-to-noise level in the feature matching, and consequently benefits the final detection of building blocks.

Figure 5.8 shows our detection handles different challenges: figure 5.8-a shows it handles scanning noise; figure 5.8-b shows it detects irregularly placed instances. Figure 5.8-c is a very challenge case, where the windows have very strong geometry variability. Nonetheless, our algorithm is able to identify most of them as the same class of building blocks. Figure 5.8 d-f are examples of our detection on non-man-made shapes.

5.6 Applications

To this end we have introduced an effective building blocks detector, and evaluated its strength. In this section we show three applications of our method: detecting clusters of constellations, un-supervised co-detection and shape retargeting.

5.6.1 Detecting Clusters of Constellations

Our method works particularly robust with rigid symmetry. Thanks to the invariance of *HOC* descriptor and the co-occurrence clustering, it also has certain strength in handling non-rigidity. Nonetheless, the model so far is fundamentally a rigid detector because only a single mapping (the one with the highest number of matches) is allowed for a pair of co-occurrence features. This causes problems in detecting strongly deforming objects. How to handle free deformation is beyond the scope of this thesis. Nonetheless, we can show that it is possible to extend our method to handle more general deformations with a simple relaxation.

To do so we allow multiple transformations between each pair of co-occurrence features. So $\Phi^{i,j} = \{\phi_1^{i,j}, ..., \phi_{N_{ij}}^{i,j}\}$ is no longer a single transformation, but a set of transformations between $\{\mathbf{D}^i, \mathbf{D}^j\}$. However, not all possible transformations have be considered because only a small subset of them represent building blocks. These transformations must satisfy two criteria: first, each transformation applies to multiple instances, which is used to filter out specious transformations; second, the selected transformations should be exclusive, which is used to remove redundancy in the mappings. For example, we are not interested in detecting a group of objects, rather than the same class of objects with shape variations.

To apply this idea in practice, we modify the matching algorithm of co-occurring words accordingly: we collect N_{ij} transformations that satisfy the above two criteria for each pair of co-occurrence words $(\mathbf{D}^i, \mathbf{D}^j)$. The first transformation $\phi_1^{i,j}$ is the one that gives the highest number of matches $\psi_1^{i,j}$. We then remove the matching instances from those two words before looking for the next transformation ϕ_1, which is the one that gives the highest number of matches among the remaining instances. We continue the process until no transformation can be supported by sufficient number of matches. In the building block detection stage, the total number of matches $\Psi^{i,j} = \sum_{k=1}^{N_{i,j}} \psi_k^{i,j}$ is used to compute the co-

occurrence matrix $\mathcal{M}$ in Equation 5.3. Figure 5.10 shows an example of detecting windows of different sizes.

5.6.2 Unsupervised Co-Detection

Another application of our method is un-supervised co-detection on a collection of data. Figure 5.11 shows an example of our results using data from [Shen et al., 2012]. We put multiple objects in the scene to have enough re-occurrence of the building blocks. The uniform distribution of objects is for visualization, our detection does not require regular spaced objects. As the figure shows that without changing the detection pipeline, we are able to find most of the common parts in this example as building blocks in a complete un-supervised fashion. Nonetheless, our implementation requires curvature based features, which will get into trouble with a broader range of subjects, for example human and animals. But still, since our algorithm is not depended on the choice of features, we would like to leave feature design issues as a future direction.

5.6.3 Retargeting

Our last application is shape retargeting (Figure 5.12), based on a similar idea as described in Chapter 4. The method has the limitation that it only works well when low level feature detection is reliable, typically on clean meshes. In contrast, with our new detection algorithm, we can use the offsets between building blocks for guidance, which is more robust for noisy input data (e. g., from distance sensing). Details of the synthesis algorithm can be found in Chapter 4. Here we briefly outline the main steps and the differences between these two methods.

First we discretize both of the ambient spaces containing input data and target volume into spacial voxel grids. We use the same parametrization, and the voxels are oriented to align with a user specified generating direction $\vec{z}$ and two other orthogonal directions. Then the task is converted into a labeling problem: each input voxel has a unique label, and for each of the target voxel, we need to assign a single label from the input volume, which means the geometry will be copied exactly from the input voxel with the same label. To well define the problem, we measure the quality of each possible label assignment, such that target border voxels can only choose labels from input border voxels, and choosing empty voxel label is always associated with a penalty.

With the weak unary boundary condition described above, we can already find a plausible solution by optimizing the measurement energy. But the result often looks messy, since we do not enforce any constraints between neighboring voxels yet. So we add another binary smoothness condition, which measures the quality of choosing a pair of labels for adjacent target voxels. For each of such voxel pairs, we only measure the continuity of their coinciding faces using a distance function. We define each side of a voxel with η% of volume as a *joint*, and normally η is larger than the median of all pairwise point distances to allow certain level of data noise (in our implementation we choose $\eta = 10$). Then the distance function is constructed by summing up the minimal distances from each point within a joint to its counterpart. We also use the prior that nearby voxels from input volume are more consistent to each other than voxels with large distance, so if the label pair is taken from voxels with distance δ, we apply a penalty multiplier $\exp(a\delta)$ to the distance function. Notice that this smoothness measurement is unaware of geometric feature, which is simple but still produces correct results by using our robust generator $\vec{z}$. In contrast, the implementation in Chapter 4 uses special spacial parametrization, so the number of generating directions is limited.

The final solution is found by solving a functional summing both of the unary and binary energy using graph-cut [Boykov et al., 2001b]. With the optimal label assignment, we copy and join geometry pieces from corresponding input voxels, and output a single point cloud.

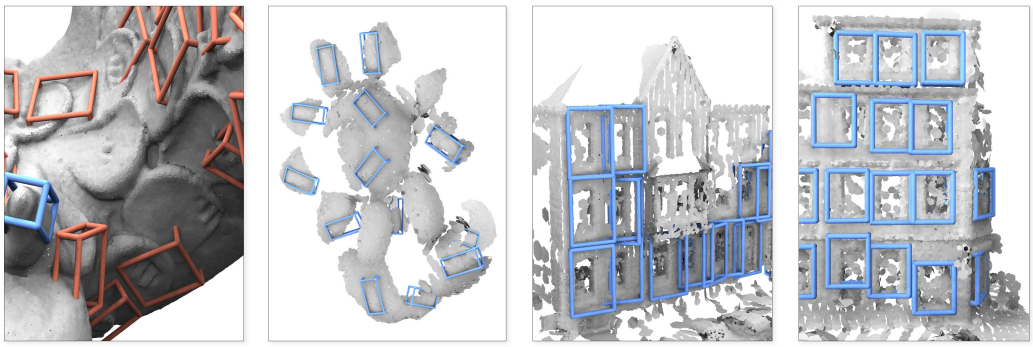
5.7 Limitation and Future work

Our method could be improved at both the feature level and the structural level. Figure 5.13-a shows missing detection due to strong deformation that is beyond the invariance of HOC features. Figure 5.13-b shows the lack of curvature in the data leads to non-discriminative features. In this case our method failed to separate building blocks from the background noise. Using the entire shape for symmetry detection works better here as shown in [Lipman et al., 2010].

On the structural level, our method detects general re-occurrence patterns instead of any specific forms of symmetry, such as reflective symmetry or lattice structure. This has been proved to be flexible while, to our best knowledge, none of the other specific methods works as general as ours. However, due to the lack of high level constraints, our method is less accurate when the noise level is too high. Figure 5.13-c shows miss-detections due to strong variance of data

sampling density, and figure 5.13-d shows misaligned building blocks. In these cases, reflective symmetry or lattice structure can be used to improve the results.

Last but not the least, our current implementation is not optimized for very big (city-scale) dataset. The main issue is that bigger data requires sensitivity to a growing variety of different items. This leads to larger dictionaries, whose sizes contribute quadratically to the run-time costs. In addition, with increasing variability of building blocks, identifying significantly different constellations might become an issue, e. g., small classes may drift towards big classes. The MDL approach, which works well enough for the scale of our examples, might need to be extended here in a more principled way.



a) Deformation b) Weak Feature c) Reflection d) Lattice

Figure 5.13: Limitations of our methods.

5.8 Summary

In this chapter We have presented a new method for approximate partial symmetry detection in 3D point clouds, a classical and foundational tool for analyzing geometry. The main idea is that coherent co-occurrence patterns in local frames reveal building blocks of the shape. By (spectral) clustering these co-occurrence patterns, we can integrate the whole (yet unknown) area of the whole (yet unknown) set of instances in order to detect building blocks. This yields three advantages: First, the simultaneous utilization of all of the available correspondence information reduces matching noise, so that we can use feature detectors with strong invariance and operate at rather ambiguous matching scenarios that cannot be handled by previous methods. To the best of our knowledge, our method is the first *object* detection method that recognizes subtle patterns such as the scales on the Stanford dragons or the cars in a city scan without any supervision and additional regularity assumptions. Second, the notion of

coherent co-occurrence patterns does not depend on rigidity; as shown in our extension towards varying clusters of feature constellations that are detected as the same class despite strongly non-rigid shape variation. Third, coherent co-occurrence patterns naturally characterize building blocks in the sense of maximal permutable pieces [Kolojanov et al., 2012], which yields a canonical encoding of the discrete symmetry structure of the scene and avoids ambiguities of previous methods that rely purely on region growing or boundary regularization.

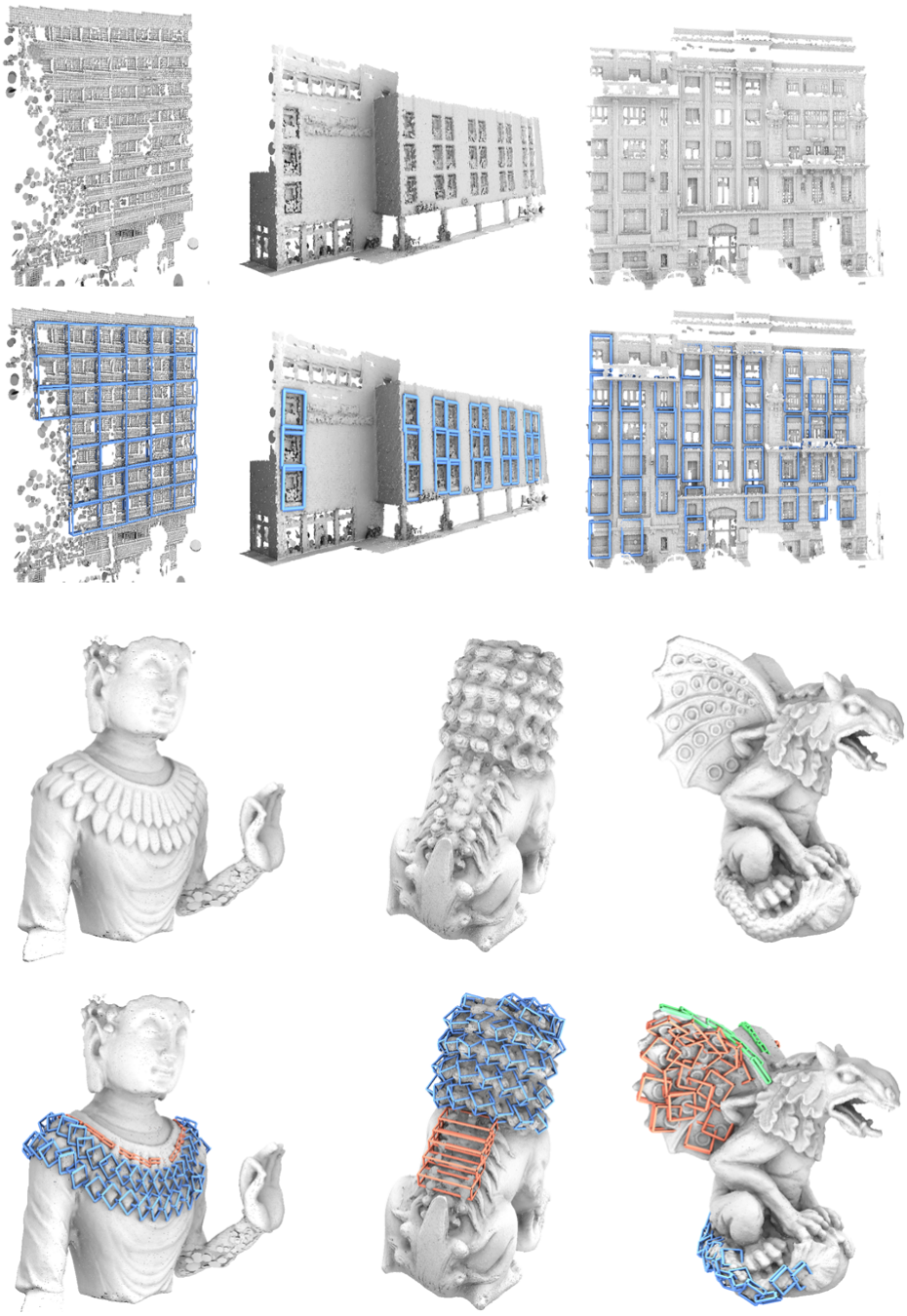
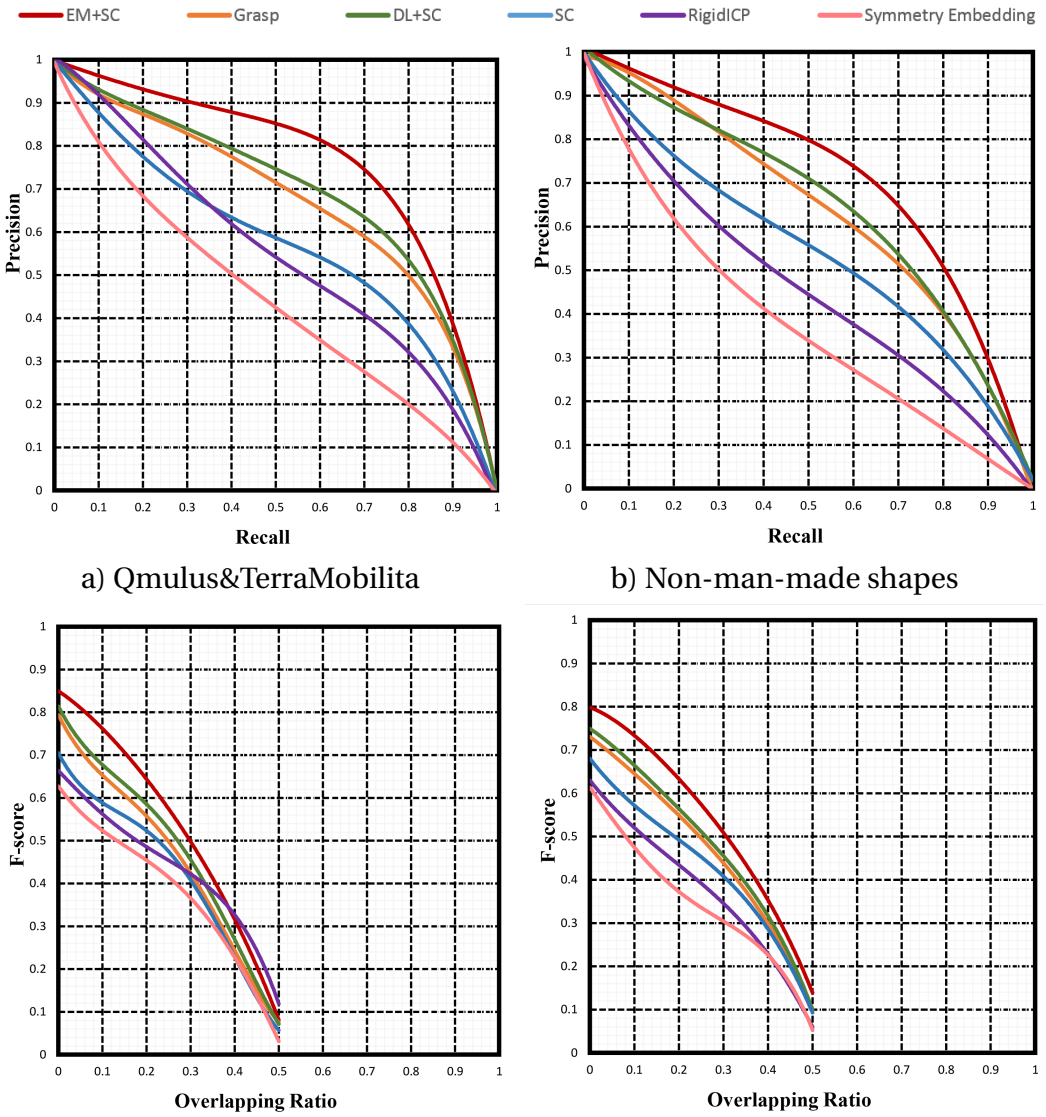


Figure 5.8: Detection results of different challenges.



a) Qmulus&TerraMobilita

b) Non-man-made shapes

c) Qmulus&TerraMobilita

d) Non-man-made shapes

Figure 5.9: Quantitative evaluation of our method. a) and b) are precision-recall curves on different datasets. c) and d) are pressure tests with different settings of overlapping ratio.

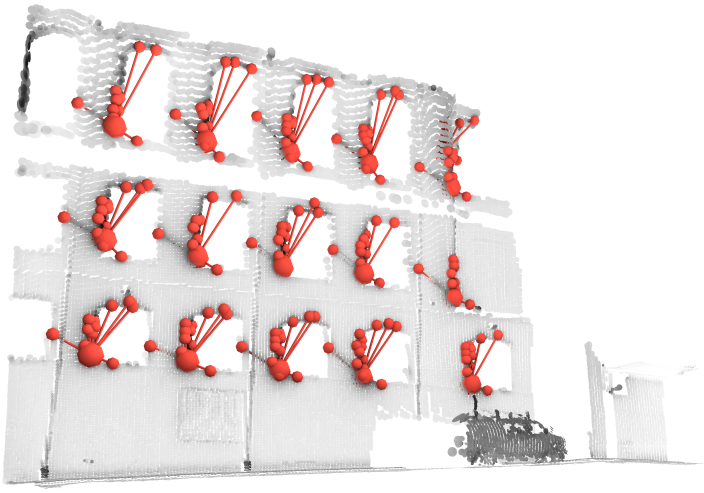


Figure 5.10: An example of detecting clusters of constellations.

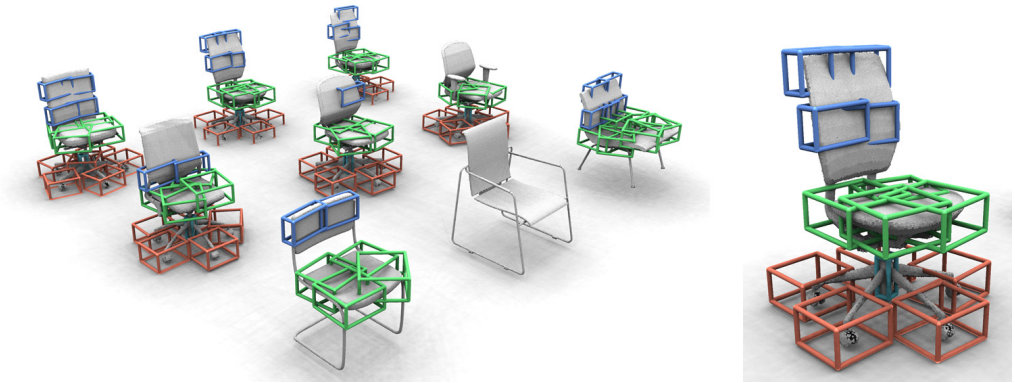


Figure 5.11: An example of un-supervised co-detection.

Figure 5.12: An example of shape retargeting. The input models are from Figure 5.7 and Figure 5.8.

Chapter 6

Conclusion

To conclude this thesis, in Section 6.1 we will briefly summarize our main points and major contributions. Then in Section 6.2 we will point out possible future directions on the basis of our constructions.

6.1 Final remarks

In our current information age, 3D digital content creation is gaining ever-increasing attentions, and becoming an important problem of compute graphics. Depending on different format of initial input data, the 3D modeling process is either based on pure manual operations, or post-processing of laser scanned data. But it is difficult to achieve full automation, due to inherent obstacles in both of those approaches. In Chapter 1, we introduced these difficulties, and answered the question why exemplar model is important, and why data analysis and knowledge extraction is necessary for the raw input.

To alleviate the heavy burden in the modeling process, people have developed many effective algorithms to solve several specific problems. Most of them rely on exploiting and conforming to hidden structure guidance, i. e., structure-aware operations. The combination of pattern-complete examples and generative rules extracted from structure analysis is the key to achieve automatic generalization, which made mass-production of shape derivatives possible. Among structure classes, symmetries attracted most attention, because this elegant mathematical tool can explain pattern regularities in very few concise and abstract axioms. In Chapter 2 we elaborated on most necessary background

knowledge, especially on formal definitions. We also listed related work there, and surveyed how other people treated this problem.

Next we have entered into the main body of this thesis. From Chapter 3 to Chapter 5, we have illustrated different experiments to support our argument for symmetry-aware content creation. These chapters are based on the published papers [Kurz et al., 2014; Wu et al., 2014a; Wu et al., 2014b; Li et al., 2015], and some further extensions by the author of the thesis.

In Chapter 3 we take the manual modeling process as the application context. To ease this tedious task, we presented a method for real-time symmetry-preserving shape editing. The basic idea is the construction of spaces spanned by low-frequency deformations that preserve symmetries. Within these low-dimensional spaces, we apply a non-linear deformation-based editing scheme. We demonstrate a method for real-time shape deformation that preserves the symmetries exactly and supports large deformations. The method is easier to implement than previous optimization-based methods and significantly faster.

In Chapter 4 we developed an automatic scheme to easily create geometric variations of a given 3D object. In this chapter we explore offset statistics for 3D shape retargeting. The main contribution is to detect structural redundancy as sparse offset statistics, and then use it for reducing the search space of suitable shapes. Our method works fully automatic, and produces significantly better results than conventional MRF models which do not use offset statistics. However, in comparison to recent strong structure conforming methods, i. e., approaches that apply symmetry or regularity as strong constraints, it retains simplicity and generality, and the ability to handle imperfect regularity. Nevertheless, the computational cost is kept sufficient low such that real-time modeling is possible.

In Chapter 5 we switched our focus to the classical low level pattern regularity mining task and presented a new method for partial symmetry detection. Our main contribution is to use consistent co-occurrence patterns as a novel invariant to reduce feature matching ambiguity. Based on that, a novel pattern detection algorithm is developed, which achieves robustness against low quality data. The algorithm achieved success by aggregating co-occurrence information across all building block instances and the volume they cover. We also conducted experiments to show that our detection approach outperforms previous methods on data with strong geometric variability and irregular instance distributions.

6.2 Future work

Our work is only a small step in the direction of 3D digital content creation, using symmetry derived regularity constraints. Certainly there are limitations in our approaches, and in turn these imperfections have introduced several open problems waiting to be solved.

In the following, we will discuss possible future works respectively for each individual theme. Then in Section 6.2.2 we will raise general open problems, in the hope of stimulating more work in this direction.

6.2.1 Respective discussions

Symmetry-Preserving Deformation Currently our scheme supports only finite symmetry groups. A direction of future work would be to integrate continuous symmetries, e. g., arbitrary rotations around a fixed axis. Another limitation is that editing operations which require frequencies below the sampling density are not supported. Hence, very localized edits require a dense sampling. It would be interesting to integrate non-uniform samplings. This would allow for localized edits in selected regions while preserving the real-time performance. A challenging problem is to extend the framework to other types of symmetries like intrinsic or Möbius symmetries.

3D Model Retargeting Our method is a small advance over conventional MRF models for 3D modeling. For future work, we can think of many interesting proposals. For example, how to model more complex shapes that cannot be represented by a single lattice structure. One possible solution might be to first segment the model into multiple grids, while using a secondary layer for capturing the relationship between these grids. Another interesting future path is to extend the model to non-translational mappings, e. g., rotation and scaling. Finally, it is interesting to apply offset statistics to applications such as structure from motion and shape completion.

Approximate 3D Partial Symmetry Detection In our algorithm, detection output have been stored as sparse feature correspondences with coordinate transformation matrices between each matching pair. While this is sufficient for discovering exact rigid symmetry in most of scenarios, we can not ignore the case when the dense correspondences problem brought up by increased invariance becomes dominant. At this point, we do not have an effective measure to

find high-quality dense correspondences between instances with considerable shape variability. Our preliminary experiments with standard deformable shape matching have turned out to be not very robust. This opened the very interesting question of how we can find more general notions of coherent co-occurrence, while maintaining statistical robustness and computational efficiency at the same time. Finally, it would be interesting to study invariant building blocks in more general applications, e. g., shape synthesis algorithms.

6.2.2 General outlook

User interaction The deformation application in Chapter 3 also illustrated that user interaction could serve as a very important hint of user intentions. We did not introduce too much user operations in our other work, in the purpose of reducing complexity by automating those applications. But we believe mild amount of user input could help improving our result, which is an interesting trade-off against automatic algorithms. So the study of user interaction's effectiveness and robustness could be a promising future direction.

General symmetry structure The algorithm proposed in Chapter 5 is fundamental to a lot of high-level applications, including symmetry preserving deformation. But as we pointed out in Chapter 3, the extension of our framework to very general symmetry structure needs certainly more work. Both theoretical study and practical implementation could be very challenging, but we expect this combination to be very powerful.

Completion problem A very challenging problem is to fill data into missing area for low quality scanning point cloud. Using pattern regularity is the most common idea for generating hypothesis. But when the data is noisy and incomplete, it becomes much harder to extract meaningful structure information. Another annoying fact is, even the delimitation of holes could be very hard to automate. Also because of information loss, there is no standard way of judging the quality of completion. In general, the completion problem is very related to this thesis, and worth more attentions.

General synthesis Given an arbitrary self-contained example model, it is very interesting to imagine how much potential difference could be achieved, if we try to derive meaningful variants of that model using arbitrary copies of its parts. This relates to the procedural modeling approaches, however, these

are not directly applicable because neither assembling parts nor generating rules are provided. Previous work heavily depended on precise geometry and very complicated coding systems, which are very hard to satisfy for ordinary examples. To solve this problem, it is necessary to develop an algorithm that can find the most concise representation of an input geometry. We expect this to be an open problem for a long time.

Bibliography

Aladdin (n.d.). Edushi maps for hangzhou. http://hz.edushi.com/. 2, 3

Alexa, M., Beht, J., Cohen-Or, D., Fleishman, S., Levin, D., and Silva, C. T. (2003). Computing and rendering point set surfaces. *TVCG*, 9(1):3–15. 12, 72

Au, O. K.-C., Tai, C.-L., Liu, L., and Fu, H. (2006). Dual Laplacian editing for meshes. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):386–395. 12, 28

Averkiou, M., Kim, V. G., Zheng, Y., and Mitra, N. J. (2014). Shapesynth: Parameterizing model collections for coupled shape exploration and synthesis. *Comput. Graph. Forum*, 33(2):125–134. 14

Avidan, S. and Shamir, A. (2007). Seam carving for content-aware image resizing. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA. ACM. 12, 14

Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11. 15

Berner, A., Bokeloh, M., Wand, M., Schilling, A., and Seidel, H.-P. (2008). A graph-based approach to symmetry detection. In *Symposium on Point-Based Graphics*, pages 1–8. 16

Berner, A., Bokeloh, M., Wand, M., Schilling, A., and Seidel, H.-P. (2009). Generalized intrinsic symmetry detection. Research Report MPI-I-2009-4-005, Max-Planck-Institut für Informatik. 16, 64, 65

Berner, A., Wand, M., Mitra, N. J., Mewes, D., and Seidel, H.-P. (2011). Shape analysis with subspace symmetries. *Computer Graphics Forum (Proc. Eurographics)*, 30(2):277–286. 16, 64, 65

Bhat, P., Ingram, S., and Turk, G. (2004). Geometric texture synthesis by example. In *Proceedings of the Symposium on Geometry Processing*, pages 41–44. 54

Bokeloh, M., Berner, A., p. Seidel, H., and Schilling, A. (2009). Symmetry detection using feature lines. *CGF*, pages 697–706. 16, 64, 65

Bokeloh, M., Wand, M., Koltun, V., and Seidel, H.-P. (2011a). Pattern-aware shape deformation using sliding dockers. *ACM Transactions on Graphics*, 30(6). 12, 13, 20

Bokeloh, M., Wand, M., Koltun, V., and Seidel, H.-P. (2011b). Pattern-aware shape deformation using sliding dockers. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 123:1–123:10, New York, NY, USA. ACM. 15

Bokeloh, M., Wand, M., and Seidel, H.-P. (2010a). A connection between partial symmetry and inverse procedural modeling. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 104:1–104:10, New York, NY, USA. ACM. 3

Bokeloh, M., Wand, M., and Seidel, H.-P. (2010b). A connection between partial symmetry and inverse procedural modeling. *ACM Trans. Graph.*, 29(4). 15

Bokeloh, M., Wand, M., Seidel, H.-P., and Koltun, V. (2012a). An algebraic model for parameterized shape editing. *ACM Transactions on Graphics*, 31(4). 13, 20

Bokeloh, M., Wand, M., Seidel, H.-P., and Koltun, V. (2012b). An algebraic model for parameterized shape editing. *ACM Trans. Graph.*, 31(4):78:1–78:10. 15, 45, 47, 54, 60

Botsch, M. and Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230. 12, 28, 29

Boykov, Y., Veksler, O., and Zabih, R. (2001a). Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239. 47, 49

Boykov, Y., Veksler, O., and Zabih, R. (2001b). Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1222–1239. 80

Brown, B. and Rusinkiewicz, S. (2007). Global non-rigid alignment of 3-d scans. *ACM Trans. Graph.*, 26(3). 12

Cazals, F. and Pouget, M. (2005). Estimating differential quantities using polynomial fitting of osculating jets. *Computer Aided Geometric Design*, 22(2):121 – 146. 72

Chen, K., Lai, Y.-K., Wu, Y.-X., Martin, R., and Hu, S.-M. (2014). Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Trans. Graph.*, 33(6):208:1–208:12. 17

Chen, Y. and Medioni, G. (1992). Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155. 16

Coquillart, S. (1990). Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In *Proc. Siggraph*, pages 187–196. 12

Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*, pages 886–893. 67

Demir, I., Aliaga, D. G., and Benes, B. (2015). Coupled segmentation and similarity detection for architectural models. *ACM Trans. Graph.*, 34(4):104:1–104:11. 17

Fish, N., Averkiou, M., van Kaick, O., Sorkine-Hornung, O., Cohen-Or, D., and Mitra, N. J. (2014). Meta-representation of shape families. *ACM Trans. Graph.*, 33(4):34:1–34:11. 17

Funkhouser, T., Kazhdan, M., Shilane, P., Min, P., Kiefer, W., Tal, A., Rusinkiewicz, S., and Dobkin, D. (2004). Modeling by example. In *ACM Trans. Graph. (Proc. Siggraph)*, pages 652–663. 14

Gal, R. and Cohen-Or, D. (2006). Salient geometric features for partial shape matching and similarity. *ACM Trans. on Graphics*, 25(1):130–150. 15, 64

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). 12, 20, 31

Gelfand, N. and Guibas, L. J. (2004). Shape segmentation using local slippage analysis. In *Symposium on Geometry Processing*, pages 214–223. 17

Golovinskiy, A. and Funkhouser, T. (2009). Consistent segmentation of 3D models. *Computers & Graphics (Proc. of SMI)*, 33(3):262–269. 17

Google (n.d.). Google maps. https://www.google.com/maps/. 1

Hahn, T. (2002). *International Tables for Crystallography Volume A: Space-group symmetry*. Springer Netherlands. 11

He, K. and Sun, J. (2012). Statistics of patch offsets for image completion. In *ECCV*, pages 16–29. 15, 44, 46, 49, 51

Hertzmann, A., Jacobs, C. E., Oliver, N., Curless, B., and Salesin, D. H. (2001). Image analogies. In *SIGGRAPH*, pages 327–340. 15

Herzog, R., Mewes, D., Wand, M., Guibas, L., and Seidel, H.-P. (2015). Lesss: Learned shared semantic spaces for relating multi-modal representations of 3d shapes. In *Proceedings of the Eurographics Symposium on Geometry Processing*, SGP '15, pages 141–151, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association. 65, 68

Hildebrandt, K., Schulz, C., von Tycowicz, C., and Polthier, K. (2011). Interactive surface modeling using modal analysis. *ACM Trans. Graph.*, 30(5):119:1–11. 12

Hu, R., Fan, L., and Liu, L. (2012). Co-segmentation of 3d shapes via subspace clustering. *Computer Graphics Forum (SGP 2012)*, 31(5):1703–1713. 17

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):1126–1134. 12, 28

Huang, Q., Guibas, L. J., and Mitra, N. J. (2014a). Near-regular structure discovery using linear programming. *ACM Trans. Graph.*, 33(3):23:1–23:17. 17

Huang, Q., Koltun, V., and Guibas, L. (2011). Joint shape segmentation with linear programming. *ACM Trans. Graph.*, 30:125:1–125:12. 17

Huang, Q., Wang, F., and Guibas, L. (2014b). Functional map networks for analyzing and exploring large shape collections. *ACM Trans. Graph.*, 33(4):36:1–36:11. 18

Huang, Q., Zhang, G., Gao, L., Hu, S., Bustcher, A., and Guibas, L. (2012). An optimization approach for extracting and encoding consistent maps in a shape collection. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 31(6). 16

Jacobson, A., Baran, I., Kavan, L., Popović, J., and Sorkine, O. (2012). Fast automatic skinning transformations. *ACM Trans. Graph.*, 31(4):77:1–10. 12

Jain, A., Thormählen, T., Ritschel, T., and Seidel, H.-P. (2012). Exploring Shape Variations by 3D-Model Decomposition and Part-based Recombination. *Computer Graphics Forum*, 31(2):631–640. 14, 45

Kalogerakis, E., Chaudhuri, S., Koller, D., and Koltun, V. (2012). A Probabilistic Model of Component-Based Shape Synthesis. *ACM Transactions on Graphics*, 31(4). 14

Kalogerakis, E., Hertzmann, A., and Singh, K. (2010). Learning 3d mesh segmentation and labeling. *Proc. of SIGGRAPH*, 29(3):102:1–102:12. 17

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2004a). Symmetry descriptors and 3d shape matching. In *Proc. Symposium on Geometry Processing (SGP)*. 13, 20

Kazhdan, M., Funkhouser, T., and Rusinkiewicz, S. (2004b). Symmetry descriptors and 3D shape matching. In *Symp. Geometry Processing (SGP)*, pages 115–123. 17

Kerber, J., Bokeloh, M., Wand, M., and Seidel, H.-P. (2013). Scalable symmetry detection for urban scenes. *CGF*, 32:3–15. 16, 65, 68

Kim, V., Lipman, Y., Chen, X., and Funkhouser, T. (2011). Mobiüs transformations for global intrinsic symmetry analysis. In *Symposium on Geometry Processing*, pages 1689–1700. 17

Kim, V. G., Li, W., Mitra, N. J., DiVerdi, S., and Funkhouser, T. (2012). Exploring collections of 3d models using fuzzy correspondences. *ACM Trans. Graph.*, 31(4):54:1–54:11. 16

Kolojanov, J., Bokeloh, M., Wand, M., Guibas, L., Slusallek, P., and Seidel, H.-P. (2012). Microtiles: Extracting building blocks from correspondences. In *Symposium on Geometry Processing*. 16, 82

Kraevoy, V., Sheffer, A., Shamir, A., and Cohen-Or, D. (2008). Non-homogeneous resizing of complex models. *ACM Trans. Graph.*, 27(5):111:1–111:9. 12, 20

Kurz, C., Wu, X., Wand, M., Thormählen, T., Kohli, P., and Seidel, H.-P. (2014). Symmetry-aware template deformation and fitting. *Computer Graphics Forum*, 33(6):205–219. 7, 13, 20, 23, 31, 41, 88

Laga, H., Mortara, M., and Spagnuolo, M. (2013). Geometry and context for semantic correspondences and functionality recognition in man-made 3d shapes. *ACM Trans. Graph.*, 32(5):150:1–150:16. 17

Leordeanu, M. and Hebert, M. (2005). A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482–1489. 17

Li, C. and Wand, M. (2015). Approximate translational building blocks for image decomposition and synthesis. *ACM Trans. Graph.*, 34(5):158:1–158:16. 69, 74, 75, 77

Li, C., Wand, M., Wu, X., and Seidel, H. P. (2015). Approximate 3d partial symmetry detection using co-occurrence analysis. In *3D Vision (3DV), 2015 International Conference on*, pages 425–433. 6, 7, 88

Lin, J., Cohen-Or, D., Zhang, H., Liang, C., Sharf, A., Deussen, O., and Chen, B. (2011a). Structure-preserving retargeting of irregular 3D architecture. *ACM Trans. Graph.*, 30(6). 12

Lin, J., Cohen-Or, D., Zhang, H., Liang, C., Sharf, A., Deussen, O., and Chen, B. (2011b). Structure-preserving retargeting of irregular 3d architecture. *ACM Trans. Graph.*, 30(6):183:1–183:10. 15, 45

Lin, J., Cohen-Or, D., Zhang, H., Liang, C., Sharf, A., Deussen, O., and Chen, B. (2011c). Structure-preserving retargeting of irregular 3d architecture. *ACM Trans. Graph.*, 30(6):183:1–183:10. 54

Lipman, Y., Chen, X., Daubechies, I., and Funkhouser, T. (2010). Symmetry factored embedding and distance. *ACM Trans. Graph.*, 29:103:1–12. 13, 16, 20, 64, 65, 74, 76, 77, 80

Liu, J. and Liu, Y. (2013). Grasp recurring patterns from a single view. In *CVPR*, pages 2003–2010. 74, 77

Liu, T., Chaudhuri, S., Kim, V. G., Huang, Q.-X., Mitra, N. J., and Funkhouser, T. (2014). Creating consistent scene graphs using a probabilistic grammar. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 33(6). 17

Liu, Y., Prabhakaran, B., and Guo, X. (2012). Point-based manifold harmonics. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1693–1703. 28

Loy, G. and Eklundh, J.-O. (2006). Detecting symmetry and symmetric constellations of features. In *Proc. European Conf. Comp. Vision ECCV*, pages 508–521. 16

Martinet, A., Soler, C., Holzschuch, N., and Sillion, F. (2006). Accurate detection of symmetries in 3d shapes. *ACM Trans. on Graphics*, 25(2):439 – 464. 17

Mattausch, O., Panozzo, D., Mura, C., Sorkine-Hornung, O., and Pajarola, R. (2014). Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum*, 33(2):11–21. 16

Merrell, P. and Manocha, D. (2011). Model synthesis: A general procedural modeling algorithm. *IEEE TVCG*, 17(6):715–728. 15, 54

Merrell, P. C. (2009). Model synthesis. *PhD Thesis, Stanford University*. 15

Mitra, N., Wand, M., Zhang, H. R., Cohen-Or, D., Kim, V., and Huang, Q.-X. (2013a). Structure-aware shape processing. In *SIGGRAPH Asia 2013 Courses*, SA '13, pages 1:1–1:20, New York, NY, USA. ACM. 3, 63

Mitra, N. J., Bronstein, A., and Bronstein, M. (2010). Intrinsic regularity detection in 3d geometry. In *Proc. European Conf. Comp. Vision ECCV*, pages 398–410. 17

Mitra, N. J., Guibas, L., and Pauly, M. (2006a). Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (SIGGRAPH)*, 25(3):560–568. 15

Mitra, N. J., Guibas, L., and Pauly, M. (2006b). Partial and approximate symmetry detection for 3d geometry. *Proc. of SIGGRAPH*, 25(3):560–568. 16

Mitra, N. J., Guibas, L. J., and Pauly, M. (2006c). Partial and approximate symmetry detection for 3d geometry. In *SIGGRAPH*, pages 560–568. 15, 64, 65

Mitra, N. J., Guibas, L. J., and Pauly, M. (2007). Symmetrization. *Proc. of SIGGRAPH*, 26(3):63:1–63:8. 16

Mitra, N. J., Pauly, M., Wand, M., and Ceylan, D. (2013b). Symmetry in 3d geometry: Extraction and applications. *Computer Graphics Forum*, 32(6):1–23. 11, 15, 22, 63

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization (2nd edition)*. Springer. 29

Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., and Guibas, L. (2012). Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4). 17

Ovsjanikov, M., Mérigot, Q., Pătrăucean, V., and Guibas, L. (2013). Shape matching via quotient spaces. *Computer Graphics Forum (Proceedings of SGP)*. 13, 20

Ovsjanikov, M., Sun, J., and Guibas, L. (2008). Global intrinsic symmetries of shapes. In *Symp. Geometry Processing (SGP)*, pages 1341–1348. 17, 64

Parish, Y. I. H. and Müller, P. (2001). Procedural modeling of cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 301–308, New York, NY, USA. ACM. 3

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H., and Guibas, L. (2008a). Discovering structural regularity in 3D geometry. *Proc. of SIGGRAPH*, 27(3):43:1–43:11. 16, 64, 65

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H., and Guibas, L. J. (2008b). Discovering structural regularity in 3d geometry. *ACM Trans. Graph.*, 27(3). 15, 47

Podolak, J., Shilane, P., Golovinskiy, A., Rusinkiewicz, S., and Funkhouser, T. (2006). A planar-reflective symmetry transform for 3D shapes. *Proc. of SIGGRAPH*, 25(3):549–559. 15, 16, 64, 65

Pritch, Y., Kav-Venaki, E., and Peleg, S. (2009). Shift-map image editing. In *ICCV*, pages 151–158. 15, 44

Rahmani, H., Mahmood, A., Huynh, D. Q., and Mian, A. (2014). Hopc: Histogram of oriented principal components of 3d pointclouds for action recognition. In *ECCV*. 65, 68

Raviv, D., Bronstein, A. M., Bronstein, M. M., and Kimmel, R. (2010). Full and partial symmetries of non-rigid shapes. *Int. J. Comput. Vision*, 89(1):18–39. 17

Rusu, R. B., Blodow, N., and Beetz, M. (2009). Fast point feature histograms (fpfh) for 3d registration. In *ICRA*, pages 3212–3217. 51

Seber, G. A. (1984). *Multivariate Observations*. Wiley. 73

Sederberg, T. W. and Parry, S. R. (1986). Free-form deformation of solid geometric models. In *SIGGRAPH*. 12

Shen, C.-H., Fu, H., Chen, K., and Hu, S.-M. (2012). Structure recovery by part assembly. *ACM Trans. Graph.*, 31(6):180:1–180:11. 79

Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., and Cohen-Or, D. (2011). Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. *ACM Trans. on Graphics (Proc. SIGGRAPH Asia)*, 30(6):126:1–126:10. 17

Simakov, D., Caspi, Y., Shechtman, E., and Irani, M. (2008). Summarizing visual data using bidirectional similarity. In *CVPR*. 15

Simari, P., Kalogerakis, E., and Singh, K. (2006). Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Symposium on Geometry Processing*, pages 111–119. 15, 17

Song, S. and Xiao, J. (2014). Sliding shapes for 3d object detection in depth images. In *ECCV*. 65, 68, 75

Sorkine, O. and Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Proceedings of Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 109–116. 12, 30

Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Symposium on Geometry processing*, pages 175–184. 12

Talton, J. O., Lou, Y., Lesser, S., Duke, J., Měch, R., and Koltun, V. (2011). Metropolis procedural modeling. *ACM Trans. Graph.*, 30(2):11:1–11:14. 54

Tam, G. K., Martin, R. R., Rosin, P. L., and Lai, Y.-K. (2014). An efficient approach to correspondences between multiple non-rigid parts. *Computer Graphics Forum (SGP 2014)*, 33(5):137–146. 17

Terzopoulos, D., Platt, J., Barr, A., and Fleischer, K. (1987). Elastically deformable models. In *Proceedings of SIGGRAPH*, pages 205–214, New York, NY, USA. ACM. 12

Tevs, A., Huang, Q., Wand, M., Seidel, H.-P., and Guibas, L. (2014a). Relating shapes via geometric symmetries and regularities. *ACM Trans. Graph.*, 33(4):119:1–119:12. 11

Tevs, A., Huang, Q., Wand, M., Seidel, H.-P., and Guibas, L. (2014b). Relating shapes via geometric symmetries and regularities. *ACM Trans. Graph.*, 33(4):119:1–12. 22, 30

Tombari, F., Salti, S., and di Stefano, L. (2010). Unique signatures of histograms for local surface description. In *ECCV*, pages 356–369. 46, 49

Turk, G. (2001). Texture synthesis on surfaces. In *Proceedings of SIGGRAPH*, pages 347–354. 54

Vallet, B., Brédif, M., Serna, A., Marcotegui, B., and Paparoditis, N. (2015). Terramobilita/iqmulus urban point cloud analysis benchmark. *Comput. Graph.*, 49(C):126–133. 74

von Tycowicz, C., Schulz, C., Seidel, H.-P., and Hildebrandt, K. (2013). An efficient construction of reduced deformable objects. *ACM Trans. Graph.*, 32(6):213:1–10. 12

Wang, H., Simari, P., Su, Z., and Zhang, H. (2014). Spectral global intrinsic symmetry invariant functions. In *Graphics Interface*. 13, 20

Wang, Y., Xu, K., Li, J., Zhang, H., Shamir, A., Liu, L., Cheng, Z., and Xiong, Y. (2011). Symmetry hierarchy of man-made objects. *Computer Graphics Forum*, 30(2). 13, 32, 45

Wei, L.-Y., Lefebvre, S., Kwatra, V., and Turk, G. (2009). State of the art in example-based texture synthesis. Eurographics STARs. 15, 43

Welch, W. and Witkin, A. (1992). Variational surface modeling. In *Computer Graphics (Proceedings Siggraph)*, volume 26. 12

Wu, X., Li, C., Wand, M., Hildebrandt, K., Jansen, S., and Seidel, H. (2014a). 3d model retargeting using offset statistics. *IEEE 3DV*. 6, 7, 88

Wu, X., Wand, M., Hildebrandt, K., Kohli, P., and Seidel, H.-P. (2014b). Real-time symmetry-preserving deformation. *Comput. Graph. Forum*, 33(7):229–238. 6, 7, 88

Xu, K., Chen, K., Fu, H., Sun, W.-L., and Hu, S.-M. (2013). Sketch2scene: Sketch-based co-retrieval and co-placement of 3d models. *ACM Transactions on Graphics*, 32(4):123:1–123:12. 17

Xu, K., Zhang, H., Cohen-Or, D., and Chen, B. (2012a). Fit and diverse: Set evolution for inspiring 3d shape galleries. *ACM Trans. Graph.*, 31(4):57:1–57:10. 14

Xu, K., Zhang, H., Jiang, W., Dyer, R., Cheng, Z., Liu, L., and Chen, B. (2012b). Multi-scale partial intrinsic symmetry detection. *Proc. of SIGGRAPH Asia*, 31(6). 17

Xu, K., Zhang, H., Jiang, W., Dyer, R., Cheng, Z., Liu, L., and Chen, B. (2012c). Multi-scale partial intrinsic symmetry detection. *ACM Trans. Graph.*, 31(6):181:1–181:11. 64

Xu, K., Zhang, H., Tagliasacchi, A., Liu, L., Li, G., Meng, M., and Xiong, Y. (2009a). Partial intrinsic reflectional symmetry of 3d shapes. *ACM Transactions on Graphics, (Proceedings SIGGRAPH Asia 2009)*, 28(5):138:1–138:10. 12, 20

Xu, K., Zhang, H., Tagliasacchi, A., Liu, L., Li, G., Meng, M., and Xiong, Y. (2009b). Partial intrinsic reflectional symmetry of 3D shapes. *Proc. of SIGGRAPH*, 28(5):138:1–138:10. 17

Zheng, Y., Cohen-Or, D., Averkiou, M., and Mitra, N. J. (2014). Recurring part arrangements in shape collections. *Computer Graphics Forum (Special issue of Eurographics 2014)*. 17

Zheng, Y., Cohen-Or, D., and Mitra, N. J. (2013). *Smart Variations*: Functional substructures for part compatibility. *Comput. Graph. Forum*, 32(2):195–204. 14, 45

Zheng, Y., Fu, H., Cohen-Or, D., Au, O. K.-C., and Tai, C.-L. (2011). Component-wise controllers for structure-preserving shape manipulation. *Computer Graphics Forum*, 30(2):563–572. 13, 20

Zhou, K., Huang, X., Wang, X., Tong, Y., Desbrun, M., Guo, B., and Shum, H.-Y. (2006). Mesh quilting for geometric texture synthesis. *ACM Trans. Graph.*, 25(3):690–697. 54