Data-driven Modelling of Shape Structure

Melinos Averkiou

Department of Computer Science

University College London

This dissertation is submitted for the degree of

Doctor of Philosophy

August 2015

Declaration

I, Melinos Averkiou confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Melinos Averkiou
August 2015

Abstract

In recent years, the study of shape structure has shown great promise, by taking steps towards exposing shape semantics and functionality to algorithms spanning a wide range of areas in computer graphics and vision. By shape structure, we refer to the set of parts that make a shape, the relations between these parts, and the ways in which they correspond and vary between shapes of the same family. These developments have been largely driven by the abundance of 3D data, with collections of 3D models becoming increasingly prominent and websites such as Trimble 3D Warehouse offering millions of free 3D models to the public. The ability to use large amounts of data inside these shape collections for discovering shape structure has made novel approaches to acquisition, modelling, fabrication, and recognition of 3D objects possible. Discovering and modelling the structure of shapes using such data is therefore of great importance.

In this thesis we address the problem of discovering and modelling shape structure from large, diverse and unorganized shape collections. Our hypothesis is that by using the large amounts of data inside such shape collections we can discover and model shape structure, and thus use such information to enable structure-aware tools for 3D modelling, including shape exploration, synthesis and editing.

We make three key contributions. First, we propose an efficient algorithm for co-aligning large and diverse collections of shapes, to tackle the first challenge in detecting shape structure, which is to place shapes in a common coordinate frame. Then, we introduce a method to parameterize shapes in terms of locations and sizes of their parts, and we demonstrate its application to concurrently exploring a shape collection and synthesizing new shapes. Finally, we define a meta-representation for a shape family, which models the relations of shape parts to capture the main geometric characteristics of the family, and we demonstrate how it can be used to explore shape collections and intelligently edit shapes.

Acknowledgements

First of all I would like to express my deepest gratitude to my advisor, Niloy J. Mitra, for his invaluable advice, guidance and support throughout my studies. During the past four years he has been a true mentor, sharing his knowledge and expertise, patiently teaching me how to conduct and communicate research, and keeping me focused on my goals. Most importantly, he has taught me to always aim high and reach out of my comfort zone.

I am truly grateful to all my collaborators and co-authors: Vladimir G. Kim, Youyi Zheng, Oliver van Kaick, Noa Fish, Daniel Cohen-Or, and Olga Sorkine-Hornung. Without their help and contribution, this work would not have been possible. I am also indebted to Leonidas Guibas, the members of his group at Stanford, and Qi-Xing Huang in particular, for giving me the opportunity to spend time in their lab, for their support, collaboration and hospitality.

I would like to express my gratitude to all members, past and present, of the Smart Geometry Processing group in UCL: Bongjin Koo, Aron Monszpart, Moos Hueting, James Hennessey, Tuanfeng Wang, Nicolas Mellado and Martin Kilian; as well as all members of the VECG group in UCL. For all the fruitful discussions, arguments, motivation and help they have provided, I am very thankful. I would also like to thank Yiorgos Chrysanthou, for introducing me to Niloy, encouraging me to pursue my research dreams and providing me with support and advice throughout my studies.

My research has been supported by EPSRC, the Rabin Ezra Scholarship Trust, the A. G. Leventis Foundation and the Cyprus State Scholarship Foundation. I acknowledge and express my immense gratitude for their support.

Last but not least, I would like to thank my friends and family. To my friends in London: Michalis, Loizos, Fotis, Andreas, Christos, Theodosis, Alexis; and my friends back home: Nikoletta, Panayiotis, Sofronis, Pavlos, Stelios, Elena, Nectarios; thanks for the countless memories, the fun times and the serious times. To my family, thank you for giving me a good education and for teaching me what matters in life. Special thanks to the most important person in my life, Christina, for her patience throughout these years, for supporting and encouraging me in every step, and for always being by my side. Thanks everyone, this is for you.

To Christina.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Geometry processing has grown into a significant research area concerned with the design of efficient algorithms for the acquisition, reconstruction, analysis, modelling, manipulation, and fabrication of complex 3D models [9]. Fundamental problems in geometry processing, such as segmentation, smoothing, remeshing, deformation and correspondence [9] have been well studied in the context of single shapes or pairs of shapes.

However, the explosion in the amount of 3D content freely available in online repositories such as Trimble 3D warehouse provides new challenges in such fundamental problems, as well as opportunities for investigating new shape properties, by exploring consistency across multiple semantically-similar shapes. These semantically-similar shapes, potentially very different geometrically and topologically, form what we now call shape families, or shape collections. For example, a family, or collection, of airplanes can be seen in Figure 1.1. It contains various different types of airplanes, such as large carrier jets or fighter jets, with one, two, or even four engines, among others. They are geometrically different, but they all belong to a family of airplanes.

This great increase in the size of online 3D repositories has been largely driven by a sharp rise in the amount of cheap geometry acquisition devices and modelling tools. Such massive shape repositories now contain millions of freely available 3D models and have given a push to data-driven methods for discovering shape structure, making novel approaches to acquisition [56, 57], modelling [15], fabrication [77], and recognition of 3D objects [81, 104] possible.

Shape structure has thus become an emerging topic of research in geometry processing over the past few years. It has shown potential in deriving semantics and function

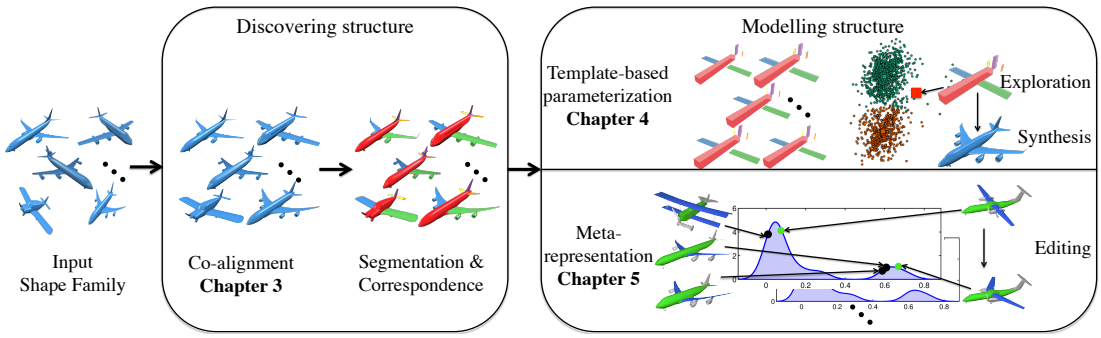


Figure 1.1: The process of discovering and modelling structure from a shape collection. Starting from an unorganized shape family, we first need to co-align shapes (see Chapter 3) in order to then solve for segmentation and correspondence between them. We can then model such structure, for example using a template-based parameterization of shapes (see Chapter 4) or a set of geometric distributions (see Chapter 5). Different shape processing applications such as synthesis, exploration and editing can be powered by having a representation of shape structure.

from pure geometry and exposing such rich information to algorithms that have traditionally operated on low-level geometry. Structure, in the case of shapes, is used to describe the set of meaningful parts which compose a shape and the mathematical relations of these parts, such as various types of symmetry, parallelism, concentricity, co-planarity and others [70]. Extending this notion to a family of shapes sharing common semantics, we additionally seek to discover and describe the ways in which these parts correspond and vary between shapes.

The common observation behind data-driven structure-aware shape processing methods is that by learning a model of structure for a shape or family of shapes, we can use such information to reason about shapes at a higher level of abstraction, departing from the existing focus on low-level geometry. This allows, for example, surface reconstruction by detecting objects [71] and reusing existing shape parts [83] rather than pure point clouds, sampling a generative model to synthesize novel shapes [48, 100] rather than creating shapes from scratch, and designing functional, ready to print models from existing primitives [59] rather than worrying about mechanical properties. This observation seems to hold well for man-made shapes, which have been the focus of most research effort so far, and are the focus of this thesis.

Such recent advancements illustrate that all stages of the content creation pipeline are significantly accelerated by harvesting models of structure. Accelerating content creation results in cost reduction and increases the size of online shape repositories, providing more data for learning structure, thus creating a feedback loop of continuous development. As shape collections continue to grow, discovering and modelling structure from existing data is both important and challenging.

A number of these challenges are related to the manner in which shape repositories are constructed. First, online shape repositories are typically unorganized and contain loose and noisy semantic tags for each object. In the example of Figure 1.1, to collect a set of airplanes, one would have to use text-based search in a shape repository to retrieve a set of models that are tagged similarly to the text query. The results of such a search would most likely include outliers that would need to be cleaned up, as many models are not tagged properly. Second, models in such collections are not consistently oriented. Most of them have consistent up vectors, but, as Figure 1.1 illustrates, they point in different directions. Third, these collections have no information on how shapes relate to each other or how they vary from one another.

Manually resolving these problems is not scalable, as these collections continue to grow. Similar challenges have been faced by early data-driven image processing algorithms, when millions of images started flooding the web. Advances in that area have now been used to organize, explore and export correlations and semantics from the millions of images online. Inspired by such advances, we would like to overcome these challenges in deriving and using structure from shape collections.

In this thesis we address the problem of discovering and modelling shape structure from large, diverse and unorganized shape collections. 1.1). Our hypothesis is that by taking advantage of the large amounts of data inside such shape collections, we can discover and model shape structure, thereby allowing us to use such information to enable several structure-aware shape processing methods, with an impact in exploration, synthesis, editing and deformation of shapes (see Figure 1.1).

Our input is a collection of semantically-related man-made shapes, for example a collection of airplanes (see Figure 1.1), which have a consistent up vector. One can retrieve such a collection from an online repository automatically using text-based search. If the models in the collection have no consistent up vector, one can use the upright orientation method of Fu et al. [29] to consistently orient them in the upward direction. By man-made, we mean any shape representing an object built by humans, excluding complicated structures such as buildings. This includes anything from small objects like cups, to furniture, mechanical devices like bikes and cars, and even helicopters and airplanes. We make no assumptions on the quality of the models or the existence of any information on how they relate to each other. This means that our input contains low-polygon, non-manifold shapes, or even polygon soups, as well as professional-quality models. Finally, shapes inside the same collection can vary widely in geometry and topology, but they still exhibit structural similarities related to their semantics and functionality, which we aim to discover.

We focus less on discovering structure and more on modelling the structure from collections of shapes. Specifically, first we propose an algorithm to rigidly align a set of shapes, which is the first step in discovering structure. Then, we introduce a method to parameterize shapes based on box templates, and finally, we suggest a model of shape part relations in a shape family, as an attempt in modelling structure. We demonstrate how our results can be used for various structure-aware geometry processing tools, including exploration of shape collections, shape synthesis, and intelligent shape editing, among others.

1.2 Contributions

In our work we make three key contributions in discovering and modelling shape structure. Figure 1.1 gives a high-level overview of the process of discovering, modelling and using shape structure, noting our contributions.

Efficient co-alignment of shape collections. Deriving structure from a collection of shapes typically requires solving for correspondences between shape surfaces and compatible segmentations of shapes into parts. Solving for such correspondences and segmentations first requires bringing shapes to a common coordinate frame, thus, alignment is the first step to discovering structure. We devise a fast algorithm for rigidly co-aligning a set of shapes belonging to the same family. In contrast to state of the art methods [41] that uniformly sample the rotation space in search for the best alignment that minimizes distances between all shape surfaces, we observe that only a small set of rotations are responsible for erroneous alignments, thus greatly reducing the amount of candidate alignments we need to examine per shape. Using rotational self-symmetries of shapes to pinpoint this small set of rotations allows us to develop an algorithm that is not only several times faster than uniform sampling, but is also more accurate on average.

We evaluate our algorithm on a large benchmark with ten different shape families by comparing to state-of-the-art methods and we report improvements in both efficiency and alignment accuracy. Chapter 3 provides details of our alignment algorithm.

Template-based shape parameterization. With the growing size of shape repositories, it is becoming more challenging to explore and understand the kinds of shapes inside such collections. At the same time, understanding these variations and exploring these datasets is becoming more necessary for various data-driven shape processing methods like part-based synthesis, which enables non-expert users to synthesize novel shapes by assembling parts extracted from model databases. We propose a way to

parameterize shapes belonging to the same family, based on positions and sizes of template boxes fit to each shape. By jointly analyzing the arrangements and sizes of parts across models, we are able to hierarchically embed the models into low-dimensional spaces.

We then demonstrate how a user can utilize the parameterization to explore the shape family by clicking in different areas of the embedded shape space and by selecting groups of shapes to zoom on specific shape clusters. More importantly, any point in the embedded space can be lifted to an arrangement of parts to provide an abstracted view of possible shape variations at that point. The abstraction can further be realized by appropriately deforming parts from neighboring models to produce synthesized geometry.

We evaluate our method's ability to explore and synthesize novel shapes through a large user study on four different shape families, and we report the plausibility and diversity of the synthesized results as judged by human participants, compared to results achieved with state-of-the-art methods. We also evaluate the quality of our shape space embedding via a reachability experiment, where we report how our parameterization is affected by excluding random shapes. Our experiments indicate that our explorative shape synthesis paradigm, combining shape exploration with synthesis, can be used by people with virtually no 3D modelling experience to rapidly generate plausible and diverse shapes comparable to the previous methods. Chapter 4 provides details of our template-based parameterization.

Meta-representation of shape families. Taking our template-based shape abstraction one step further, we introduce a model of structure to characterize a family of shapes. We call this model a meta-representation, that represents the geometric *essence* of a family of shapes. The meta-representation learns the configurations of shape parts that are common across the family, and encapsulates this knowledge with a system of geometric distributions that encode relative arrangements of parts. Thus, instead of predefined priors, what characterizes a shape family is directly learned from the set of input shapes. The meta-representation is constructed from a set of co-segmented shapes with known correspondence, relying on previously discovered shape structure.

It can then be used in several applications where we seek to preserve the identity of the shapes as members of the family. We demonstrate applications of the meta-representation in exploration of shape repositories, where interesting shape configurations can be examined in the set; guided editing, where models can be edited while maintaining their familial traits; and coupled editing, where several shapes can be collectively deformed by directly manipulating the distributions in the meta-representation.

We evaluate our meta-representation in intelligently editing shapes and exploring shape collections on several shape families and illustrate a multitude of cases where our smart editing tool outperforms state-of-the-art methods. Chapter 5 provides details of our meta-representation.

List of publications. Parts of this thesis have been published as follows:

- The efficient co-alignment algorithm in Chapter 3 has been published in:
  AVERKIOU, M., KIM, V. G., AND MITRA, N. J. Autocorrelation Descriptor for Efficient Co-alignment of 3D Model Collections. *Computer Graphics Forum 34*, 8 (2015). In Press.

- The template-based shape parameterization in Chapter 4 has been published in:
  AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis. *Computer Graphics Forum (Eurographics) 33*, 2 (2014), 125–134.

- The meta-representation of shape families in Chapter 5 has been published in:
  FISH*, N., AVERKIOU*, M., VAN KAICK, O., SORKINE-HORNUNG, O., COHEN-OR, D., AND MITRA, N. J. Meta-representation of Shape Families. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 34:1–34:11. *joint 1st authors.

Additionally, the following publications are directly connected to the work presented in this thesis:

- The template-based shape parameterization in Chapter 4 has been used to generate large amounts of synthetic 3D content to train object proposal estimators for depth images, as published in:
  ZHENG, S., PRISACARIU, V. A., AVERKIOU, M., CHENG, M.-M., MITRA, N. J., SHOTTON, J., TORR, P. H., AND ROTHER, C. Object Proposals Estimation in Depth Image Using Compact 3D Shape Manifolds. In *Proceedings of German Conference on Pattern Recognition* (2015). In press.

- An unsupervised algorithm for the detection of recurring part arrangements and shape co-segmentation, which is the second step in discovering shape structure (see Figure 1.1) has been published in:
  ZHENG, Y., COHEN-OR, D., AVERKIOU, M., AND MITRA, N. J. Recurring Part Arrangements in Shape Collections. *Computer Graphics Forum (Eurographics) 33*, 2 (2014), 115–124.

Chapter 2

Related Work

Shape structure can be defined as the set of meaningful parts which compose a shape and the mathematical relations of these parts, such as various types of symmetry, parallelism, concentricity, co-planarity and others [70]. Extending this notion to families of shapes, we would also like to know how these parts correspond and vary between shapes. Extracting structure from a shape collection therefore amounts to discovering a common set of parts, their correspondence over shapes in the collection, and the relations of these parts. Armed with a model of structure, structure-aware shape processing tools attempt to preserve it by minimizing the deviation from the relations discovered, which are formulated as constraints. Methods for shape exploration, editing and synthesis, among others, benefit from such a model. In the following, we review related work in discovering and modelling structure, as well as using structure for shape processing, noting our contributions in each stage.

2.1 Structure Discovery and Modelling

Discovering a decomposition of a shape into meaningful parts is the classic segmentation task [79]. Ensuring that these parts are consistent between the shapes in a collection requires establishing correspondence at the level of parts or points on a shape surface [93]. Segmentation and correspondence are usually jointly solved for in recent co-analysis methods. We discuss the main steps for discovering and modelling structure, starting from alignment which is the first step required by co-analysis of shape collections.

2.1.1 Shape Alignment

Solving for correspondence and segmentation requires bringing shapes to a common coordinate frame, thus, alignment is the first step to discovering shape structure. Alignment

is also a common problem encountered in shape reconstruction [6], matching [43, 54], exploration [4, 31, 53], automatic synthesis [30, 48], and classification [41]. In shape acquisition, local alignment, or registration, is a necessary step to building a 3D model of a scanned object [18], since scans are always partial and need to be pairwise-aligned in order to be brought into a common coordinate frame. In shape retrieval [50], registration is the first step in matching two shapes, since it aims to find the optimal transformation at which the shapes can be compared for similarity.

In its general form, given a set of 3D shapes, the problem is to find a set of transformations that optimally position all shapes with respect to one another. Quantifying this optimal positioning has proved difficult in general, since first one has to define a notion of distance between shapes. Thus, a variety of techniques have been developed to address the problem. First, we discuss methods for aligning pairs of shapes, which could be used to align all models to a randomly picked reference. Such methods can be influenced by the choice of the reference shape. Then, we discuss recent approaches that tackle this problem by considering multiple shapes simultaneously, ensuring that all pairs are aligned well.

Pairwise alignment. Typically, pairwise alignment algorithms fix one shape and search for a transformation for another one that minimizes the distance between them. Horn et al. [37] demonstrate that optimal translation and scale can be found by aligning centroids of objects and normalizing the variance in distance of points to their center of mass. Therefore, most techniques focus on searching for the optimal rotation that aligns objects. Exhaustive search is computationally prohibitive since the space of possible rotations is large. Thus, approaches for pairwise shape alignment have attempted to reduce the search space using ideas that fall in three main categories, namely, Principal Component Analysis [47], voting, and Iterative Closest Point [6]. Various descriptors have also been proposed in the area of shape retrieval to deal with the alignment of shapes.

Principal Component Analysis. Methods based on Principal Component Analysis (PCA) [47] for alignment have been favoured because of their simplicity and low computational complexity. This however comes at the cost of accuracy since these methods are good at aligning very similar shapes, but fail to align shapes with even slight dissimilarities. PCA-based methods involve computing the eigen decomposition of the covariance matrix of a shape represented as a mesh or point set. The rotation that aligns a pair of shapes is then chosen as the one that maps their eigenvectors. The size of the mesh faces needs to be taken into account when calculating the covariance matrix in order to produce more accurate results, as indicated by Vranic et al. [94].

PCA alignment often produces inaccurate results for two reasons. The first is the inherent ambiguity in choosing the direction of the eigenvector to map to. Flipping the direction of an eigenvector preserves its property of being a valid eigenvector, therefore there are two possibilities for mapping to each of the three eigenvectors, and a total of eight possibilities of aligning a shape to its eigenvectors. The second reason is that even when two shapes are aligned using their principal axes, they may still not be optimally aligned, as even small geometrical dissimilarities in two shapes can cause their principal axes to differ very much, as illustrated by Funkhouser et al. [31] and Kazhdan et al. [50].

Early approaches have tried to tackle the first problem, by solving the axis ambiguity using a simple 'heavier axis' flip [25]. In this method, the number of points on each side of the centre of the shape is counted. The shape's calculated principal axes are then flipped accordingly so that the shape is 'heavier' on the positive side. Later work by Kazhdan et al. [50] suggested that the 'heavier axis' flip method is unstable because most shapes have a nearly equal amount of points on both sides of an axis. An alternative way of solving the axes ambiguity problem by an efficient axis search method was proposed in the same work [50]. This is essentially an exhaustive search over the eight possible axis mappings, accelerated by calculating simultaneous dot products of spherical harmonics components. A more recent approach by Chaouch et al. [13] recognized both weaknesses of the PCA alignment methods and proposed a modification where only the most appropriate principal axes are selected for mapping to, according to a planar reflection criterion. In the worst case, only one of the principal axes may be selected, and then a local translation-invariant cost is used to find the rest of the axes.

Another way to reduce the search space is to detect upward orientation first [29], and then only look for rotations around the up axis [41, 54]. In Fu et al. [29], the upright orientation of man-made shapes can be accurately found by recognizing that they are designed to stand on flat surfaces, thus reducing the rotation search space to a small set of candidates, according to the geometric properties related to this functionality.

Voting Methods. If the correspondence of sets of points on both shapes is known, a rigid transformation can be computed which minimizes the distance of these points [23], essentially aligning the two shapes. The problem is that this correspondence is never known in advance, and is actually part of the structure discovery problem.

Voting methods, such as geometric hashing [62] or the generalized Hough transform [5] quantize the transformation space and attempt to approximate the correct transformation by trying different possible correspondence options. They repeatedly select pairs of triplets from the two shapes, compute the rigid transformation that aligns

them and cast a vote in the entry corresponding to this transformation. At the end, the transformation with the most votes is deemed to be the optimal one. These methods are quite expensive because they sample all triplets of points from both shapes.

Iterative Closest Point. A different class of algorithms seeks to estimate and refine the correspondence of sets of points on both shapes, thereby aligning the shapes. The most well known of these, a local alignment solution for shapes already in close proximity and pose, is the classic Iterative Closest Point (ICP) algorithm [6, 18]. This algorithm aims to find the optimal rigid transformation by repeatedly choosing subsets of points on the two shapes' surface, establishing correspondences between them and refining the current guess for the transformation. Rusinkiewicz et al. [74] give a thorough review of the variants of this algorithm that have been introduced over the years and describe a combination of methods that gives good alignment results. Funkhouser et al. [30] used a volumetric ICP method to align new parts with existing shapes before stitching them. The ICP method was compared to the previous PCA 'heavier axis' flip method [25], and was found to outperform it. In a different effort from Sharf et al. [82], a variant of the ICP method, dubbed soft-ICP was used to attach new parts to an existing model. This method calculates a different rigid transformation for aligning every pair of correspondent point neighbourhoods in the overlapping regions of the new part and the existing model, resulting in a non-rigid transformation for blending the new part into the model.

The main issue with ICP approaches is that they are sensitive to the point selection strategy and especially to the way correspondences are established between selected points. For this reason, good initial estimates of the relative rigid transformation between shapes are normally required for this family of methods to produce accurate alignments. More recent work from Gelfand et al. [33] has relied on computing point descriptors and selecting points according to the uniqueness of their descriptor, which in turn helps find more optimal correspondences between two shapes. There are several other variants of the ICP algorithm, including methods that solve for non-rigid deformation [12] or those that leverage intrinsic geometry [40].

Rotation-invariant descriptors. Some methods have tried to describe shapes using various rotation-invariant descriptors so that pose information is factored out. One well-known approach is that of using spherical harmonics to describe shapes in a rotation invariant way [52]. A simple approximation method was used by Chen et al. [17], who defined a 'lightfield' shape descriptor based on the Fourier coefficients extracted from the renderings of a shape from various camera positions on a rotating dodecahedron. Their approach implies that the best alignment of two shapes will be

found at the minimum distance between their descriptors. A different approach from Kazhdan et al. [51], used spherical harmonics transformations of shape descriptors based on spherical functions, along with assumptions on the axial symmetry of objects to accelerate the search for the best alignment of two shapes, by leaving out much of the space of rotations.

Co-alignment of shape collections. When aligning a collection of 3D models, considering all pairs independently is highly inefficient. To reduce the matching complexity, Huber et al. [45] introduced the idea of using a shape graph to align multiple scans of the same object in a common coordinate frame. However, when considering a heterogeneous collection of shapes, picking a single alignment reference becomes challenging.

To address this challenge, Huang et al. [41] and Zheng et al. [106] pose the co-alignment problem as a labeling problem in the alignment space. The solution space is uniformly sampled to generate candidate labels, and the cumulative pairwise alignment error is evaluated against possible labelings to pick the best co-alignment solution. Choosing the right label for every shape such that the sum of distances between all shapes is minimized is equivalent to a quadratic assignment problem, which is known to be NP-hard to solve. Various approximations exist for solving such problems, posed as Markov Random Fields (MRF) [64, 65]. The shape graph notion is also used by Kim et al. [53, 54] for co-aligning shapes and solving for correspondence and segmentation, while at the same time, a number of other methods [42, 43, 72] propose improving graph consistency as a means to improve shape correspondence.

Contribution. We would like to be able to align large and diverse collections of shapes, in an efficient and robust manner. Pairwise shape alignment methods are used as part of recent co-alignment methods, but on their own, they are not applicable to shape collections, because they are not designed to handle diverse shapes with wide geometric variations. Many of these methods are dedicated to local refinement of initial alignments for similar shapes, while our focus is on global co-alignment of shape collections, typically containing large in-class variations. Thus, trivially extending them to shape collections will lead to inaccuracies when dissimilar pairs of shapes are aligned.

Recent co-alignment methods on the other hand promise more accurate results by posing the problem as finding a global solution that minimizes distances between all shapes together, and carefully choosing which pairs shapes are to be considered using the shape graph. However, they are computationally expensive, since they uniformly sample the solution space for each space. We observe that for most shapes this uniform

sampling leads to a large number of unnecessary computations, since many alignments yield very high distance between shapes. Typically, only a few relative angles lead to local minima in the pairwise distances and need to be resolved by the joint matching. Our key observation is that these minima arise due to rotational near-symmetries of shapes, and thus can be estimated for each shape independently. For instance, bikes have two rotational near-symmetries at 0 and 180 degrees around their up vector, and this is the main source of confusion for most co-alignment algorithms. We therefore propose an efficient algorithm that generates the same quality of results with a significantly smaller computational overhead. Chapter 3 gives details on our method.

2.1.2 Shape Co-analysis

In this section we give an overview of work in segmentation and pairwise correspondence as a means of discovering structure, and focus our attention on collective analysis of shape families.

Segmenting a mesh into a set of parts is a classic topic in geometry processing, with applications in mesh parameterization, matching and editing. It can be formulated as an optimization problem, in which, given a set of mesh faces, vertices and edges, the goal is to find a discrete partitioning of the mesh such that an objective function measuring some convexity criterion is minimized or maximized, subject to a set of constraints [79]. As an optimization problem, it can be solved borrowing various ideas from image segmentation and machine learning, such as region growing, clustering and spectral analysis [79].

Finding a meaningful set of correspondences between a pair of shapes is an important topic in shape analysis. The problem can be stated as, given a pair of input shapes, find a meaningful map between their surfaces. Depending on the application, different approaches can be broadly classified to ones that compute full versus partial correspondence, or dense versus sparse correspondence [93].

Discovering structure in the case of large and diverse shape collections proves more challenging than the single shape case. Nevertheless, both correspondence and segmentation can benefit from the collective information provided by a set of shapes rather than a single shape or pair of shapes in isolation.

Correspondence for shape families. Correspondence-finding algorithms often suffer from high distortion in cases where the geometry of input shapes is vastly different, for example when shapes are related by non-isometric deformations. Blended weighting [55], which computes a weighted combination of several low-dimensional maps, has been proposed as a way to alleviate this problem. Using blending, the output consists of

the best set of correspondences taken from each map, essentially combining the best features of each individual map.

Another major problem for correspondence-finding algorithms arises for shapes with different topology and connectivity, e.g. when shapes are missing parts. Fuzzy correspondences [53] have been proposed to solve this, with the key idea being that in cases where reliable correspondence cannot be established, a probability of correspondence can be used instead. The main idea behind this method is the use of diffusion maps to connect reliable correspondences via different paths along a shape graph, together with a way to detect noisy maps via path consistency. These fuzzy correspondences provide an interesting tool to explore shape collections with rich variations in geometry and topology. Additionally, a number of methods have been proposed for improving correspondence maps by using shape graphs and enforcing cycle consistency when composing maps along graph paths [42, 43, 72].

Co-segmentation for shape families. Solving for shape segmentation and then for part-level correspondence can be difficult since the segmentation results for each individual shape do not necessarily lead to corresponding shape parts. Potentially more accurate results can be achieved if segmentation and part correspondence are both solved for in conjunction. A recent trend of research has been based upon this observation, turning to co-segmentation, or more generally, co-analysis approaches, which jointly optimise for segmentation and part-correspondence in a collection of shapes. In general, these co-analysis algorithms can be broadly classified according to the machine learning method they use. There are supervised, semi-supervised, and unsupervised co-segmentation methods.

Supervised methods [49, 91] rely on an input set of manually segmented and labeled shapes, which is then used to probabilistically segment and label the actual input set of shapes. They require more user interaction than semi-supervised methods, as they involve considerable user effort for manually segmenting and labeling shape training sets. This fact makes them less appealing for co-segmenting very large shape collections.

Semi-supervised methods require more limited user input for interactively learning the co-segmentation of a set of shapes. The recent method by Wang et al. [95] allows the user to iteratively mark constraints such as 'must-link' or 'no-link' that would refine the results, while at the same time automatically suggesting such constraints to the user. This approach indicates that with relatively limited user intervention, co-segmentation results can be nearly perfect. Some recent methods have focused on extracting a consistent hierarchy for a set of shapes [92], or consistent part arrangements [105] from an input family of shapes.

Unsupervised co-segmentation methods do not rely on any assumptions other than requiring the input shapes to belong to the same semantic class, for example a set of chairs or a set of tables. The Shuffler part-based modelling system [60], solves for a compatible segmentation between input shapes by first over-segmenting individual shapes into a hierarchical segmentation according to a convexity measure and then collecting segments together to form parts by optimizing a cost function that uses convexity and geodesic distances between would-be compatible parts.

The work from Golovinskiy et al. [34] pre-aligns input shapes and then clusters shape faces into parts by first constructing a graph connecting neighbouring faces on the same shape and corresponding faces on other shapes according to a distance measure, and then optimizing for a separation of the graph using a graph-cut algorithm. The style-content separation work of Xu et al. [99], first over-segments shapes into parts, and then clusters them into different styles according to their anisotropic part scales. Shapes in each cluster are then co-segmented using Golovinskiy et al.'s [34] approach, and inter-cluster correspondence is achieved through minimizing a deformed-to-fit criterion between potential corresponding parts. This approach achieves better quality results for collections of shapes with greater variability that can be explained through the different part-scale clusters.

Sidi et al. [84] propose spectral clustering of shape segments resulting from an initial over-segmentation procedure, according to a feature descriptor using diffusion maps. Compared to previous approaches, they can handle cases where corresponding parts differ in pose, location or even cardinality since they use a descriptor space for clustering rather than the original shape spatial coordinates. They also take advantage of connections between shapes, e.g. in the case where corresponding parts are very dissimilar, correspondence between them can still be established if there is a path of correspondences linking these parts through other parts. This has recently been improved further by Hu et al. [38], by using clustering in multiple sub-spaces rather than concatenating all shape features in a single descriptor space.

Other unsupervised methods for solving the co-segmentation problem include linear programming [39], multi-label optimization [69], and template fitting [54]. Template-fitting [54] is of particular interest, as it jointly solves for alignment, correspondence, segmentation and a deformation model to describe the variations in a shape family, starting from a set of primitive template boxes, either computed automatically, or with user supervision.

Recently, the concept of symmetric functional substructures [106] has been proposed for part-based model synthesis from semantically-related families of shapes. The key

idea is that starting with a graph of shape parts for each shape, connected through relations such as symmetry and contact, we are able to match corresponding sub-graphs based on symmetry that should bear the same functionality. This creates a notion of correspondence between the triplets of parts involved in the sub-graph, which can then be substituted in and out of models to create novel models which retain their original functionality. In this way, this approach implicitly creates a correspondence between shape parts at a higher level of part triplets, without solving for direct part correspondence. A similar idea has been explored in Zheng et al. [105], where an unsupervised algorithm was developed for detecting recurring part arrangements, based on spectral clustering in a part-pair, box-based geometric descriptor space.

Contribution. The majority of works on shape co-analysis focus on consistently segmenting sets of shapes using a range of strategies. The question of how to use such results to model and re-use structure has not yet been thoroughly investigated. In particular, one important question is how to use such results to explore and understand the types of variations that exist in large shape collections.

We try to answer this question in Chapter 4, where we propose a parameterization of a shape collection based on positions and sizes of template parts fit to shapes that allows to concurrently explore the collection and synthesize new shapes. Such a parameterization can also be used to generate large amounts of synthetic 3D content to train object proposal estimators for depth images, as demonstrated by Zheng et al. [104].

Another question is how to use such results to extract and encode a representation of structure that can capture the geometric *essence* of a shape family. The works more closely related to this idea are those of Chaudhuri et al. [15] and Kalogerakis et al. [48]. Inspired by earlier efforts to represent shapes as parts and their connections [30], they used a part-based representation as the basis for learning probabilistic models to describe shape families. These models represent topological information such as the likelihood of the presence of certain parts and the cardinality of these parts, and the existence of certain shape styles according to the topology and geometry of the shapes. The models can be used to synthesize new shapes, but parts are placed with an automatic procedure.

We attempt to answer this question in Chapter 5, where we propose a meta-representation of shape structure in the form of a probabilistic model that can be used for editing and exploring shape collections. While our model is also based on a part-representation, we instead focus on the geometry of the part configurations. We learn the relative positioning and appearance of shape parts that is typical for a family of shapes. Thus, our work complements the models from previous work, which focus on topology.

2.2 Structure-aware Shape Processing

Different research areas in geometry processing can benefit from a model of structure. We review related work in shape editing, repository exploration, and shape synthesis, and demonstrate how structure can be beneficial in these areas.

2.2.1 Shape editing

Shape editing has been a classic topic in geometry processing over the past few decades. A naive algorithm for editing would consist of low-level mesh operations such as moving individual mesh vertices and possibly adding and removing mesh faces. However, this quickly becomes inefficient and complex to handle for large meshes.

Free-form deformation. Early approaches have been based on free-form deformation [21, 78], where instead of moving individual vertices, a group of vertices are affected by a weighted combination of a small set of basis vectors, represented as a cage enclosing the shape's geometry. The goal of such approaches is to apply smooth, low-frequency deformations affecting the global structure of a shape while at the same time preserving high-frequency details of the shape. Their main drawback is that the results of user edits, applied by manually adjusting the cage control points, are not always intuitive due to the global impact of all control points to the mesh surface.

Subsequent approaches [87, 90] have relied on preserving physics-based properties such as elasticity, to remove the need for manually prescribing and manipulating control points. Such properties have been modelled according to a constraint energy which should be minimised so that the properties as preserved as much as possible. A large body of work has been devoted to preserving such properties [10].

Structure-aware editing. Structure-aware editing methods recognise the need for preserving global shape properties, unlike surface-based editing tools [88, 102], which aim to preserve local differential properties. Such global properties come in the form of relations among discovered shape parts, such as symmetry, parallelism and planarity, among others. A notable example of such methods is the iWires system [32], where a set of sharp 1D lines, called wires, are detected on a mesh, in analysis phase, along with their relations such as symmetry, parallelism, coplanarity, colinearity. Then in the interactive editing phase, the user edits the mesh, an initial elastic deformation solution is computed, and finally relations between wires are preserved in an iterative propagation algorithm. Similar in spirit is the approach from Zheng et al. [107] which uses component-wise boxes and generalised cylinders instead of wires, as proxies for

discovering and preserving part relations while editing. Xu et al. [101] use slippable motion analysis to detect joints on shapes, which can then be used as articulations to deform them by bending these articulations. Li et al. [66] introduce *arterial snakes* as feature curves used to deform 3D shapes that are inherently 1D. Habbecke et al. [35] focus on efficiently solving the constrained minimization problem that results from such relation-preserving interactive editing applications. They study under-constrained editing systems, linearize the constraint functions and examine the nullspace of possible solutions.

One notable example of a constraint-aware shape editing system is the ShapeUp framework by Bouaziz et al. [11]. This is a general framework consisting of a number of projection operators that can project shape vertices, polygons, 1-ring neighborhoods and higher-order constructs onto a set of plug-and-play constraints that can be added according to the user goals. The system then iterates between projecting the shape vertices onto the user constraints, and optimizing their positions in a linear fashion, until convergence. It can be used for a number of applications beyond shape editing, including parameterization, shape space exploration, and mesh optimization.

The structure-aware editing solutions discussed above do not change the topology of the shape being edited. A recent body of work from Bokeloh et al. [7, 8] aims to perform topological changes by discovering regular patterns of geometry in the original shape, and then copying and composing such patterns in a way that leads to the creation of novel shapes that preserve the detected regularity properties. Such regular patterns are not always easy to detect, so in a different approach, irregular resizing of geometry such as architectural buildings has been studied by Li et al. [67], by decomposing original shapes into a hierarchy of parts defined by the user, along with constraints such as replicate, delete and scale.

Contribution. State of the art structure-aware editing tools are good at detecting and subsequently preserving various kinds of relations and shape properties. However, they do so by only employing information from the shape being edited. We argue that there is more to learn by observing a family of shapes with similar semantics and functionality, rather than a single shape in isolation. Learning the common structure of a shape family has been demonstrated by Yumer et al. [103], however their approach expresses this as a simplified abstract mesh, which serves as the most representative mesh for all the shapes in the collection, not a truly usable model of structure. Our approach, detailed in Chapter 5, aims to learn and model such structure in a probabilistic model, similar in concept to the one used by Chaudhuri et al. [15], but with a multitude of unary and pairwise part relations. We use this model to illustrate various shape editing tools.

2.2.2 Shape synthesis

The concept of part-based shape synthesis was demonstrated in the Modeling by Example system [30] which allows the user to retrieve models from a database by text-based or shape-based search, cut-out desirable parts and stitch them to the model being built. The concepts behind this modelling tool were similar to existing data-driven tools like Shape by Example [86], in that new shapes were created out of existing shapes. The important difference to these tools however was that parts of shapes were recombined to create new shapes instead of interpolating between them. This work also demonstrated and addressed the main challenges in part-based modelling, namely that shapes are difficult to cut into meaningful parts, they have many degrees of freedom so they are hard to position and stitch together and they have no obvious similarity measure for locating relevant parts. Subsequent part-based tools focused on providing solutions to a number of these challenges, either comprehensively or individually.

Manual cut-paste methods. In the Modeling by Example system by Funkhouser et al. [30], new models are built by retrieving models from a shape database, manually cutting parts from them and attaching them together. Every new part is translated and scaled to match the center of mass and scale of the part it replaces, according to the method from Horn et al. [37]. Funkhouser et al. [30] recognize that this simple strategy for placing new parts is not adequate for cases where model parts have complex structural relationships between them. For example, one cannot easily replace the door of one car with the door of another type of car without the proper deformation.

In minimal-cut composition [36], given two models approximately aligned by the user and user-defined constraints on which parts to keep from each model, the minimal cut on both models is found using a volumetric approach similar to constructive solid geometry approaches, after which the models are stitched together to form a new shape. In this work, there was a part-in-whole alignment of parts of interest, e.g. alignment of a dog and a cow according to their heads and not the whole shape. This was done using simple PCA for a coarse placement and ICP for refined placement of the parts.

In the SnapPaste system [82], a cut-paste paradigm was employed where users cut parts and overlap them to existing shapes until they automatically snap into place. A variant of the ICP method, dubbed soft-ICP was used to place and attach new parts to an existing model. This method calculates a different rigid transformation for aligning every pair of correspondent point neighbourhoods in the overlapping regions of the new part and the existing model, resulting in a non-rigid transformation for blending the new part into the model. This method is interactive, so the approximate placement is done by the user, until enough overlap exists between the new part and the existing shape,

when soft-ICP takes over and creates the snapping effect, merging the new part to the shape. The problem is that both the new part and the shape need to have clear boundary loops, and if this is not the case, they need to be cut.

Automatic shuffling. In the Shuffler system [60], a collection of semantically similar shapes is compatibly segmented into parts, which are then shuffled between shapes by the user to create new shapes. A common adjacency graph is created for each pair of shapes, which becomes disconnected when a part is shuffled out, forcing parts in the broken branches to be transformed to be aligned with the shuffled in part. This method can only work for consistently segmented shapes that have the same part connectivity.

Similarly to the Shuffler system, interpolation between two shapes using parts from both shapes to create new variations has been studied recently by Jain et al. [46]. In this work, the parts selected from pairs of shapes in order to create a new shape are brought together by enforcing their contacts using a spring-mass system. This can be a good way to enforce contacts between adjacent parts from the source shapes, however, beyond contacts, it does not take into account the deformation that may be needed in order to fit parts together.

Sketch-based interfaces. Tools employing ideas from sketch-based modelling have also been developed to synthesize novel shapes. In Lin et al. [68], user sketches are used to cut parts from models and define the expected silhouette of their intersection, after which particle sampling is used to create a new mesh to fuse the parts together. Lee et al. [63], propose a system similar in concept to the Modelling by example system, where relevant parts are found and attached to shapes using sketches of the user. Every new part is placed in the scene using optimisation, in order to match the outline of the user's sketch and be in stable contact with the rest of the scene.

A different sketch based part assembly tool was developed by Xie et al. [97], for exploring design possibilities by scribbling over existing model parts in order to replace them with different parts from a set of pre-segmented models. This tool allows fast refinement of design choices and employs pre-learned context information to improve the part retrieval process. A similar tool for designing 3D scenes with many objects, rather than a single object, was developed by Xu et al. [98].

Data-driven automatic synthesis. In order to address the synthesis challenge of locating shape parts relevant to the modelling task, data-driven suggestions [16], context-based search [27, 28] and probabilistic graphical models [15] have recently been used. At the same time, automatic methods have been proposed, that take as input a collection of semantically-similar shapes and create novel variations of these shapes using probabilistic graphical models [48], or evolutionary algorithms [100]. Placing parts from

different shapes together in a plausible way is the main challenge for these automatic approaches.

The probabilistic model of Kalogerakis et al. [48] ensures that only parts with compatible contacts are instantiated together, for example a seat from a chair with four legs would be instantiated in a new shape along with four legs and not with a single stool leg. The contacts of a part in the original shape can then be used together with least-squares minimization to attach it to other parts and form new shapes.

The evolutionary algorithm approach of Xu et al. [100] uses crossover and mutation operators on shape parts to create new shapes. Crossover is defined on pairs of shapes and requires the selection of a subset of parts from one shape to replace parts on the second shape according to some correspondence confidence. Mutation is performed randomly on the properties of the controllers of some of the parts of shapes resulting from crossover. The connectivity, proximity, ground contact and other properties of the structure of the parent shapes are preserved using the structure-preserving controller framework of Zheng et al. [107]. Symmetric functional substructures [106] have also been proposed for automatic part-based model synthesis from semantically-related families of shapes.

Contribution. Most methods for shape synthesis perform a simple copy-paste operation that places parts in the new shape. Depending on the application, a simple alignment of the new part to the old part can be done in cases where replacement of a single part is performed [30], or a more involved alignment and distance minimization for enforcing contacts can be done in more complex cases [60]. Structure-aware models [15, 48, 100, 106] have been employed more recently for adjusting shape parts in order to enforce a variety of complex structural constraints inherited from the parent shapes involved in the part-based synthesis process. Such methods do not provide the user with a high-level preview of the space of possible models that can be synthesized.

Based on a template-based shape parameterization, in Chapter 4 we propose an automatic part-based synthesis tool that is both fast and intuitive to use, by embedding a shape collection in a low-dimensional, hierarchical shape space. Out tool provides structure-preserving shape synthesis, and at the same time provides a solution to one of the major problems of part-based synthesis tools, namely the difficulty in locating shapes and shape parts relevant to the goals of the modelling session. This is a very common problem for modellers, especially at the initial, open ended stages of modelling, where their image of the target 3D model is vague. The template-based shape parameterization in Chapter 4 can also be used to generate large amounts of synthetic 3D content to train object proposal estimators for depth images, as demonstrated by Zheng et al. [104].

2.2.3 Shape exploration

The ability to explore shape collections to understand the types of shapes contained inside, as well as the variations between these shapes is becoming more necessary as these collections continue to grow. Locating shapes relevant to the modelling task in hand is a crucial part of structure-aware modelling tools. A number of methods have been proposed to tackle this challenge, ranging from basic shape matching and retrieval, to recent qualitative exploration methods.

Geometric similarity. Shape retrieval based on nearest neighbour search in geometric shape descriptor space has been used in early systems for exploring collections of 3D shapes. Descriptors can be rotation-invariant, in which case alignment is not needed beforehand, and ideally they must be low-dimensional and sufficiently descriptive. Earlier systems [31] required an initial shape to be given as input in order to locate shapes similar to it. The Modelling by Example system [30] used text search as well as shape-based search to retrieve models relevant to the modelling task. Shape-based search was implemented using a volumetric descriptor, and the shape being built was used as the query shape. The user was also given the ability to add more weight to parts of the shape that were more important in the search. A review of shape matching using geometric features can be found in Kazhdan et al. [50]. Later efforts in geometric-based shape retrieval employ sketch-based interfaces [24, 63, 80], allowing the user to search for models by drawing 2D sketches of their outline.

Context-based search. Context from 3D scenes has been used as a means to explore collections and retrieve shapes. In Fisher et al. [27], a box was used to locate models that fit in specific parts of a 3D scene. The box is drawn approximately at the part of the scene and the size that the user requires, and the most relevant models are suggested as a result. This is made possible by building scene graphs of objects in 3D scenes and extracting tags related to their name. Given the geometry, the tags and the spatial relationship of two models using the scene graph, it is possible to predict the relevance of a query model and the models in the database. In later work [28], this concept was extended to use kernels on scene graphs to compare query models as well as whole scenes for similarity. The same idea has also been applied for synthesizing novel 3D scenes using a sketch-based interface [98].

Correspondence-based search. Shape retrieval methods based on geometric descriptors or context are only applicable when a user has prior knowledge of the shapes inside a collection, which is itself a challenging task for unorganized datasets. Ovsjanikov et al. [73] observed this fact and proposed an alternative exploration tool. Their system

is based on extracting a template-based deformation model that explains the continuous variations in the input collection, and then deforming the extracted template to locate relevant models. Kim et al. [53] focus on extracting point-wise fuzzy correspondences between shapes in a collection, and using these correspondences to guide shape exploration.

Semantic similarity. Probabilistic graphical models have been used for retrieving parts relevant to a modelling task [15]. In this work, a shape collection is used to build a Bayesian network that describes the semantic as well as the geometric relationships of shape parts. Then at runtime, inference on the Bayesian network, given the observed shape built so far, is used to suggest parts semantically and stylistically relevant to the observed shape. This model is extended by Kalogerakis et al. [48] to not only suggest relevant parts but also build entire shapes automatically in order to enrich shape collections with new shapes.

Qualitative exploration. Tools based on qualitative organization [44] and dynamic embedding [58, 75] have been proposed for exploring shape collections. Although such systems provide intuitive local exploration of existing models, none of these methods allow synthesizing novel shapes during the exploration session. Talton et al. [89] propose an intuitive interface that tightly couples exploration and synthesis for parameterized model families. This is the first example of a coupled exploration and shape synthesis interface in the literature. However, it assumes a compact parameterizable design space [1, 96], which is impractical for large shape collections with complex variations.

Contribution. Early approaches for exploring shape collections relied on text-based, geometric-based, and sketch-based search. Recently, context, probabilistic graphical models, correspondence, and qualitative exploration methods have been used to improve the exploration process. We offer a solution to the problem of exploring large, diverse and unorganized shape collections, through an approach that parameterizes shapes using a template abstraction and embeds them in a low-dimensional, hierarchical shape space that lends itself to fast and intuitive exploration. Our approach provides the added benefit of allowing rapid and intuitive part-based shape synthesis, tightly coupled with the exploration interface. Chapter 4 provides details on our approach.

Chapter 3

Efficient co-alignment of shape collections

3.1 Motivation

Consistent alignment of 3D shapes is a fundamental problem in the process of analyzing shape collections and discovering structure. Bringing all shapes to a common coordinate frame is required for shape matching, analysis and visualization of variations, as well as classification of large geometric collections. Placing shapes in a canonical frame facilitates part- or point-level correspondence establishment and subsequent transfer of information across the different shapes, which is necessary for many of these methods. The raw shape collections, however, rarely come consistently aligned. While for small collections such models can be manually pre-aligned, unsupervised automatic algorithms are essential for large to very large-scale model collections. For instance, a collection of 1000 chairs can be aligned manually by a human in around 8 hours, compared to a few minutes using our unsupervised co-alignment algorithm. In the scope of this thesis, consistent alignment of 3D shapes is the first step needed for discovering shape structure from large and diverse shape collections. The results from this chapter are used in Chapters 4 and 5 to model structure and power structure-aware shape modelling tools.

A simple solution to the consistent alignment problem is to individually align each shape by mapping its principal axes to the global x-, y-, z- axes [30]. Although this approach scales linearly with the number of models, it is unfortunately unstable, and can suffer from misaligned axes (see Figure 3.1,b). Another solution is to align each shape to an arbitrarily chosen reference shape by exhaustively searching for the best alignment in the space of pairwise relative orientations. This method, however, is heavily biased

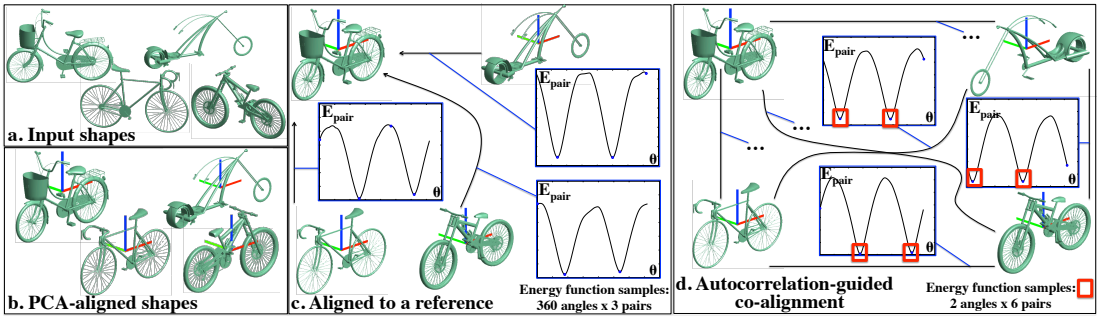


Figure 3.1: Shape collections typically come with inconsistent orientations (a). PCA-based alignment (b), or aligning to an arbitrarily chosen base model (c) is prone to error. The problem with pairwise alignments is attributed to several minima in alignment distances (E_{pair}), arising due to near-symmetries of shapes. We introduce an autocorrelation-guided algorithm to efficiently sample the minima (red boxes) and jointly co-align the input models (d).

on the initial choice of model and can degrade in the case of large shape variations across model collections (see Figure 3.1,c).

A better strategy that avoids the bias of aligning shapes independently is to co-align all the models simultaneously, without arbitrarily committing to a single reference model. However, directly comparing all the model pairs at all possible relative alignments is expensive and quickly becomes unattractive as the size of shape collections grows. An alternative is to select a subset of model pairs from the collection, uniformly sample their pairwise alignments, and use consistency in the alignment of these pairs to co-align the models using a labeling formulation (e.g., [41, 53, 106]). Even such methods are computationally expensive, as the uniform sampling of pairwise alignments leads to a large number of unnecessary comparisons which yield orientations that would never be chosen as optimal.

We focus on the co-alignment problem in the context of large and diverse shape collections and propose a method that goes beyond redundant uniform sampling of relative orientations, leading to efficient and accurate alignments. Our hypothesis is that by taking advantage of the rotational near-symmetries of shapes, we can drastically reduce the complexity of shape collection co-alignment, without sacrificing alignment accuracy. We first make two key observations: (i) A pair of similar shapes is easy to align even using simple alignment methods; while shapes with large geometric variations are difficult to align due to multiple candidate alignments with small inter-surface distance. (ii) Comparing a shape to itself, i.e., the autocorrelation function of a shape reveals insights into the possible sources of confusion arising out of self-similarity. Based on the first observation, we only align similar shape pairs and then diffuse the alignment information to other shapes through a shape graph. However, this requires first to

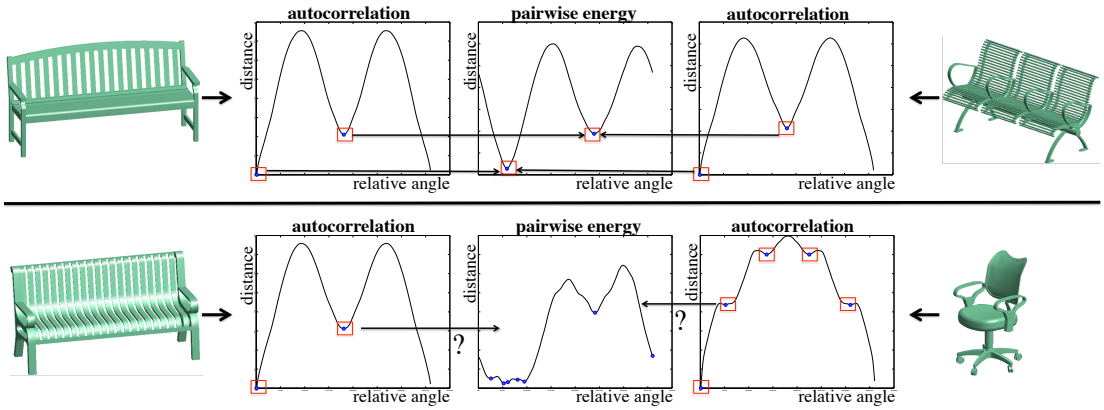


Figure 3.2: We use autocorrelation shape descriptors to predict model similarity *without* explicitly comparing them. If two shapes S_i and S_j are indeed similar with respect to their rotational near-symmetries, we use their autocorrelation descriptors E_i and E_j to predict potential relative alignment configurations between S_i and S_j that should be further investigated. In this example, the top half shows similar models whose autocorrelation functions can be used to predict their relative alignment; the bottom half shows dissimilar models whose autocorrelation functions provide confusing signals for their relative alignment.

identify which shape pairs are similar *without* explicitly comparing them. We exploit the second observation to not only determine which shape pairs to explicitly compare, but more importantly, to discover which relative alignments can potentially lead to ambiguity and hence should be further examined. For example, in Figure 3.1,d our method efficiently co-aligned a set of bikes, sampling only two low-energy alignments for each pair of shapes.

Models in shape collections (e.g., Trimble 3D Warehouse, TurboSquid) typically come with consistent up vectors. If this is not the case, one can use the method of Fu et al. [29] to consistently orient the shapes upward. Therefore, co-aligning them effectively involves resolving a 1-dimensional rotational ambiguity about the up vector. We propose a descriptor based on rotational autocorrelation of a shape, and an associated method that allows us to intelligently sample only a small number of candidate alignments for shapes. This results in an efficient algorithm that is also input sensitive, i.e., the running time depends on the extent of co-alignment ambiguity in the corresponding shape collection.

Figure 3.2 illustrates the relationship between rotational autocorrelation and ambiguities in alignments, in the context of a chair-bench shape collection. The algorithm is robust as it only considers symmetrically-similar shapes while deciding which shape pairs to compare. For example, it would not choose to compare the bottom two shapes in Figure 3.2 since they have a different set of self-symmetries.

Starting from an input set of shapes with similar semantics, for example a set of bikes, we compute the autocorrelation descriptor per shape, and use it to cluster input shapes such that shapes with similar rotational symmetry end up in the same cluster. We then build an alignment graph per cluster of shapes, and align shapes inside the cluster by efficiently sampling their candidate alignments and minimizing a formulation of the sum of pairwise shape distances allowing multiple local minima, using the autocorrelation descriptor as a guide. Finally, we align shape clusters by a modified formulation of the sum of pairwise shape distances, this time from shapes between clusters.

We prepared ten diverse benchmark datasets with ground truth alignments, and evaluated our approach against alternative alignment strategies. We report comparable alignment accuracy to state-of-the-art methods at only a fraction of the time. Specifically, we observe 2-16x speed improvement in our tests.

3.2 Overview

Our algorithm takes a collection of shapes $\mathscr{S} = \{S_i, i = 1, \ldots, n\}$ as input, and produces a canonical transformation for every shape $T_i = R_i T_i^{\text{norm}}$, where normalization T_i^{norm} is performed for each shape independently, and the key focus of our work is effectively estimating R_i that minimizes distances between all pairs of shapes. We assume that T_i^{norm} aligns the up vector of shape S_i to z-axis, and parameterize R_i by a rotation around up vector, $R_i = Rot_z(\theta_i)$. We formulate our objective function as:

$$E := \sum_{i,j} E_{i,j}(\theta_i, \theta_j), \tag{3.1}$$

where $E_{i,j}$ estimates how well S_i and S_j align if rotated by θ_i and θ_j respectively. Sampling the energy function $E_{i,j}$ is the most expensive step of the algorithm since it requires computing distances between surfaces for all relative angles of the form $\Delta_{i,j} = \{\theta_i - \theta_j\}$.

They key observation behind our work is that ambiguity in shape alignments usually arises due to approximate rotational symmetries of shapes. For example, in Figure 3.2 the individual autocorrelation descriptors of the benches already provide valuable clues as to which relative angles between the two benches can be ambiguous, even without explicitly comparing the two benches. In other words, if shapes are expected to be similar, one does not need to evaluate pairwise energy to predict this potential ambiguity since the shapes would exhibit similar near-symmetries. Thus, we estimate the number

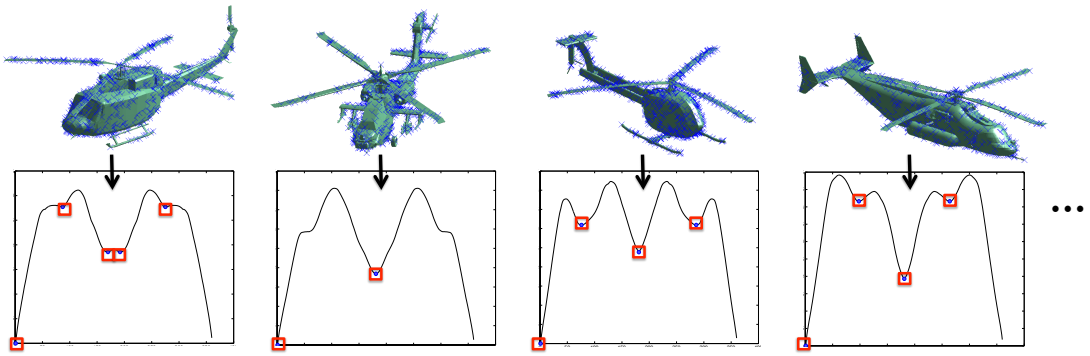


Figure 3.3: We normalize and sample points on each shape, compute its autocorrelation descriptor E_i, and store the set of E_i's local minima (highlighted by red boxes).

of ambiguous alignments between pairs of objects, as well as the relative angle between the ambiguous alignments by analyzing autocorrelation function:

$$E_i(\theta) := E_{i,i}(0,\theta), \tag{3.2}$$

which captures how similar is a shape S_i to itself under a rotation θ. Figure 3.3 shows an example.

In particular, we expect self-symmetries of shape S_i to form an algebraic group of k_i elements: $G_i^{\text{symm}} = \{\theta_{i,l_i}^{\text{symm}} = 0, \ldots, \theta_{i,k_i}^{\text{symm}}\}$, where S_i is self-similar under rotation $\theta_{i,l_i}^{\text{symm}}$. Thus, for each shape S_i we only need to consider k_i canonical alignments related by angles in the symmetry group. In order to evaluate $E_{i,j}$, however, these canonical alignments have to be consistent between S_i and S_j, thus groups have to be co-aligned by an offset alignment θ_i^{off} (see Figure 3.4). Note that finding offset alignment θ_i^{off} is much easier than finding canonical alignment θ_i, since we only need to find how to align one element of a group, without needing to resolve ambiguities. We estimate the offset by aligning each shape to an arbitrary reference S_r as: $\theta_i^{\text{off}} := \arg\min_{\theta'} E_{i,r}(\theta', 0)$. Thus, for any $E_{i,j}$, we only need to consider:

$$\Delta_{i,j} = \{(\theta_i^{\text{off}} + \theta_{i,l_i}^{\text{symm}}) - (\theta_j^{\text{off}} + \theta_{j,l_j}^{\text{symm}})\},\ l_i = 1, \ldots, k_i;\ l_j = 1, \ldots, k_j \tag{3.3}$$

alignments, which drastically reduces the number of pairwise distances we need to compute to find all interesting minima in $E_{i,j}$.

The above formulation assumes that all objects in the collection exhibit the same approximate rotational symmetries. This, however, may not be true for heterogeneous data. One possibility is to take a product of symmetry groups to take all canonical orientations into account, but this would increase the number of pairwise distance

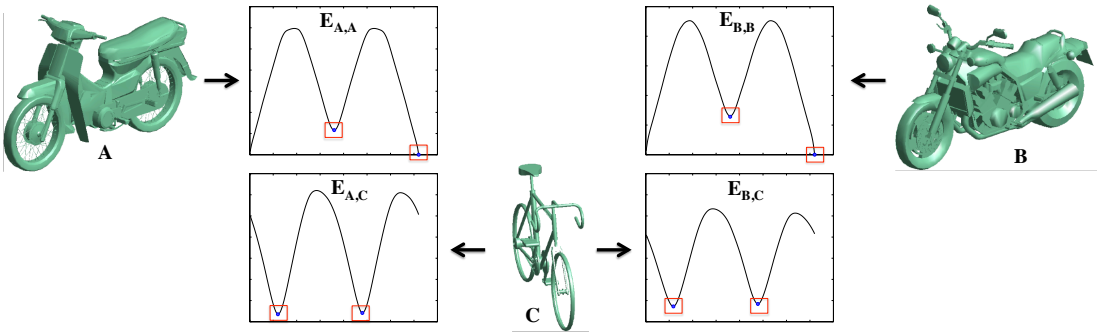


Figure 3.4: We show the autocorrelation functions $(E_{A,A}, E_{B,B})$ for two motorcycles (A, B), and their pairwise energy function $(E_{A,C}, E_{B,C})$ when comparing to a bike (C). A and B are aligned to the global axes while C is rotated 60° around z axis. Note how this causes the two minima of $E_{A,A}$ and $E_{B,B}$ (at 180° and 360°) to shift by 60° in $E_{A,C}$ and $E_{B,C}$. We can therefore expect to find an alignment of the two motorcycles and the bike at 240° or 60°, so there is no need to sample other rotations.

computations. Instead, we group objects based on their rotational self-symmetries using the similarities in their autocorrelation descriptors and optimize for alignments of objects within the same group. After all objects within the same group are co-aligned, we estimate inter-group alignment by solving a smaller optimization problem that only includes a few edges between shapes in different groups.

3.3 Method

Given a collection of shapes $\mathscr{S} := \{S_i\}$ our method produces canonical transformations $\{T_i\}$. The algorithm starts with per-shape analysis to find normalizing transformations T_i^{norm} and autocorrelation descriptors E_i for each shape. Next, we group shapes based on their descriptors, extracting clusters of shapes that share similar symmetries. The method co-aligns models in each cluster by leveraging autocorrelation descriptors to decide which pairs of models to align and which alignments to sample. After this intra-cluster alignment, our algorithm aligns the different clusters via another optimization. Our pipeline is summarized in Figure 3.5, and the rest of this section describes each step in detail.

3.3.1 Normalization

To estimate per-shape normalization T_i^{norm}, we scale the height to be one, and translate the center of bounding box to $[0,0,0.5]$, such that the ground plane is $z = 0$. All shapes used in our experiments have consistent upward orientation. For other datasets one can

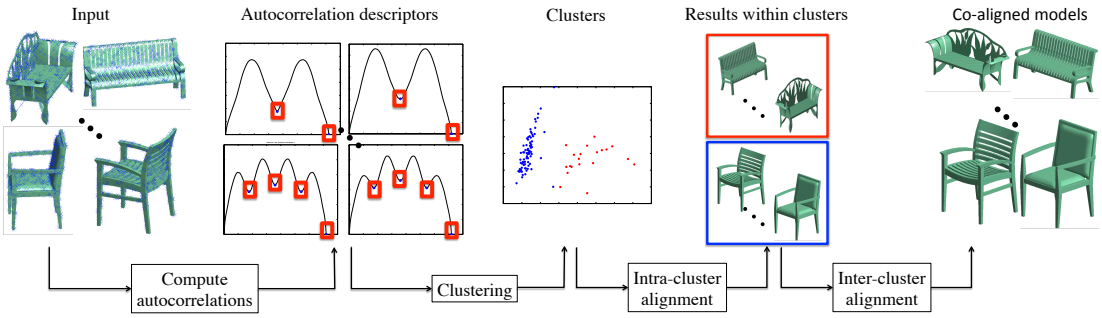


Figure 3.5: Algorithm overview. Starting from a set of shapes, we normalize and compute their autocorrelation descriptors to cluster the shapes. We then align the shapes first within and then across the clusters using a graph-based discrete formulation wherein we intelligently sample candidate alignments for each shape guided by their autocorrelation descriptors.

also use the method of Fu et al. [29] to consistently orient the shapes upward. The goal then is to find a rotation around the up-vector for each shape that would consistently co-align all shapes.

3.3.2 Autocorrelation descriptor

We leverage understanding of rotational near-symmetry of a shape to group shapes that have similar symmetries and efficiently sample good co-alignments. To represent the symmetry of each shape S_i, we compute the autocorrelation descriptor E_i, which measures how much the shape correlates with itself, under a rotation:

$$E_i(\theta) = [D_S(S_i,\ Rot_z(\theta)\ S_i)],\ \theta \in [0, 2\pi], \tag{3.4}$$

where $Rot_z(\theta)$ is a rotation around the up vector by θ degrees, and $D_S : S_i \times S_j \to \mathbb{R}$ measures distance between surfaces S_i and S_j. In order to compute distances in our experiments we uniformly sample 1000 points, P_i, on each surface, compute the mean distance from all P_i to their nearest point in P_j and the mean distance from all P_j to their nearest point in P_i, and take the maximum of the two. We uniformly sample E_i with 360 samples.

Figure 3.3 shows an example for a set of helicopters. The autocorrelation descriptor is normalized by dividing each entry of $E_i(\theta)$ by $\sum_\theta E_i(\theta)$. We use k-d trees to speed up the nearest point search which is the most expensive part in the distance function computation. There exist alternative measures of shape surface distance, such as Hausdorff distance. We chose the mean nearest point distance described above as it is less prone to outliers.

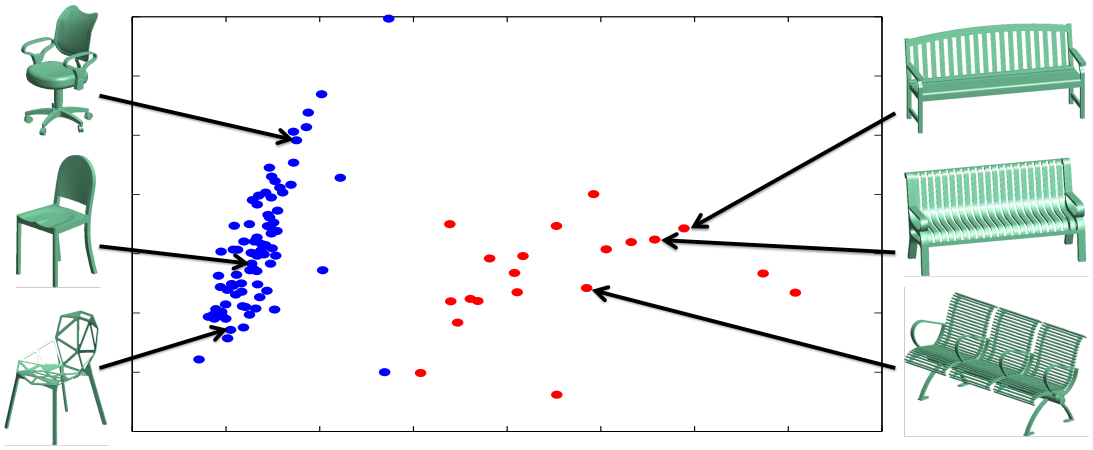


Figure 3.6: Multi-Dimensional Scaling embedding of autocorrelation descriptors for a dataset of chairs in 2D, with points colored according to the cluster they belong. Note how chairs are separated from long benches as they have different sets of self-symmetries.

3.3.3 Shape clustering

We group shapes based on their symmetries to ensure that we can effectively sample the pairwise alignment energy function and to find groups that can be co-aligned robustly. In particular, we group the input shapes into a set of clusters C_i using a graph-based technique. We take the kth largest pairwise distance in autocorrelation descriptor space as threshold t and connect all shapes with distance smaller than t with an edge. We then find connected components of the resulting graph to get the set of clusters C_i. In our experiments we set k to 10% of the number of shape pairs, which resulted in 6-21 clusters, depending on the dataset.

We use L1-norm distance between the autocorrelation descriptors to measure approximate earth-movers distance. Figure 3.6 demonstrates the clustering result on a chair-bench dataset, embedded in 2D using Multidimensional Scaling. Note how chairs and benches are separated in two clusters: benches typically have two near-symmetries with a potential confusion in a rotation by π, while chairs pose a bigger challenge with a larger number of near-symmetries.

3.3.4 Intra-cluster alignment

Next, we co-align all shapes within each cluster C_i. Note that the shapes in C_i share similar symmetries, and thus we can take advantage of E_i to efficiently sample pairwise energy $E_{i,j}$. First, we smooth E_i using moving average with a span of 0.03π and compute all of its local minima which gives us the group of symmetric rotations: G_i^{symm}, where

each rotation $\theta_{i,l_i}^{\text{symm}} \in G_i^{\text{symm}}$ leads to an ambiguity that needs to be resolved by joint optimization. Note that the minima in G_i^{symm} are defined in shape S_i's coordinate system, starting with 0 rotation, thus we need to bring all shapes to some canonical alignment space. This is however an easier problem than joint alignment, because it is sufficient to find one element of the group to estimate the offset alignment θ_i^{off}, which would define canonical alignments of shapes. We pick the shape S_r nearest to the cluster's center in descriptor space and estimate offset angle θ_i^{off} as: $\theta_i^{\text{off}} := \arg\min_{\theta'} E_{i,r}(\theta', 0)$, which we solve using an exhaustive search over 1 degree increments. With this offset, we can now effectively sample the pairwise energy function using the offsets described in Equation 3.3, as:

$$E_{i,j}(\theta) = [D_S(S_i,\ Rot_z(\theta)\ S_j)],\ \theta \in \Delta_{i,j}. \tag{3.5}$$

In practice, we use a Markov Random Field (MRF) labeling problem to minimize Equation 3.1, where possible canonical rotations for each shape S_i define the number of labels. In our case, these canonical rotations are $\theta_i^{\text{off}} + \theta_{i,l_i}^{\text{symm}}$ with $l_i = 1, \ldots, |G_i^{\text{symm}}|$. In order to select the set of shape pairs (i, j), we sparsely sample pairs of shapes by looking at $m = 20$ nearest neighbours in autocorrelation descriptor space. This parameter is set so that the resulting shape graph is kept sparse but remains connected.

We use the Maximum A Posteriori estimation method described by Leordeanu et al. [65] to optimize Equation 3.1. The input to this method is a matrix containing the pairwise potentials for each pair of connected shapes and each possible label (alignment) assigned to the shapes, as well as a vector of unary potentials, which we use to keep the shape with the median autocorrelation descriptor fixed, by assigning a large score to only one of its labels. The method tries to maximize the labeling score $x^T\ B\ x + U\ x$, where B is the pairwise potential matrix, U is the unary potential vector and x is the binary vector containing the solution. The solution vector x obeys discrete many-to-one labeling constraints such that $x(i) = 1$ if node i is labeled with label i, which means that shape S_i should be rotated according to the angle corresponding to label i. This method has a running time of only a few seconds for a dataset containing 100 shapes, and places no smoothness assumptions on the objective function, in contrast to standard graph-cut optimization methods. The output of this step is an angle θ_i^{ic} for each shape in the cluster, chosen from the candidate canonical rotations.

3.3.5 Inter-cluster alignment

After co-aligning shapes in each cluster, our method estimates inter-cluster alignments. In particular, we construct another joint alignment MRF problem where we seek to

select one rotation for every cluster out of a set of possible rotations (labels). First we apply rotation $Rot_z(\theta_i^{ic})$ from intra-cluster alignment to each shape of every cluster. We keep the largest cluster fixed, and compute the energy for each pair of clusters by connecting pairs of shapes from the two clusters and summing their pairwise energy $E_{i,j}$. Since shapes in different clusters do not share similar symmetries, we densely sample the pairwise energy $E_{i,j}$ with 32 uniform samples. We choose $m_c = 20$ edges for each pair of clusters (same as in intra-cluster alignment), connecting shapes that have the most similar autocorrelation descriptors E_i from the two clusters. The output of this optimization is an angle for each cluster, which can then be added to the angle θ_i^{ic} from the intra-cluster alignment step to obtain the final canonical rotation θ_i for each shape.

3.4 Evaluation

In this section, we evaluate our method on ten diverse shape datasets, ranging in size from 32 shapes to 1000 shapes. We design experiments to evaluate the performance of our method on different shape classes, quantify the efficiency improvement, the effect of various choices we made when designing the algorithm, and scalability. The following results are reported using an implementation of our method and competing methods in Matlab, run on a quad-core 2.2 GHz laptop with 16GB RAM.

3.4.1 Experimental setup

Datasets. Although many previous techniques rely on shape co-alignments, there is no standard benchmark for quantitatively evaluating these methods. Hence, in order to evaluate our approach we create a large and diverse benchmark with ground truth alignments for ten different datasets.

First, we use the correspondence benchmark provided by Kim et al. [54] that includes a small number of consistently annotated feature points for 100 bikes (including bicycles and motorcycles), 100 chairs (including armchairs and benches), 100 helicopters, and 104 airplanes. All models have consistent upward orientation aligned to the global z axis. After normalizing the models as described in Section 3.3.1, we fix one shape in the collection, and align every other shape to it by finding the optimal rotation around the z axis that minimizes the L2 distance between the ground truth correspondences.

Second, we downloaded six additional datasets from Trimble 3D Warehouse including 32 snowmobiles, 100 cars, 100 cups, 100 ships, 100 sofas, and 1000 chairs. All

models in each collection have a consistent upward orientation. Finally, we manually prescribe ground truth rotation for each model. We believe that the benchmark can be valuable for evaluating future algorithms and is thus made freely available along with an implementation of our algorithm in Matlab and all our results, in [3].

Evaluation metric. We evaluate the performance of our method based on how accurately it co-aligns shapes in a dataset. For each shape S_i, let the ground truth rotation around the up vector be denoted by θ_i^{gt}, and an algorithmically predicted angle by θ_i. We measure the alignment error $a(i,j)$ of a method based on the distance between the true and predicted relative angles as:

$$a(i,j) = |(\theta_i^{gt} - \theta_j^{gt}) - (\theta_i - \theta_j)|. \tag{3.6}$$

In the following comparisons, we plot the fraction of shape pairs (y-axis) whose alignment error $a(i,j)$ is smaller than a threshold (x-axis).

3.4.2 Per-class performance

Figure 3.7 shows the accuracy of our method for different datasets, and Figure 3.12 shows images of resulting alignments for a random selection of shapes. For a relatively strict threshold of 15° angle our method correctly aligned 80% of models for most datasets. Note that in all cases the results are significantly better than a random rotation, which would only achieve about 8% for 15° threshold. Low accuracy in some datasets is caused by near-symmetries. For example, the position of wings along the fuselage of an airplane can incorrectly favour nose-to-tail alignment. This problem is common for all shape matching algorithms, and our method is not designed to resolve this issue: it only speeds up the optimization time significantly by focusing on these challenging ambiguities.

Near-symmetries decrease accuracy both in intra-cluster and inter-cluster alignment. For example, the two biggest clusters of the airplanes dataset get around 70% and 85% accuracy for 15° angle. In comparison, models in the biggest clusters of the (small) chairs dataset are aligned with about 90% accuracy within each cluster, which is then reduced to about 60% by the inaccurate inter-cluster alignment. Please note that user supervision can significantly improve accuracy at a very small cost in the latter case. Section 3.4.5 gives the details and Figure 3.11 shows how accuracy is affected by supervision; in particular, we can get above 90% accuracy at a 15° angle threshold for (small) chairs with just 6 manual alignments (only one per cluster).

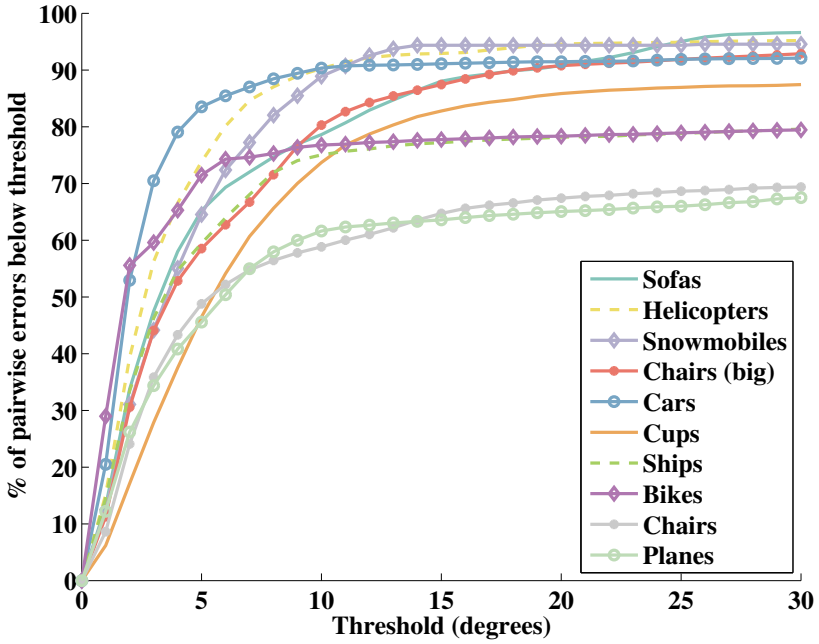


Figure 3.7: We plot the fraction of models (y-axis) aligned within a prescribed angle threshold (x-axis), for 10 datasets. All our results are substantially better than a baseline of random alignments, which would only give about 8% accuracy for a threshold of 15°. Similarly to all shape matching techniques, our method suffers from near-symmetries of shapes.

3.4.3 Efficiency improvement

Our main contribution is the method for efficient sampling of shape rotations based on potential ambiguities caused by near-symmetries. Hence, we compare our method to the state-of-the-art approach that uniformly samples the rotations as proposed by Huang et al. [41]. In their approach they assume that shapes are consistently aligned in the upward direction and take 32 uniformly-sampled rotations around the up vector for each pair of shapes S_i and S_j. Their method is designed to compute fine-scale correspondences, and thus they further co-deform the shapes using affine transformations and free-form deformations. Since the goal of our work is rigid co-alignment of models we only keep the first step of their pipeline that optimizes for rotations. To ensure fair and consistent comparison, we modify our implementation to mimic the method of Huang et al. [41] (titled as UNIFORM in all figures). In particular, we uniformly sample the rotation space, but we keep the same graph connectivity and inter-cluster alignment step for both methods.

Figure 3.8 demonstrates accuracy results averaged over all datasets. For the full set of results and per-dataset comparisons please refer to Appendix A. Our method achieves better accuracy with significantly less computational overhead. We gain the

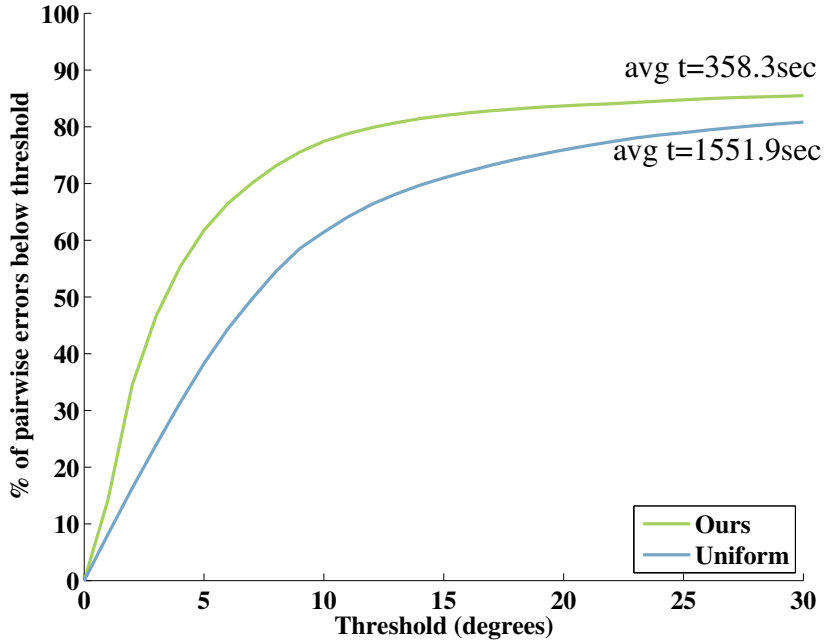


Figure 3.8: We compare our method (green) to our implementation of the UNIFORM method (blue), where the results are averaged over 10 datasets. Our method outperforms the UNIFORM method in accuracy. Most importantly, it has less computational complexity than UNIFORM, as it requires 2-16 times (depending on the dataset) less samples of $E_{i,j}$ to align shapes within all clusters, compared to UNIFORM (see Table 3.1).

most significant improvement over $0^\circ - 15^\circ$ thresholds since UNIFORM method is constrained to samples at 11° increments. Table 3.1 further provides detailed timing for each dataset. We get up to 16 times speedup (e.g., for ships) due to the symmetry-guided sampling of the energy function (the two rightmost columns compare the number of times the energy was computed). Table 3.1 also suggests that our method is input-sensitive, since datasets with more near-symmetries such as cups, helicopters, and planes take longer to align. However, even for these datasets our method is 2-3 times faster than the UNIFORM method.

One can further improve the accuracy as well as change the running time of the UNIFORM method by changing the sampling frequency. Figure 3.9 shows a representative comparison on a bike dataset, of our method and the UNIFORM method with various numbers of samples. As the number of samples increases, both accuracy and complexity of UNIFORM method increase. In contrast, our method achieves accuracy that is comparable to the 64-sample UNIFORM method, at a fraction of the time. Note that, theoretically, by taking 360 samples of the rotation space for each shape, it is possible to get up to 100% alignment accuracy for a very small threshold, if the shape distance function has a single global minimum. This comes at a great cost however,

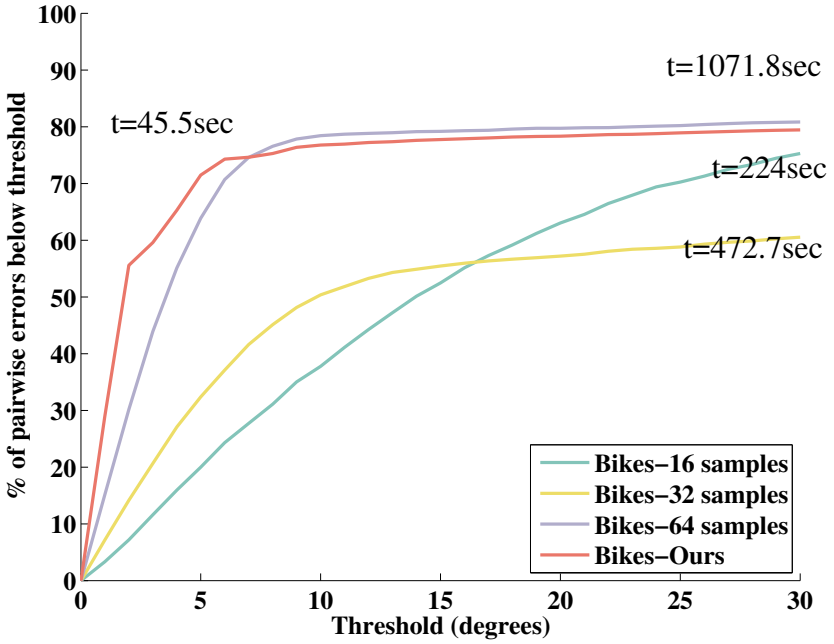


Figure 3.9: We plot the fraction of bikes aligned within a prescribed angle threshold, for our method and UNIFORM method with increasing number of samples. The accuracy of UNIFORM increases as the number of samples increases, at the cost of longer running time, while our method achieves higher accuracy at a fraction of the time.

as it is a brute force approach that is more than 10 times slower than the 32-sample UNIFORM method, and has a very large space complexity. For example the MRF optimization requires a 36000×36000 pairwise potential matrix for co-aligning 100 shapes with 360 labels each. Assuming a single-precision floating point representation for its entries, this matrix needs 5.184 gigabytes to be stored in memory, which is very large, even for high-end computers.

3.4.4 Effect of clustering

We examine the effect of the clustering step in our alignment pipeline. This step is designed to handle heterogeneous collections where shapes do not have similar symmetries. In particular, we cluster shapes based on their symmetry descriptors and then optimize for per-cluster rotations.

We compare the alignment accuracy of the results produced with the clustering step and by aligning all shapes jointly. The average alignment accuracy over all datasets can be seen in Figure 3.10. Note that the use of clustering and the two-step alignment procedure increases accuracy by around 15% for the 15° threshold. For the individual accuracy results of all datasets, please refer to Appendix A.

	Ours (sec)	UNIFORM (sec)	Ours ($\#E_{i,j}$)	UNIFORM ($\#E_{i,j}$)
Sofas	18.2	345.1	1902	29184
Helicopters	124.6	450.4	12180	36544
Snowmobiles	7.6	69.4	818	5824
Chairs(big)	3000.5	12149.5	291273	930528
Cars	23.8	435.3	2614	37696
Cups	116.3	434.6	11699	36320
Ships	17.6	384.6	1900	30400
Bikes	45.5	472.7	3560	32224
Chairs	93.6	428.4	9046	34688
Airplanes	135.2	349.5	14300	30944
AVERAGE	358.3	1551.9	34929	120435

Table 3.1: The first two columns show the time, spent on solving the optimization from Section 3.3.4 for our method and UNIFORM. Inter-cluster alignment time is excluded since it is the same for both methods. The third and fourth columns show the number of samples taken from $E_{i,j}$ in the same optimization problem. Note that our method is faster than UNIFORM, which takes 32 samples for pairwise alignments. Our method becomes more computationally expensive for classes of shapes that exhibit more symmetries, such as cups, airplanes and helicopters.

3.4.5 Effect of human supervision

While all results presented in previous sections were created fully automatically, our method can also efficiently leverage human supervision during the inter-cluster alignment step. After aligning shapes within each cluster, the user can further align different clusters to avoid unreliable matching of dissimilar shapes. In particular, our method picks a representative shape from each cluster (i.e., the shape nearest to the cluster's center in descriptor space). Then the user is prompted to consistently rotate representative shapes for all clusters. This bears little overhead since the number of clusters is small in comparison to the size of the collections (6-21 clusters for the small datasets, 34 clusters for the 1000-chair dataset) and the number of required manual alignments is equal to the number of clusters.

We simulate the human supervision using the ground truth angles. In particular, we pick a rotation that correctly aligns the representative shape of each cluster to the representative shape of the biggest cluster. Figure 3.11 shows the accuracy of our method with and without human supervision for the chairs and airplanes datasets. For the results on all other datasets, please refer to Appendix A. These two datasets had the lowest performance in terms of accuracy for the unsupervised approach (see Figure 3.7). With the supervision, the accuracy increases to over 90% for chairs and

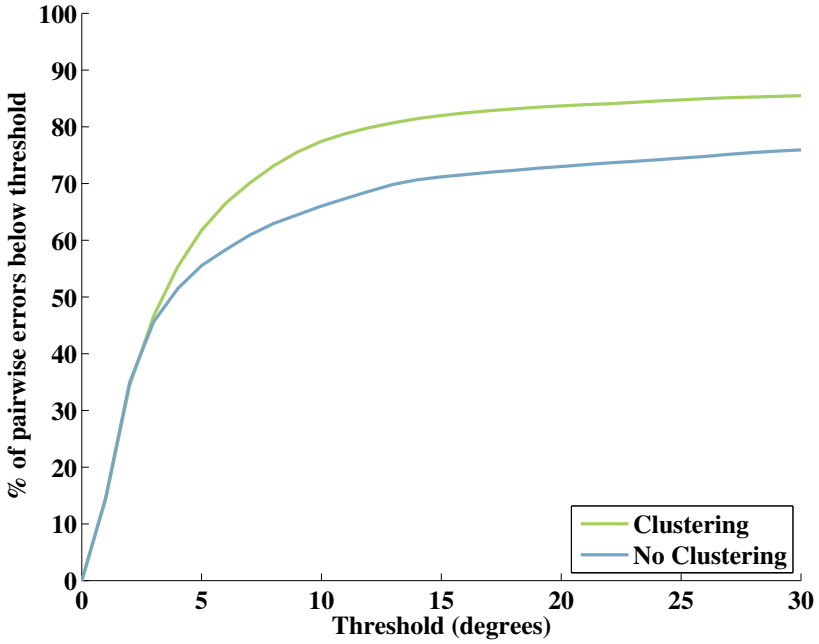


Figure 3.10: We evaluate the effect of the clustering part of our pipeline by comparing our method with clustering performed (green) to our method with clustering turned off, i.e. all shapes jointly aligned in one step (blue). This plot shows results averaged over 10 datasets. Clustering and aligning shapes in two steps increases accuracy on average, compared to jointly aligning all shapes.

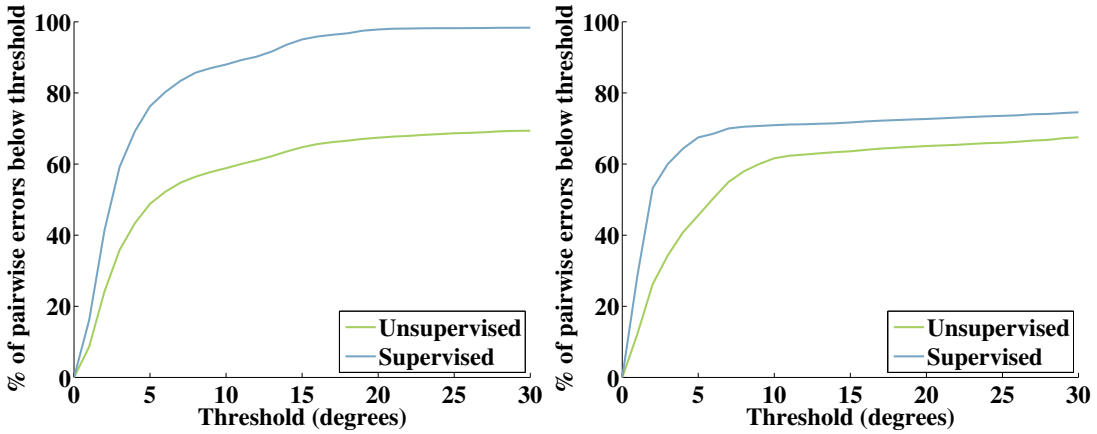
70% for airplanes. The improvement is prominent for chairs since most of the errors in that dataset are due to incorrect inter-cluster alignment.

3.4.6 Scalability

To demonstrate the scalability of our method, we test it on a large dataset of 1000 chairs. We set the parameter k, which controls the clustering radius to $k = 2.5\%$ since the dataset has higher diversity in comparison to the smaller collections. Since this dataset is much larger than the other datasets, we also iteratively increase the number of nearest neighbours that are connected by an edge, starting from $m = 20$ (used for all other datasets) and increasing m in increments of 15, until all individual graphs are connected (terminating at $m = 50$). Table 3.1 demonstrates the timings for the 1000 chairs dataset, where our method outperforms UNIFORM by 4 times.

3.4.7 Computational complexity

Since most of the computation time is spent on sampling the pairwise objective function $E_{i,j}$, we measure the computational complexity in terms of number of samples that have



(a) Chairs **(b)** Planes

Figure 3.11: We evaluate the effect of human supervision by comparing the accuracy of our unsupervised alignment pipeline (green) to the accuracy achieved with human supervision (blue), for the chairs and the airplanes datasets. Supervision helps improve accuracy for these low-performing datasets, using just 6 manually-prescribed rotations for chairs and 21 for planes.

to be computed. The intra-cluster alignment for a given cluster c requires $N_c m M_i$ samples, where N_c is the number of shapes in a cluster, m is the number of edges per shape (set to 20 in our experiments) and $M_i = |G_i^{\text{symm}}|$ is the number of candidate alignments (i.e., the number of approximate symmetries) per shape (2-12 in our experiments). The inter-cluster alignment requires $K m_c M_c$ samples, where K is the number of clusters (6-21 for all datasets), m_c is the number of edges we use to connect clusters (set to 20 in our experiments), and M_c is the number of candidate rotations per cluster pair (set to 32 uniform samples). The UNIFORM method has similar complexity, but the candidate per-shape alignments M_i is set to constant $M_{\text{uniform}} = 32$ rotations. Thus our method only speeds up the intra-clustering step as long as $M_i = |G_i^{\text{symm}}| < M_{\text{uniform}}$ and $N_c > 1$.

We report timings in seconds for each dataset for our method versus UNIFORM in Table 3.1. Our method outperforms UNIFORM between 2 (for planes) and 16 times (for ships) in terms of number of $E_{i,j}$ evaluations.

3.4.8 Discussion

As indicated by our comparisons, our method is several times faster and more accurate on average than the UNIFORM method for the ten datasets we tested it on. While our method makes no assumptions regarding symmetry of the input shapes, it is designed to improve efficiency only if shapes have a small number of near-symmetries, e.g., as we can see in Table 3.1, the improvement for bikes is more significant than for cups. In practise however, we have not observed any shape with more than 12 near-symmetries.

Our technique also assumes that there is a meaningful alignment for each shape in the input collection, and thus it can be sensitive to outlier objects. To test this, we collected a dataset of 109 cars from Trimble 3D Warehouse, with around 10% of the cars having additional noisy geometry or wrong up vectors. Our method achieved 75% accuracy at 10 degrees threshold, compared to over 90% for our original clean 100 car dataset. For results on the original clean car dataset as well as the noisy car dataset please refer to Appendix A. We also note that our method's efficiency degrades significantly if the clustering step produces a large number of small clusters, since we uniformly sample alignments in the inter-cluster alignment step. We have not encountered this in practice when dealing with collections that contain shapes from the same category. Lastly, our method only relies on geometry and ignores any additional cues such as texture, which often come with shapes retrieved from online collections. It would be interesting to study how such cues can affect alignment efficiency and accuracy.

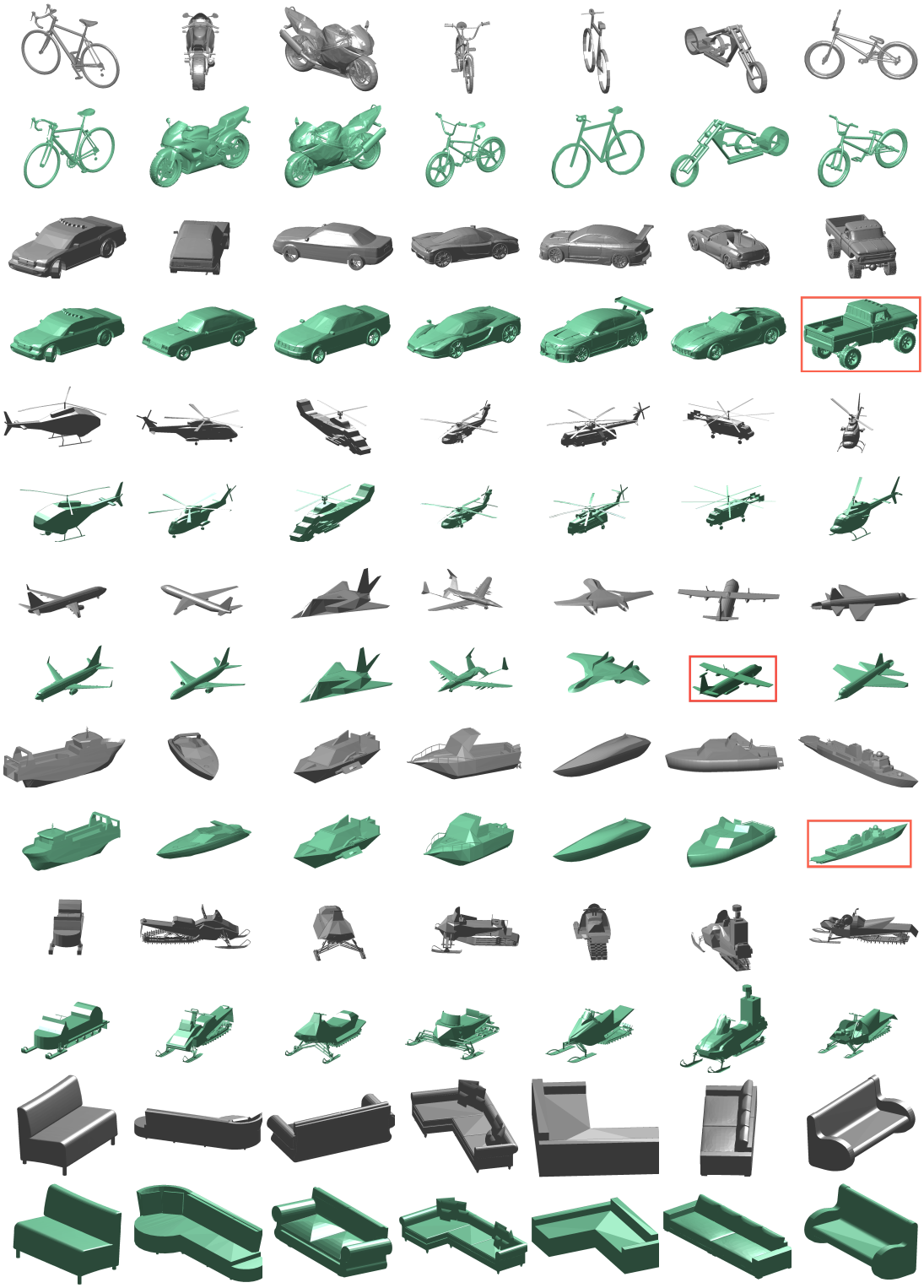


Figure 3.12: Randomly selected shapes from our datasets, indicating their pose before (odd rows - in gray) and after (even rows - in green) alignment. Red boxes mark mis-aligned shapes.

Chapter 4

Template-based shape parameterization

4.1 Motivation

In Chapter 3 we presented a fast method for co-aligning semantically similar shape collections, taking the first step towards discovering and modelling shape structure. Armed with consistently aligned shapes, we are now ready to propose a method for modelling structure from a shape collection, and demonstrate its application in two challenging problems, namely, exploring shape collections and synthesizing novel 3D content.

Creating 3D content is a fundamental problem in computer graphics, one that requires skill, artistic sense and training. It is a difficult task that requires a lot of time with existing modelling tools. This is time that 3D artists often do not have, especially in the initial, open-ended stages of modelling, when the artist's image of the target object to be modelled is vague. In this exploratory modelling phase, it is beneficial for a modeller to quickly browse existing shapes, seeking for inspiration. Online 3D shape repositories like Trimble 3D Warehouse are now becoming commonplace, containing millions of free 3D models that can be used for this purpose. However, they are typically unorganized and do not lend themselves to such inspiration-seeking exploration. The unorganized nature of such collections, and the modeller's goal of inspiration-driven exploration reveal two important challenges. The first is, how to organise and effectively navigate such model collections. The second and perhaps more important is, how to provide previews of possible shapes that might be missing from such collections, so that the modeller can gain some inspiration of new shapes to create. So far these two challenges have been treated separately in the literature. Our belief is

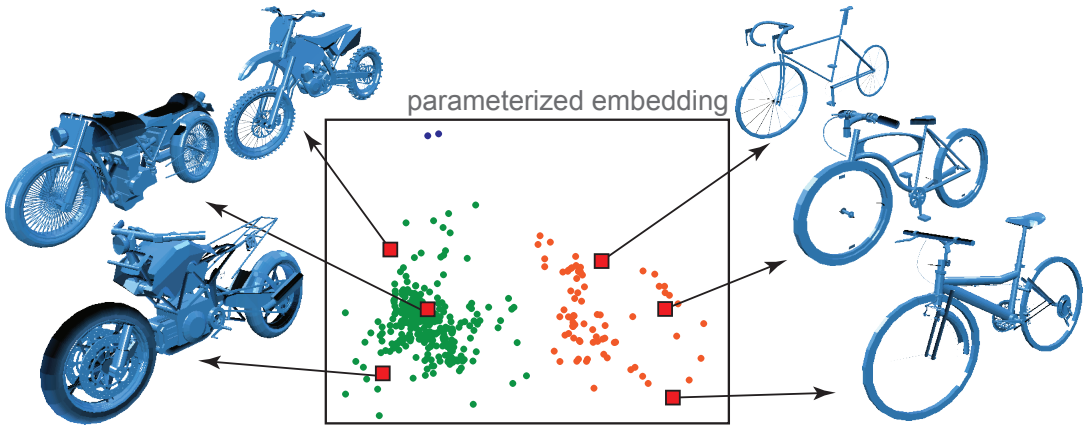


Figure 4.1: We analyze unorganized model collections using template-based abstractions to create a low-dimensional embedding of the underlying shape space. The user can then explore this low-dimensional space to create novel shapes by clicking in the empty regions (for example, the red rectangles). In each case, a model was synthesized by deforming and recombining parts from neighbouring models.

that they can be better addressed in a coupled manner. We propose a solution to both challenges by introducing a method to effectively organise such model collections, that provides fast and intuitive exploration of existing shapes, as well as rapid overview of novel shapes that can be synthesized (see Figure 4.1). Our hypothesis is that by modelling shape structure in terms of shape part positions and dimensions we can parameterize the shapes inside a collection, which will allow us to explore the collection in a meaningful way and synthesize novel shapes in a coupled manner. The challenge is how to construct this parameterization as shape collections have no information on how shapes are decomposed into parts and how these parts correspond from one shape to another shape.

There are a few common paradigms for exploring model repositories. Previous approaches have ranged from text-, shape- or sketch-based retrieval [24, 31], to template-based exploration [73], fuzzy correspondences [53] and qualitative exploration using dynamic embedding [44, 58]. Shape- and sketch-based retrieval methods rely on the user having a clear image of the target shape and repositories containing sufficiently similar shapes, an assumption which is often violated. Although the rest of the methods provide intuitive local exploration of existing models, they do not support synthesis of novel shapes.

In the context of shape synthesis, since the Modeling by Example system [30] introduced the idea of cutting parts from existing shapes and merging them into new shapes, a number of methods have tried to solve individual challenges of this part-based synthesis paradigm. The main challenge, that of locating and extracting appropriate

parts, has been tackled using geometric [60] and relational [106] similarity, or by sampling probabilistic graphical models [15, 48]. Such methods, however, do not provide the user with a high-level preview of the space of possible models that can be synthesized. Additionally, the user is required to have a clear idea of the final shape in order to effectively retrieve suitable models or model parts. This is often difficult, especially in collections with significant model variations (see Figure 4.4).

Most similar to our work, Talton et al. [89] propose an interface that tightly couples exploration and synthesis for parameterized model families. However, this method assumes a compact parameterizable design space [1, 96], which quickly becomes infeasible for raw model collections with diverse shape variations.

Our approach directly integrates exploration with synthesis for large and diverse shape collections. The key to our method is extracting a template-based parameterizable shape space that effectively factors out and encodes (part-) deformation across the collection. An offline analysis reveals the structure of the shapes in the collection by fitting a set of deformable templates made of primitive boxes to each input shape. The automatically-learned templates produce co-aligned and co-segmented models with known part deformations. An online hierarchical encoding step is then used to extract local 2D embeddings of the underlying shape space.

The embeddings facilitate fast exploration of existing shapes by revealing the different shape groups inside the collection and exposing the main modes of variability for each group. At the same time, they also provide fast and intuitive synthesis of novel shapes. We performed a user study to evaluate our system on several large model collections and compared with alternative systems. Figure 4.1 shows some of the models created by the different users of our system.

In summary, our contributions are:

- A template-based parameterization of shapes that allows a hierarchical organization of large model collections by embedding part-aware shape descriptors into low dimensional spaces, providing a basis for coupled shape exploration and synthesis;

- a novel exploration interface based on the embedding that is able to reveal groupings of shapes in separate clusters and summarize the main modes of variation inside each cluster; and

- a novel tool for the synthesis of new shapes from existing shapes, seamlessly tied to the exploration interface.

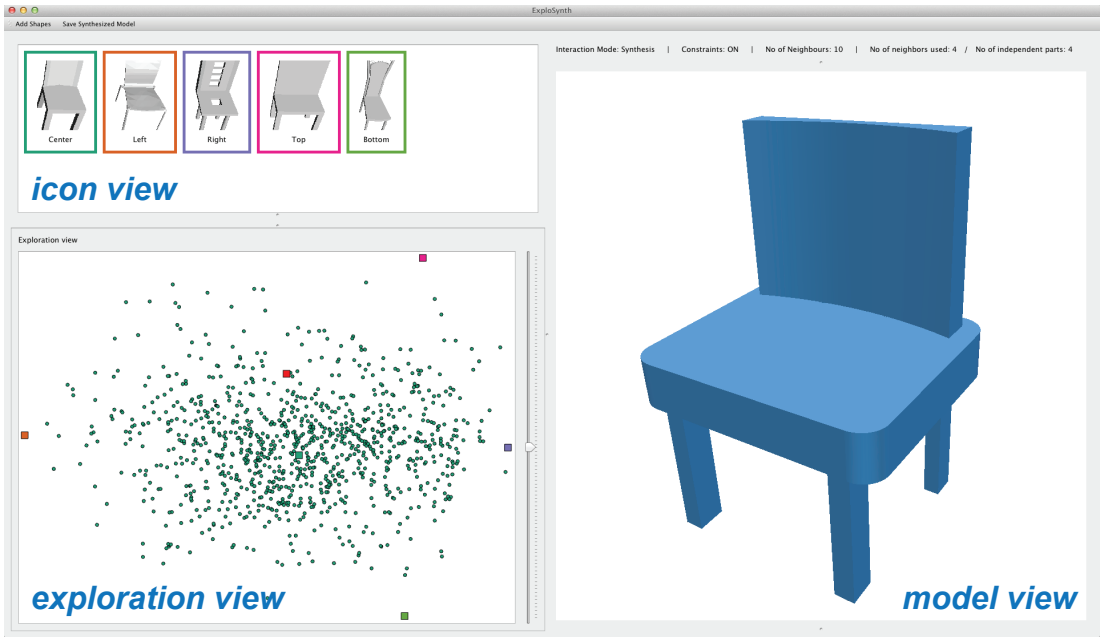


Figure 4.2: Our system comprises of an *exploration view* to show the 2D embedding of the input models; an *icon view* to show representative models for the current group(s); and a *model view* for showing the abstracted or synthesized models created using our system.

4.2 Overview

Our method takes as input a collection of semantically-related man-made shapes, for example, a set of 3D chair models that can be downloaded from the web. An offline analysis is then run on the input shape collection to compute the template-based abstractions and descriptors for each shape (see Section 4.3). At runtime, an interactive system is used to load the pre-analysed collection and allow concurrent exploration and synthesis of shapes.

The system interface, shown in Figure 4.2, is split into three panels:

- the *icon view* that presents the different representatives for selecting among different shape groupings;
- the *exploration view* that presents a set of embedded points, one for each shape among the current selection of models; and
- the *model view* that presents the currently synthesized model, either abstracted as a set of box proxies, or as a part-based synthesized model.

Input models are embedded in a 2D space using PCA on the template-based descriptors calculated in the offline stage, such that models with similar deformations end up

as neighbours, while dissimilar models are embedded to distant points (see Section 4.3). The embedded points are automatically clustered using mean-shift clustering, and the user is presented with a high-level overview of representative shapes for each cluster in the *icon view* panel, with different colors indicating different clusters. When the user selects one of the clusters, the corresponding models are re-embedded and the process repeats. In this way, the user can traverse the hierarchically-organized models that are grouped based on their descriptor similarity. Upon selecting a cluster that cannot be split into more sub-clusters, one can study the main variation modes across models in that cluster.

More importantly, the user can preview plausible shapes in the empty regions of the shape space of the original input collection. As the user hovers over any empty region in the *exploration view*, the system shows a set of box proxies in the *model view* which abstract a shape that can possibly be synthesized in that location of the shape space. In this way, we provide a quick glimpse of possible models by exposing the space spanned by the input collection. Constraints such as symmetry and contact are directly preserved.

Once the user is satisfied with one of the abstracted shapes, she can click in the location currently being hovered in order to view a plausible part-based synthesized shape corresponding to that location. This novel shape is realised by first ranking corresponding parts from neighbouring models, choosing the top-ranked parts, deforming them to fit the abstract shape presented to the user, and finally combining them together. The user can also browse through parts from neighbouring models lower in the ranking by clicking on a part of the synthesized model.

4.3 Method

Starting from an unorganized collection of 3D shapes, the goal is to allow the user to explore the shapes inside the collection and provide a preview of the possible shapes that can be synthesized by appropriately combining parts from different input models. In order to provide such exploration and synthesis capabilities, we have to overcome a few key challenges:

- the input models typically have large shape variations that in turn obscure any inherent consistency across the collection;

- the models do not come with any consistent segmentation; and

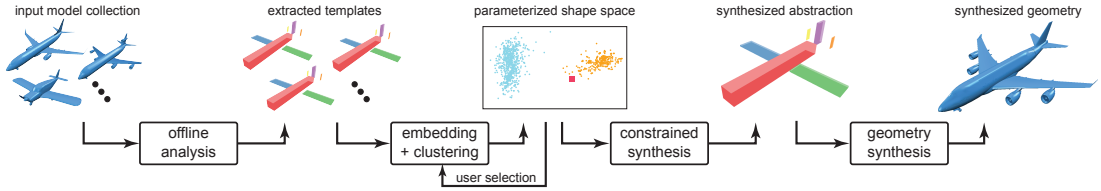


Figure 4.3: We start from a collection of semantically-related shapes and first analyse them in an offline step, solving for alignment, correspondence, segmentation and a deformable template, in order to obtain template-based shape descriptors. Then, in an interactive interface, we compute 2D embeddings and clusterings of the shape descriptors to reveal variations and shape groups that in turn guide the user selections. The user can quickly explore the template-based abstract shape space by hovering the mouse over the 2D embeddings, and click in empty regions of the space to synthesize new shapes based on parts selected from neighbouring shapes and deformed appropriately.

- the collections typically include thousands of models, making realtime analysis non-trivial.

For example, a visual investigation of the models in Figure 4.4 does not immediately reveal the space of possible models that can be realized by combining parts from the input models. We overcome these challenges by computing 2D embeddings of the models, which facilitate exploring existing shapes and previewing possible part-based synthesised shapes. The embeddings help to identify *which* models can be combined; *what* respective parts can be combined; and finally *how* the parts can be deformed to produce a plausible model. Below are the key steps of the method (see Figure 4.3).

4.3.1 Initial analysis

First, in an offline phase, we analyze the input collection of models $\{M_1, \ldots M_N\}$, using the method from Kim et al. [54]. Starting with an initial template represented as a set of axis-aligned box proxies, the method jointly optimizes for alignment, part segmentation with point-wise surface correspondence, and a compact deformation model to best explain the input shapes. The deformation model assumes that each shape can be approximated with a set of box-like parts that differ in position and scales. The method is an iterative one and interleaves three steps: fitting, where every template is deformed to fit to every shape; clustering, where shapes are assigned to their best-fitting template; and refinement, where the set of templates is updated, with new templates spawned for clusters not well explained by existing templates. The output of the analysis is a small set of part-based templates, along with the alignment, segmentation and deformation parameters that match each of the input models M_i to their corresponding template.

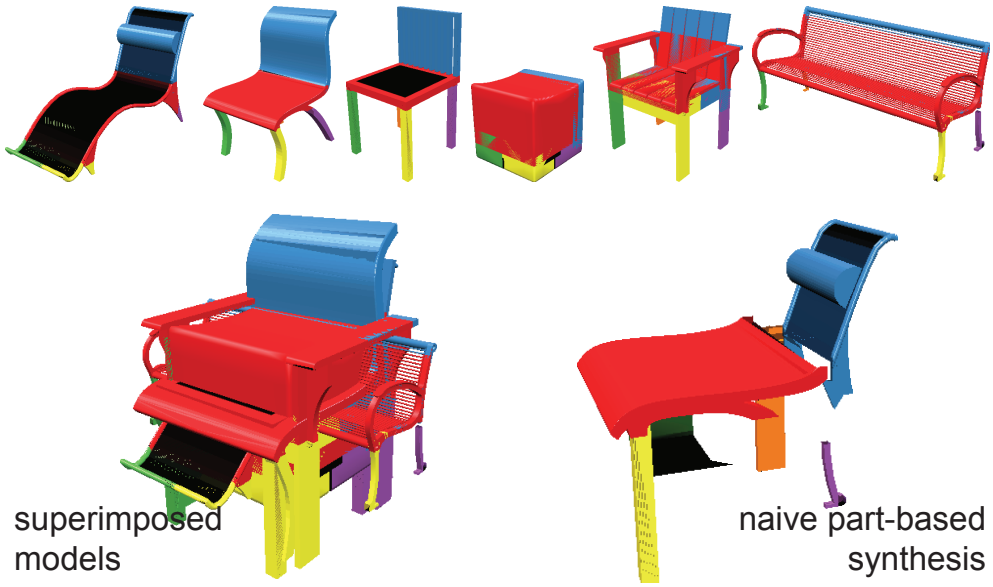


Figure 4.4: Combining a random selection of chair models (top), even when they are consistently segmented, is challenging. The models have different proportions of parts that make a part selection based on visual inspection of the superimposed models (bottom-left) confusing and can easily result in meaningless part ensembles (bottom-right). Instead, we expose a constrained and intuitive part-based model shape space for easy exploration and synthesis.

4.3.2 Abstracted encoding

Based on the extracted distribution of the template parameters, we refine the segmentations of the individual models M_i. We assume that each model comes with multiple disconnected components, which is true for most models in Trimble 3D Warehouse that we used in our experiments (see Figure 4.5-left). If this assumption does not hold, we simply assign each triangle to its nearest box. Let $\{p_1, p_2, \dots\}$ be the set of components for model M_i. Our task is to associate each component with a template box. This is essentially a labeling problem, where each p_i can be assigned to a set of t candidate boxes $\{l_1, \dots, l_t\}$. We formulate the labeling as a Markov Random Field minimization:

$$\{l_i\}^\star := \arg\min_{\{l_i\}} \sum_i E(p_i \to l_\alpha) + \sum_{i,j} E(p_i \to l_\beta, p_j \to l_\gamma) \tag{4.1}$$

The unary term $E(p_i \to l_\alpha) := vol(B_{p_i \cup l_\alpha}) - vol(B_{l_\alpha})$, where $vol(B_{p_i \cup l_\alpha})$ denotes the bounding volume of component p_i and template box l_α, and $vol(B_{l_\alpha})$ denotes the bounding volume of template box l_α, measures the increase of bounding volume of the template box l_α when component p_i is assigned to it. The pairwise term $E(p_i \to l_\beta, p_j \to l_\gamma)$ measures the penalty when two neighboring components (based on shortest distance between them) are assigned different labels (set to 1e-5 in our tests). In the

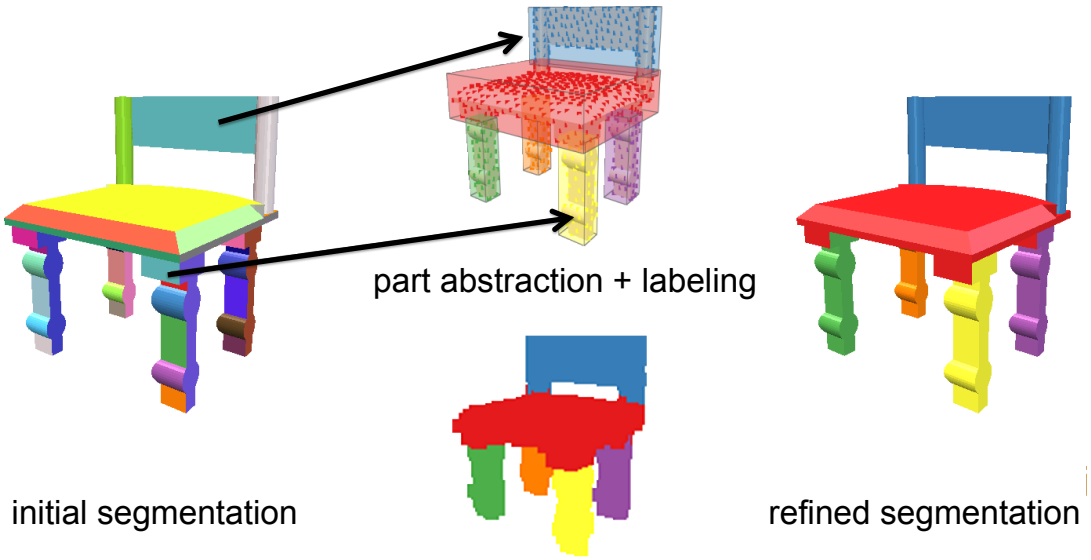


Figure 4.5: In the case of models with multiple components (left), we use the extracted part distributions obtained from the shape collection [54] to obtain an initial point labeling (middle-bottom) and part abstraction (middle-top). We refine the segments using a labeling optimization (right).

end, for each model we get a set of abstracted boxes, each enclosing a part of the input model (see Figure 4.5-right).

Let there be t different parts discovered across all of the extracted templates in the initial offline analysis. We represent each input model M_i as a configuration vector $X_i \in \mathbb{R}^{6t}$, where each (axis-aligned) template box is represented by its centroid $\mathbf{c_i}$ and its dimensions $\mathbf{s_i}$, i.e., its length/breadth/height. Parameters corresponding to missing parts are set to zero.

We also detect the potential relations among the individual parts, i.e., symmetry and contact relations, and propagate the information to their associated templates, which are later used in the constrained synthesis phase. Note that the relations should be unified across templates within a given family. To address this in a consensus stage, we collect the relations among all the templates and use a greedy selection strategy to filter out falsely detected relations. Improperly identified or conflicting relations can be manually corrected, although advanced automated methods can potentially be used [70].

4.3.3 Efficient embedding

Both exploration and synthesis require a notion of *neighbourhood* among the models. We use the abstraction obtained above to define such a dissimilarity distance between model pairs M_i and M_j as follows: $d(M_i, M_j) := \|X_i - X_j\|$. Note that since we also have

Data: Input model collection $\mathbf{M} := \{M_1, \ldots M_N\}$.
Result: Embedding coordinates of the selected models.
1. Analyze the input collection $\mathbf{M}$ in an offline stage [54];
2. For each model $M_i \in \mathbf{M}$, refine initial segmentation to obtain configuration vector $X_i \in \mathbb{R}^{6t}$;
3. Set selection $\mathbb{M} \leftarrow \mathbf{M}$;
4. **while** *new selection* $\mathbb{M}$ *available* **do**
   i. Randomly pick n models $\tilde{M}_j$ for $j = 1, \ldots n$ from $\mathbb{M}$ as landmarks;
   ii. Construct distance matrix $\mathbf{D}_{n\times n}$ with elements $d_{i,j} := d(\tilde{M}_i, \tilde{M}_j)$ for $i, j = 1, \ldots n$;
   iii. Compute MDS embedding of the landmark models $\tilde{M}_j$ using $\mathbf{D}$ to obtain $\tilde{Y}_j \in \mathbb{R}^2$;
   iv. For all $M_i \in \mathbb{M}$ and $M_i \notin \{\tilde{M}_j\}$, find its k nearest neighbor models among the landmark models and use distance-based interpolation to obtain embedded coordinates $\{Y_i\}$.
   v. Compute (linear) basis vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ for inverse mapping of embedded coordinates to configuration vectors.
   vi. Apply mean-shift clustering on the points $\{Y_i\}$;
   vii. Update selection $\mathbb{M}$ and repeat hierarchically by returning to step #4 ;
end

Algorithm 4.1: Iteratively embed a selection of models $\mathbb{M}$, which is then used for exploration and synthesis.

the parameter distributions, one can instead use Mahalanobis distance. Thus, similar models have near-zero dissimilarity score, while dissimilar model pairs get high scores.

We use the similarity values to embed the models into a low-dimensional parameterized space. One option is to construct a $N \times N$ matrix with all the pairwise similarity values (e.g., $\exp(-d(M_i, M_j)^2/2\sigma^2)$) between the input models and compute its spectral embedding [53]. Such a direct computation, however, can be prohibitively expensive (i.e., $O(N^3)$) for large model collections.

Instead, we propose a sampling-based approach to efficiently build an embedding of the models (see Algorithm 4.1). The key observation is that the embedding is largely dictated by the members from different (unknown) clusters (c.f., [22]). Hence, working with a random sampling of models as representatives yields an approximate embedding. Note that we use the approximate embedding only when the number of models is large, otherwise we perform the embedding with all the selected models.

We start by picking at random n landmark models, say $\{\tilde{M}_j\}$, from the current set of models $\mathbb{M}$. Using the n landmark models, we compute their pairwise distance matrix $\mathbf{D}_{n\times n}$ and embed the models to $\mathbb{R}^2$ using multi-dimensional scaling (MDS) based on singular value decomposition (SVD), to get $\{\tilde{Y}_j\}$. For any other model $M_i \in \mathbb{M}$, we

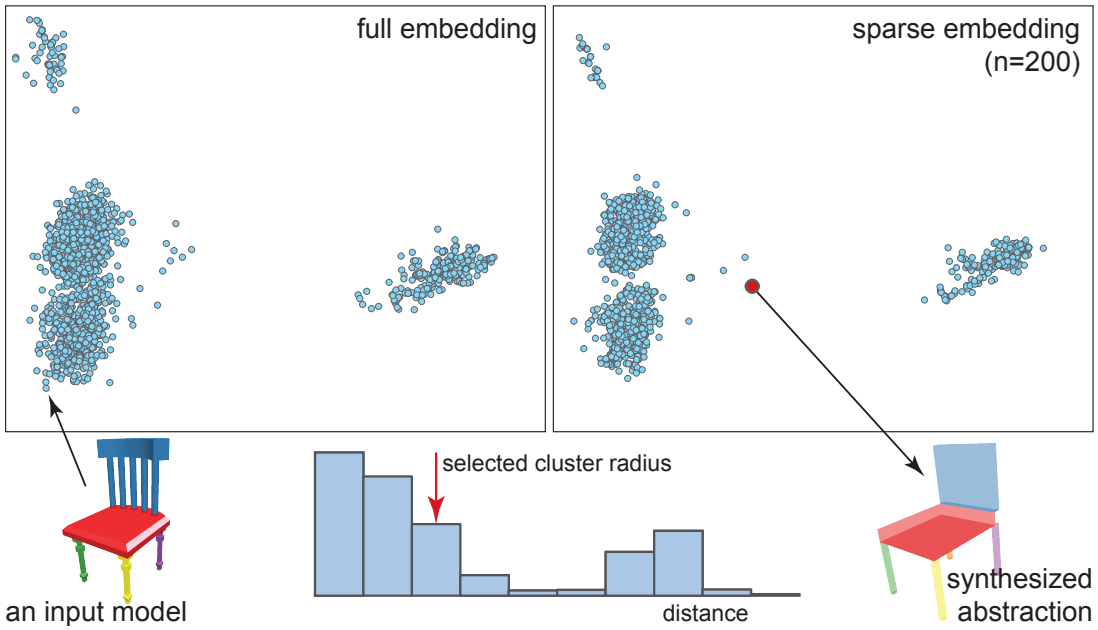


Figure 4.6: We embed the input models using their corresponding fitted template-based abstractions. We perform an efficient landmark-based embedding and analyze the points to obtain a parameterized template abstracting the underlying shape space. As the user navigates the embedded space, the extracted variation modes are used to lift the points (shown in red) back to high-dimensional configuration vectors to synthesize template abstractions. The distribution of the pairwise distances between the embedded points is used to estimate a suitable clustering radius for mean-shift clustering.

compute its k nearest neighbors among the landmark models. We then interpolate the embedded coordinates of the landmark models to compute the embedding of M_i, i.e., $Y_i \leftarrow \sum_{j=1:k} w_j \tilde{Y}_j / \sum_{j=1:k} w_j$, where $w_j := \exp(-d(M_i, \tilde{M}_j)^2 / 2\sigma^2)$, with σ set to the diameter of set $\{\tilde{X}_i\}$ and $\tilde{M}_j$ denoting the j-th closest of the landmark models.

The above embedding method has a complexity of $O(n^2 + Nk \log n)$. For example, for a chair dataset with 2036 models, the sparse embedding takes about 0.6 sec, which is about 12 times faster than full embedding (see Figure 4.6).

4.3.4 Abstracting missing shapes

At this stage, we have mapped the initial coordinates $\{X_i\} \rightarrow \{Y_i\}$. We solve for the dominant linear variation modes $\mathbf{e}_1$ and $\mathbf{e}_2$ such that $Y_i \approx [(X_i \cdot \mathbf{e}_1), (X_i \cdot \mathbf{e}_2)]$ for all $i \in [1, N]$ using a least squares formulation. Now, given any point (α, β), we can lift up the point (from the empty region) to the configuration vector as $(\alpha, \beta) \rightarrow \alpha \mathbf{e}_1 + \beta \mathbf{e}_2$, thus providing an abstracted model X as a preview for the empty region.

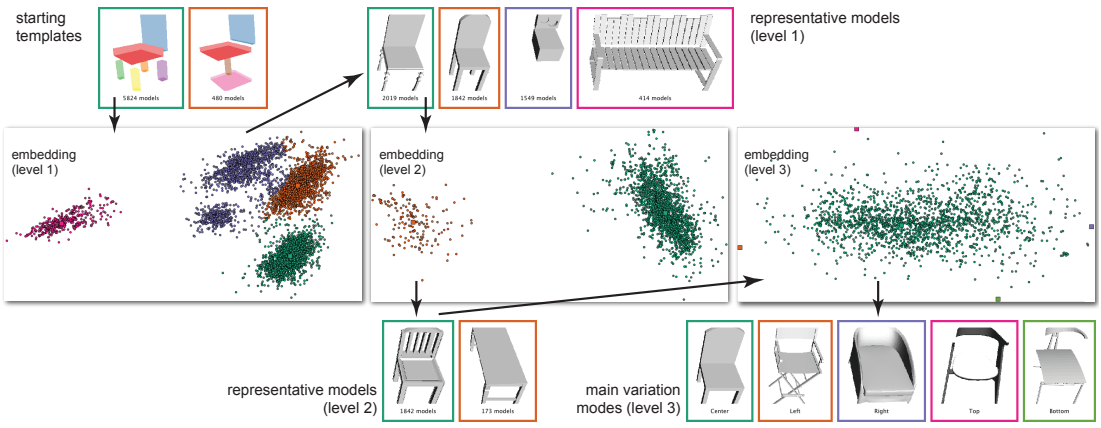


Figure 4.7: A typical hierarchical exploration session using our interface. After initial analysis, the system displays the top level templates (top-left). As the user selects the green mode, the member models are embedded (level 1) and 4 dominant clusters are detected. The user selects the next representative and its member models are re-embedded. When a single cluster is discovered, its representatives and dominant variation modes are shown (bottom-right).

4.3.5 Grouping shapes

We cluster the embedded points using mean-shift clustering [20] in order to organize the data into a hierarchy. We automatically select the clustering radius based on the histogram of the pairwise distances between the embedded points. In the case of points that can be grouped into multiple clusters, we can estimate a good clustering radius based on the first valley (if any) of the histogram (see Figure 4.6).

Each of the extracted clusters can then be re-embedded to extract new basis vectors (i.e., $\mathbf{e}_1$ and $\mathbf{e}_2$), eventually forming a hierarchical organization (see Figure 4.7). As the user selects one of the clusters, she zooms into that particular cluster in order to better study the fine-scale variations. The mean-shift clustering procedure gives a small number of compact clusters at each level of the hierarchy. For example, in Figure 4.7, a dataset of chairs gives five clusters at the top level, and two clusters at the second level, when one of the top-level clusters is selected.

4.3.6 Constrained abstract shape synthesis

A direct derivation of box configuration $X = \alpha\mathbf{e}_1 + \beta\mathbf{e}_2$ from the embedding space can result in models that deviate from a semantically valid one, e.g., symmetry being broken, part-to-part contacts being lost, etc. We project the box configuration parameters to the valid shape space using a constrained optimization. We observe that many relations of interest (c.f., [70]) simply amount to linear constraints involving parameters of the configuration vector, e.g., contact, reflective symmetry about known plane etc. Our goal

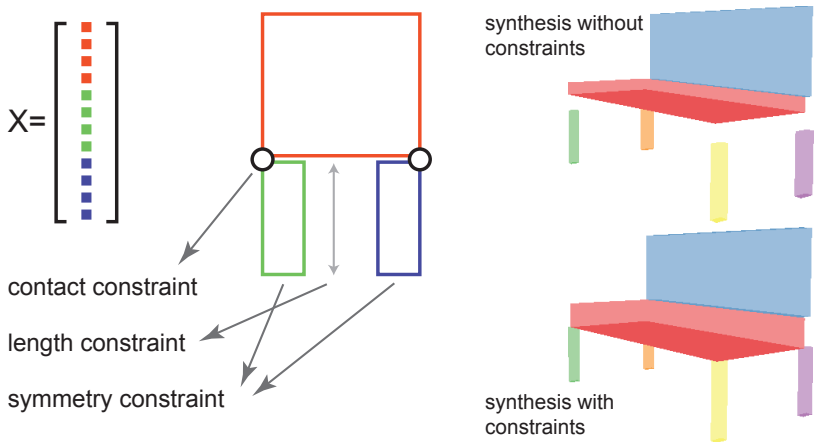


Figure 4.8: Illustrative example of the different constraints handled in our framework. (Left) In this 2D example, the configuration vector $X \in \mathbb{R}^{12}$ represents the abstracted model with 3 parts. For example, the two contact constraints involve the orange-green and orange-blue boxes and hence the corresponding $f_i(X)$ involves the corresponding coordinates of X. (Right) Our system restores these constraints during the real-time exploration using a QP formulation.

is to obtain a new configuration $\tilde{X}$ such that potential symmetry and contact relations among parts are restored, while $\tilde{X}$ being as close to X as possible (see Figure 4.8). This amounts to solving the following minimization:

$$\arg\min_{\tilde{X}} \|\tilde{X} - X\|^2 \quad \text{such that} \quad f_j(\tilde{X}) = 0 \; \forall j = 1, \ldots c \tag{4.2}$$

where, $f_i(\tilde{X})$s is a set of c semantic constraints derived from the relations among the parts. We support three main types of relations: symmetry, contact and equal length. We detect these in the abstracted encoding step, as described in Section 4.3.2. Alternatively, the user can directly specify them once on the initial template in the offline analysis.

Symmetry. Let two boxes, represented as $(\mathbf{c}_i, \mathbf{s}_i)$ and $(\mathbf{c}_j, \mathbf{s}_j)$, where $\mathbf{c}_i$ is the center of the first box (likewise for the second box) and $\mathbf{s}_i$ is the scale vector (scale in x, y, z dimension) of the first box (likewise for the second box), be reflective symmetric with respect to a given plane. The corresponding constraints take the form:

$$((\mathbf{c}_i + \mathbf{c}_j)/2 - \mathbf{o}) \cdot \mathbf{n} = 0; \;\; (\mathbf{c}_i - \mathbf{c}_j) \times \mathbf{n} = \mathbf{0}; \;\; \mathbf{s}_i - \mathbf{s}_j = \mathbf{0} \tag{4.3}$$

where, $\mathbf{n}$ and $\mathbf{o}$ are the normal to the reflection plane and a point on the plane, respectively.

Contact. When two boxes are in contact, they share a common contact point. Between two boxes in contact, we assign the closest points as contact points. Thus, between two

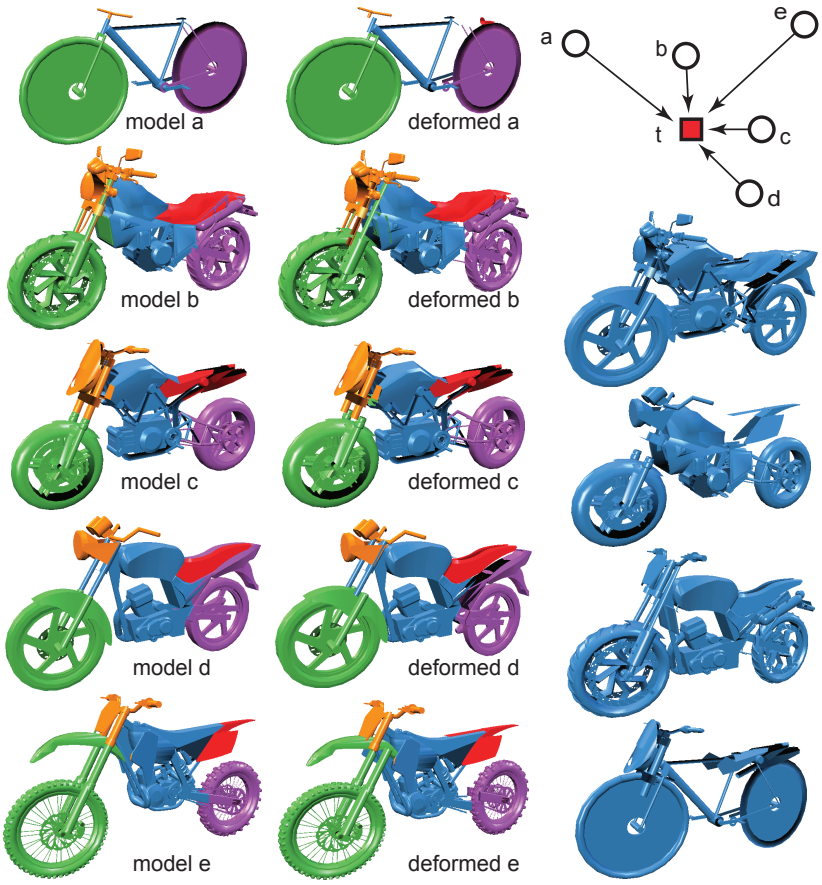


Figure 4.9: Our system allows to preview possible geometric realizations in an empty region around the embedded points (top-right). Each of the retrieved models (models a-e) is deformed to match the query configuration (indicated as a red box). Parts from the deformed models (middle) are then combined to create different plausible shapes (right).

boxes, the contact constraint takes the form: $\mathbf{c}_i + \mathbf{s}_i/2 = \mathbf{c}_j + \mathbf{s}_j/2$ (up to sign changes due to which corners get selected).

Equal length. In certain cases, we want pairs of boxes to have similar dimensions (for example, the legs of a chair should have equal height even if they are not symmetric). Since the abstracted boxes are axis-aligned, such a constraint simply takes the form: $\mathbf{s}_i^y = \mathbf{s}_j^y$, for example when equality along y-direction is desired.

The optimization in Equation 4.2 amounts to solving a quadratic program with linear constraints. As the user explores the configuration space extracted from the input collection, we perform the optimization to directly show the constrained solution (see Figure 4.8). Note that the input templates, i.e. $\{X_i\}$, do not necessarily satisfy the constraints. However, we do not project them to the constrained space since the parts taken from the input models are later deformed to a constrained box model, as described next.

4.3.7 Part-based geometric shape synthesis.

As the user moves the mouse cursor over a point (α, β), our system shows the corresponding box model, $\alpha \mathbf{e}_1 + \beta \mathbf{e}_2$, which has been constrained to produce abstracted box model $\tilde{X}$. Each such feature vector $\tilde{X} \in \mathbb{R}^{6t}$ represents concatenated parameters for t boxes $\tilde{X} = [x_1, \ldots, x_t]$, where x_i is a 6-dimensional vector that encodes position and dimensions of the i-th box. When the user is satisfied with the coarse arrangement of parts, she can click and lock the system to the current box model. This immediately prompts our system to *fill* the boxes with corresponding geometric parts from the k nearest neighbours of $\tilde{X}$ in the 2D space. The goal is to select parts that are to be least deformed in order to *fill* the boxes. Therefore, for each of the abstracted model's boxes x_i, we select the corresponding geometric part $\bar{x}_i$ with the smallest distance, from one of the k nearest neighbours, i.e. $\arg\min_{\bar{x}_i} \|x_i - \bar{x}_i|$. The selected set of geometric parts are deformed, i.e. translated and anisotropically scaled, so that their box matches the corresponding abstract model's box. The user can continue exploring and visualizing alternative part arrangements drawn over the selection, or refine the choice of selected parts by clicking on a corresponding part $\bar{x}_i$ in the *model view*. The click prompts the system to cycle through the candidate geometric parts taken from the k nearest models M_j, according to the distance $\|x_i - \bar{x}_i|$ (see Figure 4.9). One can cycle through all k possible geometric parts to fill a box of the abstract model in this manner.

4.4 Evaluation

In this section, we evaluate our template-based shape parameterization and the proposed shape exploration and synthesis tool. First, we describe the data and experimental setup, and evaluate the performance of our algorithm on diverse datasets obtained from Trimble 3D Warehouse. While our coupled analysis, exploration, and synthesis framework is the first of its kind, we compare parts of our system to state-of-art shape synthesis methods from Chaudhuri et al [15] and Jain et al. [46], and evaluate our constrained synthesis algorithm.

4.4.1 Datasets

We tested our framework using Trimble 3D Warehouse models obtained from Kim et al.'s method [54]. We selected a subset of models such that the template fitting energy is below 30, and hence our final analysis included 2062 chairs, 636 airplanes, and 114 bikes.

As an additional dataset, to compare with the authors' implementation of the part-based synthesis method of Chaudhuri et al. [15], we used their manually-segmented dataset of 100 airplanes from Digimation ModelBank.

4.4.2 User feedback

Several participants with very little or no prior 3D modelling experience used our system to explore existing model collections and create new shapes. The feedback was positive and a variety of different models were created (see Appendix B for the full set of results). Figure 4.10 shows a sample of models created by users of our system. Most people appreciated the ability to quickly obtain a high-level overview of the modelling space, refine the selection to a region via the hierarchical navigation, and finally obtain an immediate preview of shapes to be synthesized.

Some users complained that the final models were at times disconnected, making them look unrealistic. This is expected as we only enforce contact constraints at the abstracted level of the boxes, and not on the original geometry. Section 4.4.7 gives details as to how this problem can be alleviated.

4.4.3 Baseline comparison

We tested our initial hypothesis that the proposed template-based parameterization can be used for synthesizing meaningful novel shapes, via a user study. The hypothesis of the user study was that our system can generate more diverse sets of shapes, compared to the original shape collections, thus proving that the goal of exploring and visualizing missing shape variations for providing inspiration to modellers is achieved. For this purpose, we evaluated our system on the chairs, airplanes and bikes datasets from Trimble 3D Warehouse. We asked 8 volunteers from a computer science department to use our system to perform an open-ended task on each dataset:

T1: *Create the most diverse set of 5 shapes.*

After a short demo instructing each person how to use our system and letting them use the system for around 5 minutes, each of the 8 users created 5 chairs, 5 airplanes and 5 bikes, for a total of 40 chairs, 40 airplanes and 40 bikes. The order of completing task **T1** for the three datasets was randomised between users. Next, we validated the results, by comparing the shapes created by the volunteers to a random selection of shapes from the original datasets. Similarly to the previous work of Chaudhuri et al. [14, 15], we recruited a different group of volunteers to compare randomly selected pairs of results created by users of our method and models belonging to the random selection. Using

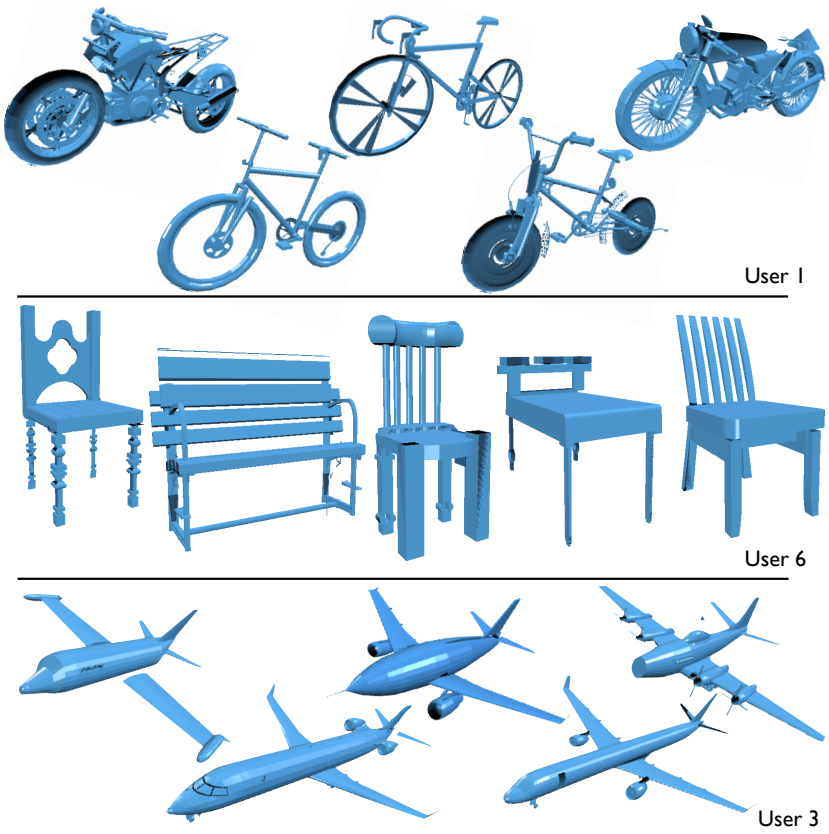


Figure 4.10: Sets of models created in our user experiment. Please refer to Appendix B for the full set of results. Our system enables rapid synthesis of diverse models.

a web interface, we presented two randomly chosen groups of results; 5 results from one of our volunteers and 5 of the randomly selected models from the corresponding original dataset. We described task **T1** and asked:

- if the shapes look plausible; and
- if the shapes look diverse.

For each question, the evaluator had an option of selecting one of the groups, both, or none. The first question was designed to evaluate whether our system trades plausibility for diversity in the synthesized shapes.

Table 4.1 shows the results of the user study. In Figure 4.10, we present user-created models in each category. All user-created models are provided in Appendix B. According to Table 4.1, our method can indeed generate more diverse sets of shapes compared to all the original datasets, which proves the hypothesis of the user study. Our method also allows very fast creation of such diverse sets of results, with the average time to complete task **T1** per user ranging between just 3 and 5 minutes for each dataset. Note however, that our method's results are less plausible than the original models for

Dataset	Voting						Time (min)
	Plausibility			Diversity			
	Our	Random	None	Our	Random	None	Our
bikes	77	64	2	69	58	11	5
chair	48	104	4	100	19	5	4
planes	53	81	2	68	48	6	3

Table 4.1: User study on Task **T1** comparing our method with a random selection from a dataset. Voting indicates number of time users voted for our method vs random selection (where votes for both are summed with individual votes), and timings are in minutes.

all datasets except the bikes. This can be explained by the fact that final models look disconnected at times, since we only enforce contact constraints at the abstracted level of the boxes.

4.4.4 Comparison to Chaudhuri et al. [15]

We tested our initial hypothesis further, by comparing our system to a state-of-the-art interactive synthesis tool from Chaudhuri et al. [15] via another user study. The system from Chaudhuri et al. [15] is a suggestion-based modelling tool which takes advantage of a probabilistic model trained offline using manually segmented and labelled shapes to provide the most useful suggestions for parts related to the shape being modelled. Compared to our automatic coupled exploration and synthesis tool, Chaudhuri et al.'s [15] system is interactive and seeks to accelerate part-based modelling by providing the most relevant shape parts to the modeller according to the shape being built. It also relies on low-level geometric shape descriptors, unlike our method which only relies on a coarse template based parameterization.

Using the same diversity hypothesis as before, we compared against this system by focusing on high-level exploratory synthesis tasks. We used the authors' implementation of the method on their dataset of 100 consistently manually-segmented airplanes from the Digimation ModelBank. Our initial analysis was modified to create deformable templates from manually-segmented models and fit the templates to all models without changing the segmentation. We recruited a different group of 10 volunteers with a background in computer science, asking them to perform task **T1**, and:

T2: *Create an airplane that is best suited to win a 'dogfight' (one-on-one aerial combat) in a computer game. This is the same task as in [14].*

Each user performed both tasks using both systems in a consistent order (**T1, T2**), while we randomly permuted the order in which systems were used. Figure 4.11 summarizes the results produced by the same user (selected at random) in both systems. All user-

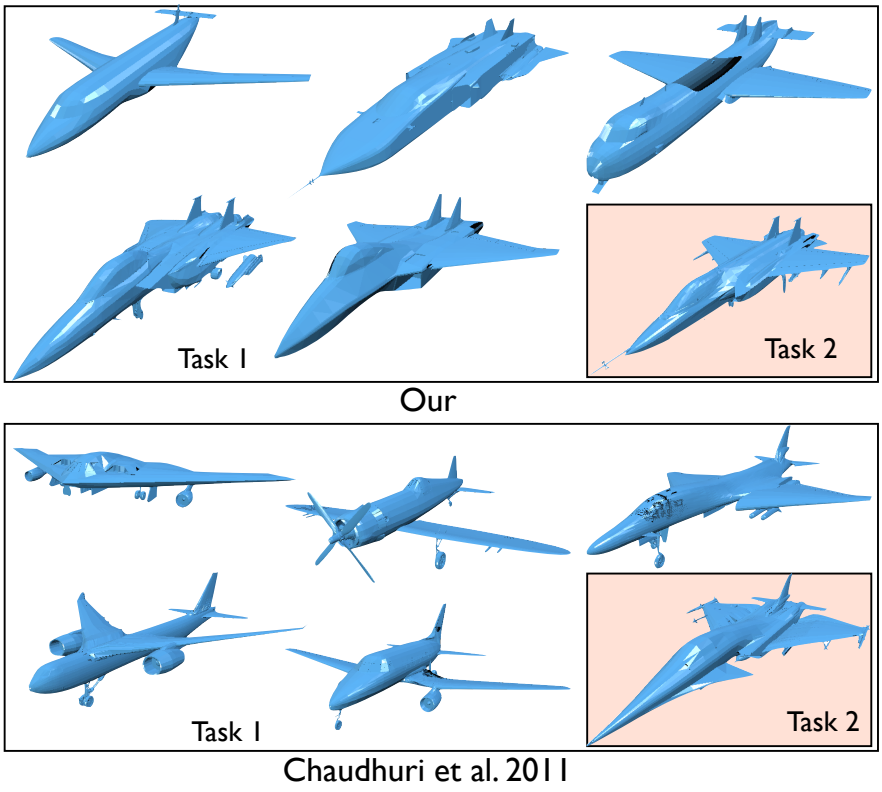


Figure 4.11: Example models created by User 1 (picked at random) for comparison to Chaudhuri et al. [15] system. Note that both sets of models created for task **T1** contain diverse and plausible shapes, as was requested in the task.

created models for both systems are provided in Appendix B. We validated the results using another group of volunteers in the same way as in the baseline comparison. For task **T1** we presented groups of 5 airplanes created from our system and 5 airplanes created from the Chaudhuri et al. system [15], asking people whether the airplanes are plausible and whether the user succeeded in her goal for the task, i.e. whether the airplanes are diverse. For task **T2** we presented the airplane created from our system and the airplane created from the Chaudhuri et al. system [15], asking people whether the airplane is plausible and whether the user succeeded in her goal for the task, i.e. whether the airplane would win the aerial combat. We summarize the statistics in Table 4.2.

Although users of our system often produced implausible models when aiming at diversity in task **T1**, resulting in lower plausibility scores for our method compared to Chaudhuri et al. [15], the resulting sets of models were deemed comparable in diversity, with our results getting 81 votes compared to 131 for Chaudhuri et al. [15], but with a relatively large number of uncertain votes (38 votes for None). Furthermore, we found that once the users became familiar with the space of shapes, they could very

Task	Voting						Time (min)	
	Plausibility			Task Accomplished?				
	Our	Chaudhuri	None	Our	Chaudhuri	None	Our	Chaudhuri
T1	67	174	17	81	131	38	6	14
T2	132	150	13	158	180	12	2	4

Table 4.2: Comparison to Chaudhuri et al. [15], tasks **T1** and **T2** were accomplished with two different interfaces. Voting columns indicate number of time users voted for results produced with our method vs their approach (where individual votes are summed with votes for both methods).

rapidly synthesize airplanes for task **T2** that are comparable to results produced by Chaudhuri et al. [15] in plausibility, as well as achieving the goal of the task which was to create a combat-winning airplane. Note that the average time needed by every user to complete both tasks with our system was about half the time they needed using Chaudhuri et al.'s [15] system, which demonstrates that our system can be used to complete such exploratory modelling tasks much faster than competing systems. We find the diversity result particularly surprising since our embedded space is based on a coarse box-abstraction rather than geometric details. Even then, our system exposes interesting high-level part placement variations.

4.4.5 Comparison to Jain et al. [46]

Our method can also be used to interpolate between pairs of shapes, as in Jain et al. [46]. The system from Jain et al. [46] takes a pair of shapes as input, decomposes them into parts based on connected-component analysis, builds a relation graph between the parts and tries to match corresponding parts that can be safely exchanged between the two shapes based on the relation graphs. A continuous slider then is used to control the amount of parts taken from the source and target shapes, similar to an interpolation, where intermediate shapes can be created combining parts from both shapes. Unlike our method, their system only relies on pairs of shapes and no deformations are applied to parts coming from the source and target shapes, causing visible artefacts when the two shapes are not well-matched in terms of part proportions. In Figure 4.12 we demonstrate two interpolations produced by our method, and our simplified implementation of Jain et al.'s [46] method.

In our simplified implementation of Jain et al.'s [46] method, we manually select two models in the 2D space, and use the line joining them as the interpolation space. By moving along this line and forcing the system to only select parts from these two models according to Jain et al.'s [46] strategy for picking parts, we simulate the output

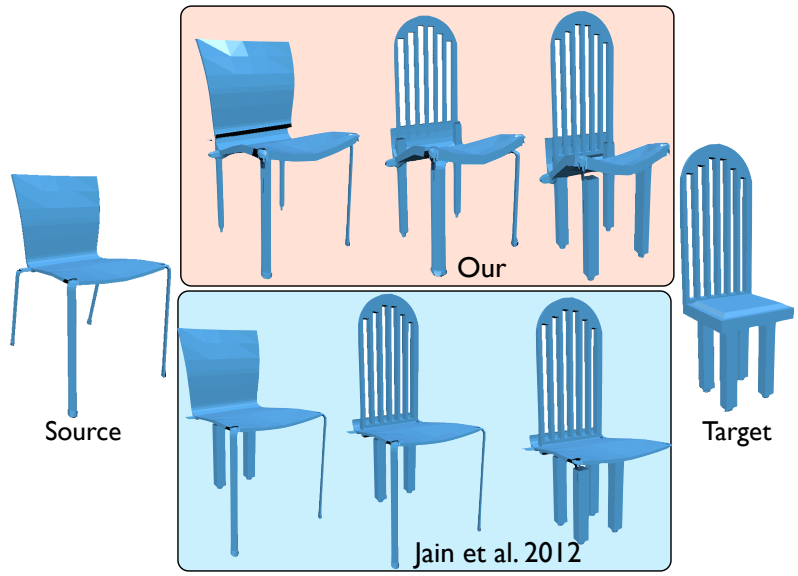


Figure 4.12: We evaluate our method in a shape interpolation scenario such as in Jain et al. [46]. First (middle-left), the back legs of source are replaced with the back legs of target. Then (middle-center), the back of source is replaced with the back of target. Finally (middle-right), the front legs of source are replaced with the front legs of target. Note that our method is more robust to strong deformations because it uses the full shape space to model the shape variations and thus deforms parts appropriately to fit a particular point in shape space, unlike Jain et al. [46] who do not deform parts.

of their system. The difference is that in our system we compute the necessary part deformations for synthesizing a plausible shape at any point of our shape space, in contrast to Jain et al.'s [46] system, which does not deform shape parts but only enforces contacts based on a spring-mass model.

A visible advantage of our method is that it produces more plausible shape variations for intermediate shapes where the deformation is high (e.g., look at the chair legs). The quality comes from: (i) appropriate part scaling to facilitate model assembly; and (ii) neighboring models contributing to the deformations of intermediate models, allowing richer variations.

Finally, unlike Jain et al. [46] who use contact and relation graphs to establish approximate correspondence between parts, our analysis automatically segments the input models and establishes part-level correspondence. It should also be noted that as our approach is data-driven, performance improves as the dataset grows. The example in Figure 4.12 is illustrated on a small dataset and hence it is an extreme case where our method can produce visible distortions. In practise, we did not observe this effect when working with the full datasets.

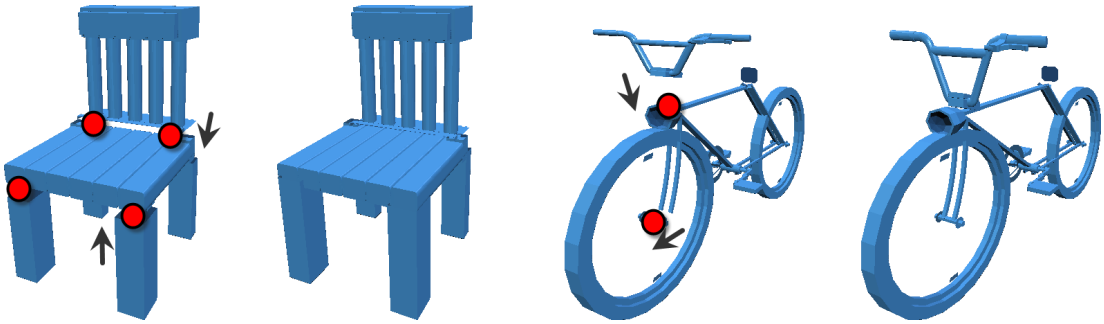


Figure 4.13: The synthesized models can be further refined using docker-based part deformation. In these examples, the parts of the chair and bike are brought back into contact based on nearest part dockers.

4.4.6 Constrained synthesis

We evaluate the quality of our constrained synthesis via a leave-one-out experiment. First, starting with model collection $\mathbf{M}$, we compute its embedding $\mathbf{e}_1, \mathbf{e}_2$ and then select a particular model M_i embedded as (α^i, β^i). We then remove the model M_i and re-analyze $\mathbf{M} \backslash M_i$ to obtain embedding $\mathbf{f}_1, \mathbf{f}_2$ and re-synthesize the model as $\alpha^i \mathbf{f}_1 + \beta^i \mathbf{f}_2$. We then compare the effect on the embedding of leaving out M_i.

We found the variation marginal and negligible in most cases. This is not surprising since our landmark-based embedding, and hence the extracted dominant modes are mainly dictated by the n random selected models. If the landmarks remain unchanged, the embedding does not change. Even with different landmarks the changes are small as shown in Figure 4.6. We further use the new embedding to reconstruct the box structure for the model M_i given its coordinates (α^i, β^i), and we found that reconstruction error is within 1% of original box dimensions.

4.4.7 Restoring contacts

The low plausibility scores we observed in the two user studies and while talking to users of our system can be explained due to our choice to enforce contacts at the abstracted level of template boxes. In a post-processing step, it is possible to restore contacts using an optimization procedure that minimizes the distance between compatible dockers detected on each pair of parts in contact, as shown in Figure 4.13. We expect that the plausibility scores for our results will reach the scores of Chaudhuri et al.'s system [15] if this post-processing step is applied. This step was not the focus of our work, as we have chosen to focus on parameterizing the shape space and creating a fast coupled exploration and synthesis tool that gives rapid inspiration to modellers.

4.4.8 Discussion

We evaluated our template-based model of structure and the associated exploration and synthesis method using 4 datasets ranging from 100-2000 models with 18 users synthesizing more than 500 novel shapes, validated by more than 30 different people, with 164 pairwise comparisons. Synthesized shapes were deemed plausible and diverse by users of our system.

In this Chapter we focused on abstracting the input models by axis-aligned box-templates, thus providing our first model for shape structure based on a template-based shape parameterization. Given that, our system is not suitable for datasets where the coarse structure of the shape does not correlate with its functionality or desired properties. In Chapter 5, we extend our work in this Chapter and we present a different, probabilistic model for shape structure which is based on oriented box templates. Such an abstraction is challenging to parameterize, but can provide better understanding of the underlying shape space, improved clustering and more meaningful exploration of shapes, as demonstrated in Chapter 5. We also demonstrate the application of this probabilistic model in intelligent and coupled shape editing in Chapter 5.

As our method relies on coarse box templates, it would also be interesting to investigate supplementing the shape space with part-based geometric descriptors. The challenge then is to appropriately combine the high-level box descriptors with low-level geometric descriptors. Fully evaluating the effect of enforcing contact constraints at the level of part geometry rather than at the level of template boxes is also something that remains to be done. Furthermore, it will also be interesting to use contact and relation graphs, as in Jain et al. [46] to maintain the structure of the synthesized shape variations.

Perhaps most importantly, it would be intriguing to explore the possible applications of our method to other areas. Since the extracted parameterized space provides an abstracted representation of the underlying shape space, problems like pose estimation, scan completion and object recognition can be investigated as projections to this parameterized space. In object recognition specifically, the template-based shape parameterization presented in this Chapter can be used to generate large amounts of synthetic 3D content to train object proposal estimators for depth images, as demonstrated by Zheng et al. [104].

Chapter 5

Meta-representation of shape families

5.1 Motivation

In Chapter 4 we presented a method for modelling shape structure using data from large unorganized shape collections. The method relies on a parameterization of a collection's shapes using axis-aligned box primitives. We demonstrated its application in exploring a shape collection and rapidly creating novel 3D content that can provide inspiration to a modeller. Such a model of structure is not suitable for all datasets as not all shape parts are well approximated by axis-aligned boxes, and relations between shape parts are not taken into account in that model. In this Chapter, we extend our work from Chapter 4 to model structure in a probabilistic manner, taking into account shape part relations and moving into high-level shape co-analysis. We illustrate how such a probabilistic model can be used to explore shape collections as well as intelligently edit shapes both individually and in a coupled manner.

High-level shape analysis goes beyond low-level analysis of the geometric properties of shapes and attempts to extract higher-level semantic information to aid in shape manipulation [70]. As part of this effort, single shapes, particularly man-made objects, have been analyzed to extract semantic relations that can be made useful in various applications. One example is in applying constraints on a shape editing process, thereby achieving an intelligent edit where the prescribed geometric characteristics of the manipulated shape are preserved [32, 66, 101, 107]. Ideally, the aim is to understand the *essence* of the family of shapes directly from their object geometry, in order to use this knowledge in the applications to determine the geometric shape and configuration of shape parts.

Analyzing an individual shape, however, is inherently limited as we are looking at a single example from a specific class of shapes. It is difficult to understand what

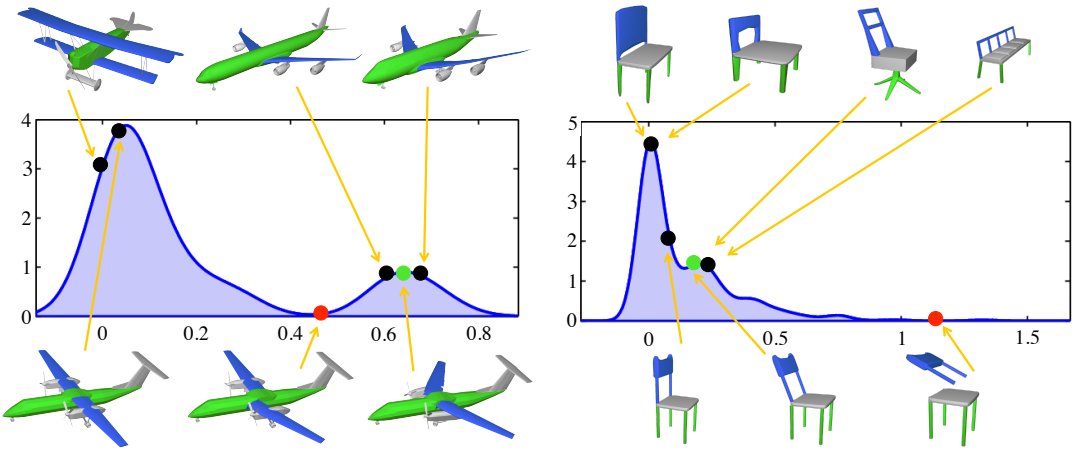


Figure 5.1: Meta-representations of two shape families, where we show one probability distribution from each representation. Here, we see the distribution for the angle between the main axes of airplane wings and fuselage, and the angle between the main axes of chair backs and legs. There are two major modes in each distribution, where examples of shapes corresponding to the black dots are shown. Besides such exploration, the meta-representation can be used for applications like guided editing: the user deforms selected shapes, taking them to lower probability states (red dots), and then the system, guided by the meta-representation, returns them to higher probability states (green dots).

really characterizes the family of the shape without additional information or other examples from the same class. Humans typically rely on their prior knowledge to infer this information. Hence, with the growth of shape repositories, researchers have focused on co-analyzing sets of shapes in order to benefit from the collective information in the group [39, 49, 54, 84, 95]. However, the typical outcome of these methods, a segmentation and/or correspondence, does not really summarize properties that define the shapes as elements of a set.

Our hypothesis is that by modelling shape structure in a probabilistic manner using the relations of shape parts, we can summarize shape families and thus pose problems such as shape editing as preserving the familial validity of shapes. Starting from a set of co-segmented shapes, we create a representation that captures the *essence* of the shape family, towards the goal of quantifying validity of the shapes. We call this representation a *meta-representation*. Specifically, we learn a system of geometric distributions to encode relative arrangements of parts across the family. Note that our representation is complementary to the one proposed by Kalogerakis et al. [48] who learn a Bayesian network to capture co-occurrence statistics of parts for the purpose of shape synthesis. Instead, we focus on the distribution of part arrangements as learned from a collection of shapes. For example, in the case of a family of planes, we discover that there are two main modes for the angle between the wings and fuselage (see Figure 5.1).

The meta-representation can then be used in a wide array of applications where one seeks to preserve the familial identity of a manipulated shape. Essentially, instead of assuming generic priors on allowed deformations (e.g., as-rigid-as-possible deformation [87]) or semantic relations (e.g., parallel or orthogonal part configurations being preferred [32, 107]), we directly use the representation to determine likely arrangements of parts. As our main application, we use the meta-representation to guide the editing of a shape to ensure that it remains a *valid* element of the set by keeping its main geometric characteristics. Thus, a desirable deformation amounts to refining part geometries and positions to increase validity as quantified by the meta-representation. The meta-representation then provides such guidance. In practice, this is realized by an editing tool that constrains parts to a valid space as the shape is edited.

Furthermore, by summarizing the input shape family, the meta-representation can also be used for exploring the set of shapes. In addition, it provides a collective handle to simultaneously refine a collection of shapes. For example, the user can directly edit the system of distributions, while our system adjusts the input shapes according to the prescribed meta-representation. This leads to a novel and intuitive *coupled editing* tool for sets of shapes.

We evaluate the proposed representation on various datasets and demonstrate the advantage of having such a meta-representation both for analyzing and manipulating shape families.

5.2 The Meta-representation

In this section, we describe the meta-representation at a high-level. In the subsequent sections, we give details on its construction and how it can be used for different applications.

Assumptions about the input. We assume that the input shapes are coming from the same family and are pre-segmented and consistently labeled. That is, the label of each shape segment is taken from a pre-defined set of labels that are relevant to the particular family. We follow this assumption both for the training set that defines the meta-representation and those shapes that are handled by the applications (which may not be part of the training set). Several unsupervised algorithms exist to automatically obtain such a labeled segmentation from an input set of shapes [38, 39, 54, 61, 69, 84, 105], as well as semi-supervised algorithms [95]. Note that most of these algorithms automatically segment the shapes and assign generic labels. The user can then assign semantic names to the labels.

Next, each shape part is represented as an oriented bounding box, and we compute a set of *relations* for the boxes. The relations are functions that capture geometric configurations of the boxes and they can be *unary*, capturing the appearance of a single box in relation to the entire shape, and *binary*, capturing the relative positioning and appearance of a pair of boxes. The purpose of the relations is to capture any consistency of geometric configuration between the parts across the input shapes. The box representation and the relations that we compute are described in detail in Section 5.3.

The meta-representation. The goal of the meta-representation is to capture the *essence* of a specific family of shapes in terms of the geometric relationships between their shape parts. The meta-representation can then be used to estimate whether the parts of an unknown shape are in a typical arrangement for the family of shapes in question. Thus, it is natural to encode the meta-representation as a probabilistic model of the relations. In our work, we encode it as a probability density function (PDF) *independently* for each relation, and associate the PDFs to the part labels. Then, given parts with respective labels, we can query the PDFs to infer the probability of the part configuration. Figure 5.1 shows an example.

More formally, given a set of labels $\mathscr{L} := \{l_1, \ldots, l_m\}$, and a superset of relations $\mathscr{R} := \{R_1, \ldots, R_n\}$, divided into sets of unary relations $\mathscr{U} := \{U_i\}$ and binary relations $\mathscr{B} := \{B_j\}$, the meta-representation encodes a PDF for label l_i and unary relation U_k:

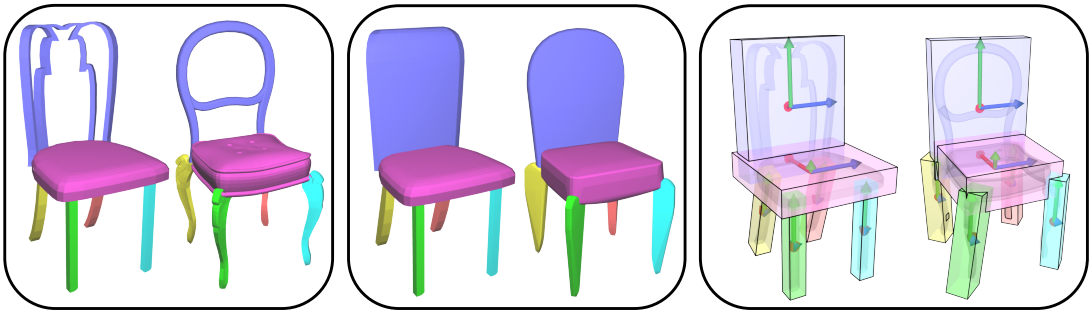
$$\mathrm{PDF}_{l_i,U_k}(r) : \mathbb{R} \to \mathbb{R}, \tag{5.1}$$

and a PDF for every pair of labels $\{l_i, l_j\}$ and binary relation B_k:

$$\mathrm{PDF}_{l_i,l_j,B_k}(r) : \mathbb{R} \to \mathbb{R}. \tag{5.2}$$

The collection of PDFs can be used to estimate the probability of a specific value $r \in \mathbb{R}$ for any relation R_k. They can be learned from the observed relation values extracted from a training set, as explained in Section 5.3, and used for different applications, as discussed in Section 5.4.

Complexity of learning the model. Our main assumption when designing the meta-representation is that the relations and labels are statistically independent, implying that we learn a PDF separately for each relation and individual or pair of labels. Such a simplified model is unable to capture any correlation that may exist between the relations. For example, the angle between the front wing and fuselage of an aircraft may be strongly tied to the angle between the fuselage and stabilizer of the aircraft,



(a) (b) (c)

Figure 5.2: Part abstraction: given the segmented and labeled shapes in (a), we compute the convex hull of each part (b), and then use the hulls to extract an OBB for each part (c), while also consistently ordering the OBB axes across different shapes.

characterizing the style of the aircraft as a fighter jet or a passenger airplane. However, learning such a model involving correlations between all relations would require a great amount of training data (on the order of thousands of shapes), since otherwise outliers can easily bias the learning.

Nevertheless, our simplified model, which encodes only unary and pairwise relations while ignoring their correlation, makes it easy to learn a more robust model from much smaller training sets (only hundreds to dozens of shapes). It is also more space- and time-efficient, allowing us to obtain a quick assessment of the likelihood of a part configuration. However, this comes at a cost: the meta-representation can sometimes lead to conflicts and contradictory constraints. We handle such errors by looking for a solution where we recover correlation by consistency of the pairwise relations (see Section 5.4). In summary, our design choice makes the representation and learning simpler, albeit at the cost of a more involved framework for applications.

5.3 Learning the Meta-representation

The meta-representation is learned from a training set of shapes from the same family $\mathscr{S} := \{S^1, \ldots, S^n\}$. As a pre-processing step, the shapes are all normalized to the unit sphere, aligned using the algorithm presented in Chapter 3 and consistently segmented using a state-of-the-art unsupervised co-segmentation method [84]. First, an abstracted representation is computed for each shape part (see Figure 5.2). Next, a set of relations is computed for every individual part and between every pair of parts (see Figure 5.3). Finally, a statistical model that describes the relations is learned. We now discuss these steps in detail.

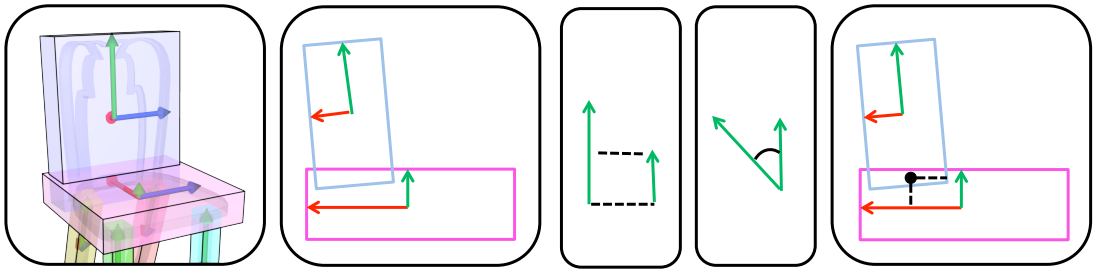


Figure 5.3: (a) Given a pair of parts represented as two OBBs with their axes colored in red, green, and blue, illustrated in 2D in (b), we compute a set of binary relations that describe their relative arrangement. We consider: (c) SCALE, (d) ANGLE, and (e) CONTACT relations.

Part abstraction. Each input shape S^i is pre-segmented into a set of parts $\mathscr{P}^i := \{P^i_1, \ldots, P^i_m\}$. This is done using the unsupervised co-segmentation method of Sidi et al. [84], which performs spectral clustering of potential shape parts in a geometric descriptor space. Any minor discrepancies in the segmentation are manually fixed. Note that some shapes may not have all the parts (e.g., a chair may not have arms). We represent each shape part P^i_j of a shape S^i as an oriented bounding box (OBB). The OBB for part P^i_j is described by its center $\mathbf{c}^i_j$, three normalized axes $(\mathbf{a}^i_{j,1}, \mathbf{a}^i_{j,2}, \mathbf{a}^i_{j,3})$, and the extents $(e^i_{j,1}, e^i_{j,2}, e^i_{j,3})$ of the axes. In order to extract an OBB for a part, we build a set of candidate OBBs and select the one that better captures the symmetries of the part, using the method described below. This method yields more meaningful results than other approaches we experimented with, such as principal component analysis (PCA), or selecting minimum volume boxes.

The construction works as follows [1]: We first compute the convex hull of the part's vertices. Each of the faces of the convex hull defines a plane onto which we project all the vertices of the hull. Next, we compute the 2D bounding box (with minimum area) of the projected vertices [76], and extrude the bounding box by following the direction of the plane's normal until we reach the most distant vertex of the hull. This defines a candidate OBB. Finally, we choose the candidate with the maximum number of reflective symmetry planes.

To test whether any of the three planes defined by the center $\mathbf{c}^i_j$ and the axes $(\mathbf{a}^i_{j,1}, \mathbf{a}^i_{j,2}, \mathbf{a}^i_{j,3})$ possesses a reflective symmetry, we uniformly sample points on the surface of the part and reflect them across the potential symmetry plane. We then measure the distance of the reflected points to the surface. If a sufficient fraction (> 0.9) of the reflected points is closer than a threshold (0.0001), we mark the corresponding

[1] Noa Fish was responsible for developing the OBB extraction method.

plane as a reflective symmetry plane. If there is more than one box which maximizes the number of symmetry planes, we break tie by selecting the one with the minimum volume. If none of the candidate boxes have symmetry planes, the smallest volume box is selected. Note also that, similar to [107], we detect parts with rotational symmetry and mark them as special primitives with only one meaningful (rotation) axis.

Consistent axes ordering [2]**.** To ensure that the axes of part boxes are consistently ordered across different shapes, we assume that all the shapes have the same upright orientation and face the same direction. We sort the OBB axes so that they best align with the global shape axes. Specifically, the first axis is set as the one that best aligns with the global x-axis, and the second axis as the one that best aligns with the y-axis and is orthogonal to the first chosen axis. This procedure ensures a consistent ordering for the majority of shapes. We manually override the automatic fix when the orientation is not consistent. However, due to the regularity of part arrangements in the selected sets, we only needed to fix the orientation for two shapes. An example of the part abstraction and axes ordering is shown in Figure 5.2. The consistent ordering leads to a meaningful representation of relations, as described next.

Inter-part symmetry [3]**.** A shape may contain multiple parts with the same label and in many cases these parts are reflectively symmetric. In order to detect reflective symmetry between two parts with the same label, we employ a variant of the reflective symmetry detection described above. Here, the candidate plane is simply that given by the vector connecting the two centers of the parts along with the half-way point between the centers.

Part relations. Given a shape S^i, we compute a set of unary relations for every part P^i_j and a set of binary relations between every pair of parts (P^i_j, P^i_k). We define a set of relations to describe the geometric configuration of the shape parts. In our work, we choose unary relations that capture mainly the extent of each part axis relative to the scale of the entire shape:

$$\text{EXTENTS}(P^i_j) := \{e^i_{j,t}/d_i\},\ \forall t = 1\ldots 3, \tag{5.3}$$

where d_i is the diagonal of the bounding box of shape S^i.

[2] Noa Fish was responsible for developing the Consistent axes ordering method.
[3] Noa Fish was responsible for developing the Inter-part symmetry detection method.

The binary relations capture relative rotations, translations and scales between parts (illustrated in Figure 5.3):

$$\begin{aligned}
&\text{SCALES}(P_j^i, P_k^i) := \{e_{j,t}^i / e_{k,u}^i\},\ \forall t = 1\ldots 3; u = 1\ldots 3,\\
&\text{ANGLES}(P_j^i, P_k^i) := \{\angle(\mathbf{a}_{j,t}^i, \mathbf{a}_{k,u}^i)\},\ \forall t = 1\ldots 3; u = 1\ldots 3,\\
&\text{CONTACTS}(P_j^i, P_k^i) := \{t_{j,1}^i, t_{j,2}^i, t_{j,3}^i, t_{k,1}^i, t_{k,2}^i, t_{k,3}^i\},
\end{aligned} \tag{5.4}$$

$$\begin{aligned}
\text{where} \quad t_{j,m}^i &= 2\|\mathbf{v}_j\| \cos(\angle(\mathbf{v}_j, \mathbf{a}_{j,m}^i))/e_{j,m}^i\\
\text{with} \quad \mathbf{v}_j &= \mathbf{p}_{\text{int}}(P_j^i, P_k^i) - \mathbf{c}_j^i.
\end{aligned}$$

Essentially, the contact relation between parts (P_j^i, P_k^i) is represented as the relative placement of the intersection point between their two boxes ($\mathbf{p}_{\text{int}}$), in the scaled coordinate system of P_j^i and of P_k^i. The intersection point, if it exists, is found by forming a grid of points on the frame of each box and testing containment of each grid point within the other box. We set $\mathbf{p}_{\text{int}}$ to be the average of the set of points which are found to be contained.

Note that the chosen set of relations is redundant and may over-constrain the configuration between two boxes. The redundancy, however, makes estimation more robust.

Probability density function. As outlined in Section 5.2, we collect the relations for individual boxes and between pairs of boxes for all the shapes in the set, and then build the PDFs for unary (PDF_{l_i,R_k}) and binary (PDF_{l_i,l_j,R_k}) relations. Each PDF is effectively represented by a 1D kernel density estimator (KDE) [85]. Kernel density estimation is a standard non-parametric technique for estimating the PDF of a random variable, and represents the density as a sum of kernels g, each centered at one training sample (one relation value) $x_l \in X$:

$$\text{KDE}(r) := \sum_{l=1}^{|X|} g(r - x_l, h)/|X|, \tag{5.5}$$

where X is the entire set of training samples, and h is the *bandwidth* of the kernel. For our model, we use the common Gaussian kernel:

$$g(t,h) := \exp(-t^2/2h^2)/\sqrt{2\pi h^2} \quad \text{with } t \in (-\infty, \infty).$$

Selecting an appropriate kernel bandwidth is an important problem, illustrated in Figure 5.4. If the chosen bandwidth is too small, as shown in (b), the distribution does not generalize well, and several modes and gaps exist in the density function. If a large

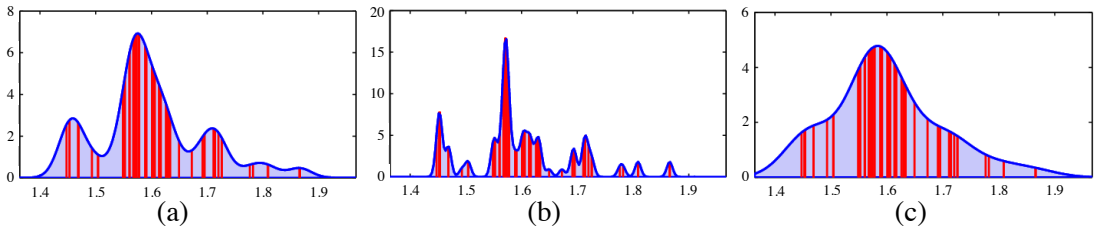


Figure 5.4: Bandwidth selection to create the kernel density estimator (KDE): (a) Automatic selection with our criterion (red bars are training values). (b) Small bandwidth: note how there are many modes and gaps. (c) Large bandwidth: a single mode is created.

bandwidth is selected, as in (c), then important low probability regions are smoothed out in the distribution. Finally, with the correct bandwidth, as in (a), a more meaningful distribution with three large modes is created.

In our work, the bandwidth h is set based on a fixed scale parameter relative to the range of data in the distribution:

$$h := \sigma \cdot (\text{perc}_{95}X - \text{perc}_5X), \tag{5.6}$$

where perc_5 denotes the 5th percentile of X, and the scale $\sigma = 0.05$ was determined experimentally by observing common PDFs for our datasets. Using the percentiles instead of the full data range makes the selection more robust by ignoring outliers. This criterion works well in practice when compared to other well-known alternatives such as cross-validation with the mean integrated squared error or rules of thumb [19]. These criteria are more suitable for finding a cluster structure in the data, and tend to separate the distribution into several modes. On the other hand, our criterion based on a scale parameter allows us to select the appropriate level of detail so that the distributions generalize well.

In general, note that the PDFs capture the commonality of the data in terms of the frequency of values. In regions of the KDE with a large number of training samples, the sum of Gaussians will create peaks with high function values, spread according to the variance of the samples, while regions with only a few samples will have lower function values. Thus, since the KDE represents a continuous density function, the area under the curve corresponds to the probabilities. In practice, to extract a probability $p(r)$ for a specific value r, we integrate the function around a small ε-interval of r:

$$p(r) := \int_{r-\varepsilon}^{r+\varepsilon} \text{KDE}(r)\, dr. \tag{5.7}$$

In our implementation, we set $\varepsilon = 0.01$ of the range of values in the PDF. Note that the probabilities cannot be easily used in an absolute sense, as they will be influenced by the structure of modes in the distribution. However, they can be used in a comparative manner, to compute the probability gain after changes to the relations.

5.4 Using the Meta-representation

In this section, we discuss different usages of the meta-representation. The representation, which summarizes the input shape collections, can be directly used for finding interesting shape configurations in the collection, reshaping any input model guided by the meta-representation, or for collectively editing all the input shapes by directly manipulating the meta-representation.

5.4.1 Exploration of shape families

Given the set of PDFs that define the shape meta-representation, we developed a tool for exploring interesting shape clusters closely tied to the configurations of certain parts. The motivation behind this is that any relation's PDF can exhibit multiple areas where probability density is higher, which in turn means that several shapes posses similar values for that relation. For example, by looking at the angle between the fuselage and wing of an airplane, we might observe a cluster of planes with orthogonal wings, and a cluster of planes with angled wings (see Figure 5.1).

In our exploration interface, the user starts by loading a family of shapes along with their extracted meta-representation. The user then selects any shape as a guidance, and picks one or two shape parts that she wants to use for exploring configurations inside the shape family. Then, any of the unary and binary relations can be selected, and the corresponding PDF can be inspected. Clicking anywhere on the PDF causes the set of loaded shapes to be sorted according to their distance from the clicked value. Then the user can browse through the nearest shapes around the clicked value, in increasing distance. This immediately defines an ordering of the shapes that allows the user to explore the shapes which are most similar in terms of exhibiting that specified value for the relation and the parts in question (see Figure 5.5).

5.4.2 Guided shape editing

Editing a shape can be a difficult task as various geometric and semantic aspects of the shape need to be considered to ensure a *valid* result. We take advantage of the

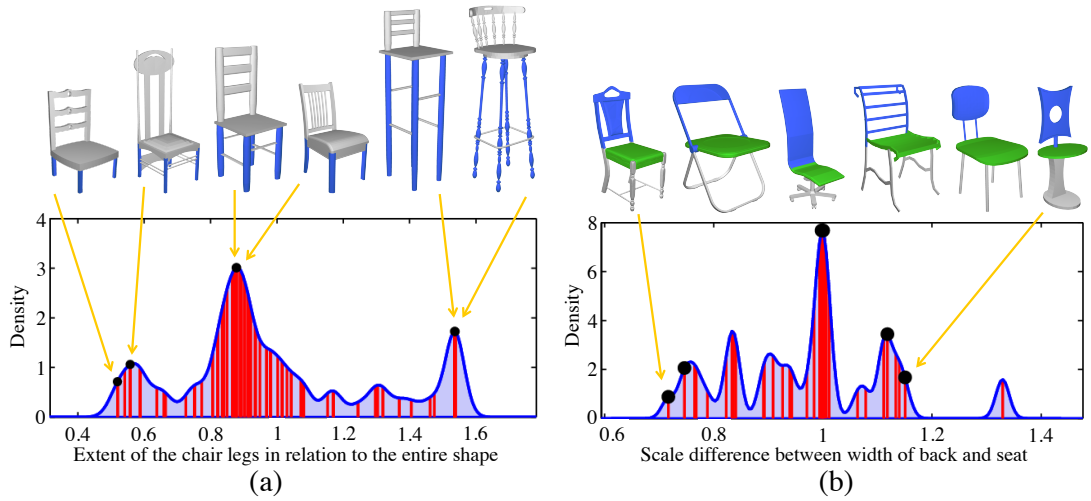


Figure 5.5: The meta-representation enables the exploration of shape repositories: when clicking on different locations of the distributions, the exploration tool presents models with the selected relation values. (a) shows a unary relation for the blue parts, while (b) shows a binary relation between the green and blue parts. The shapes are ordered according to an increase of the selected relations values (black dots). Note that the 3rd and 4th chair both correspond to the highest peak. The red bars are all the training samples used to build the distributions.

meta-representation as a facilitator for this task, and use it as a guidance tool when editing shapes, to create a shape where the part configuration is similar to that observed in the shape family.

We propose an interactive shape editing tool where a user can manipulate the parts of a shape to create a variation of that shape. The user can scale, rotate, and translate one or more parts of the shape. Once an editing action has been carried out, the tool consults the meta-representation to restore the validity of the deformed shape. We can think of the user edit as taking the shape to a certain point in the space of all relations (i.e., points on the different PDF curves), which is associated with a probability given by the meta-representation. Next, our goal is to achieve a part arrangement so that the shape parts move to a (nearby) configuration with higher probability, corresponding to a valid shape. In this process, we constrain the deformation to take the shape to the closest valid state, so that the current part configuration is preserved as much as possible. We first formulate the problem using the meta-representation, and then propose an optimization to enable interactive applications.

Guided editing formulation. We pose the problem of taking an edited shape to a valid state as an optimization where we seek to increase the probability of the part configurations. This can be accomplished by modifying the part configurations so that the probability of their relations is locally maximized. Thus, assuming that the

configuration of the shape parts $\mathscr{P}$ is described by a matrix C with normalized entries, we can pose the deformation goal as

$$\text{Def}(\mathscr{P}) := \arg\max_{C} \text{Obj}(C, \mathscr{P}), \tag{5.8}$$

where the objective function is given by

$$\begin{aligned} \text{Obj}(C, \mathscr{P}) := \quad & \exp\left(-\lambda \|C - C^0\|\right) + \prod_{\forall P_i} p_{P_i}(C_i) \times \\ & \prod_{\forall \{P_i, P_j\}} p_{\{P_i, P_j\}}(C_i, C_j). \end{aligned} \tag{5.9}$$

Here, C^0 denotes the initial configuration of the parts, C_i is the entry of C corresponding to the configuration of part P_i (i.e. it is the row of C that contains the center, three axes and extents of these axes for the OBB describing part P_i), p_{P_i} is the unary probability of part P_i, given by

$$p_{P_i}(C_i) = \prod_k p_{l_i, U_k}(f_{U_k}(C_i)), \tag{5.10}$$

and $p_{\{P_i, P_j\}}$ is the pairwise probability of parts P_i and P_j:

$$p_{\{P_i, P_j\}}(C_i, C_j) = \prod_k p_{l_i, l_j, B_k}(f_{B_k}(C_i, C_j)), \tag{5.11}$$

where l_i and l_j are the labels of P_i and P_j, respectively, p_{li, U_k} is the probability according to Equation (5.7) applied on PDF_{l_i, U_k}, p_{l_i, l_j, B_k} is the probability from $\text{PDF}_{l_i, l_j, B_k}$, f_{U_k} is a function that computes the value of relation U_k according to the configuration C_i of part P_i, and similarly f_{B_k} is a function that computes the value of relation B_k according to the configurations C_i and C_j of parts P_i and P_j. We used a scale factor $\lambda = 0.1$.

Thus, the first term of the objective function ensures that the solution does not deviate far from the initial part configuration, while the second term captures the probability of the entire part configuration as a combination of probabilities for individual parts and pairs of parts, computed according to the meta-representation. We now describe how to find such a solution.

Global optimization. We can directly search for a solution to the objective function in (5.9) with a non-linear optimizer, such as the BFGS quasi-Newton method. Note that the representation for the part configurations in the objective function is independent of the relation set. The relations $\mathscr{R}$ are designed to capture the properties that the configuration should satisfy to ensure shape validity, while the configurations C give the actual part positioning. In our implementation, we encode each C_i in terms of the

position of the OBB center, the scales of its three main axes, and three Euler angles that describe the rotation of the OBB. Thus, the optimizer can directly search for the part configurations that satisfy the objective function.

However, this problem is a non-linear optimization of size $|C|$, which can take a considerable amount of time to solve when the input shapes consist of several parts. For example it would take several minutes for a chair with four legs, a seat and a back, as C would be a 6×15 matrix). Since our goal is an interactive tool, we propose a heuristic solution to speed up the computation of this objective, which we describe next.

Progressive solution. We break the global problem into a series of local optimization steps. In each step, we pick one part $P^\star$ and solve for its position according to a set of parts that have already been fixed. Next, $P^\star$ is added to the set of fixed parts, and we continue with the remaining part(s). We first describe how to determine a good propagation order (i.e., which part to fix next), and then how to position (i.e., fix) the selected part.

Determining a propagation order: Given a set of fixed parts $\mathscr{F} := \{P_1, \ldots, P_m\}$, our goal is to select part $P^\star$ from the set of remaining parts $\mathscr{M} := \{P_{m+1}, \ldots, P_n\}$ that still need to be moved to a fixed position. For each pair of parts (P_k, P_l) with $P_k \in \mathscr{F}, P_l \in \mathscr{M}$, let ${B_i}^c(P_k, P_l)$ be the current value of relation B_i between parts P_k and P_l, and ${B_i}^{opt}(P_k, P_l)$ its value after searching for a higher probability state of P_l according to the fixed part P_k. Let, $p({B_i}^c(P_k, P_l))$ and $p({B_i}^{opt}(P_k, P_l))$ be the corresponding probabilities. We define the probability gain as,

$$\mathrm{PG}[B_i(P_k, P_l)] := \delta \cdot [p({B_i}^{opt}(P_k, P_l)) - p({B_i}^c(P_k, P_l))] \tag{5.12}$$

where,

$$\delta = |{B_i}^{opt}(P_k, P_l) - {B_i}^c(P_k, P_l)| / R_i(k, l),$$

$$R_i(k, l) = \max_c \{{B_i}^c(P_k, P_l)\} - \min_c \{{B_i}^c(P_k, P_l)\}.$$

We select the part with the maximum gain as the part to be fixed next, i.e. we choose, $P^\star \leftarrow \arg\max_{l,i} \mathrm{PG}[B_i(P_k, P_l)]$.

Given a relation value ${B_i}^c(P_k, P_l)$, we follow the ascending gradient direction from this value to a local maximizer in the distribution, according to the PDF corresponding to B_i. This provides a target value ${B_i}^{opt}(P_k, P_l)$. Note that if ${B_i}^c(P_k, P_l)$ happens to be in a very low probability region, we jump to the closest mode of significant probability. We also bypass modes that have a probability that is lower than a threshold (0.01 in our experiments). Figure 5.1 shows examples of such movements.

Solving for a part position: We now position the selected part $P^\star$ given a set of already fixed parts $\mathscr{F}$. One option is to directly use a modified formulation of Equation (5.8) wherein configuration C only considers the parts in $\mathscr{F}$ and $P^\star$; and only relations involving $(P^\star, P_k)$ for all $P_k \in \mathscr{F}$ are considered.

However, in practice, we found an approximate method to be much faster and more suitable for interaction. In this approximate method, each of the fixed parts independently suggests a new part position for $P^\star$; these positions are then combined together for the final position. Specifically, each part $P_k \in \mathscr{F}$ proposes a position $P_k^\star$ based on the relations $\{B_i(P_k, P^\star)\}$. For an even smoother interactive experience, we also experimented with defining the influence of a fixed part based on its adjacency. In this setting, we only consider suggestions for $P^\star$ given by its neighboring parts. We found this approach to be approximately three times faster with little to no difference in the resulting configuration. Since the relations that we consider over-constrain the position of the part, for each part $P_k \in \mathscr{F}$, we sort the relations according to their probability gain and select the subset of relations $\mathscr{B}_k$ that both maximizes the gain and determines a unique position for $P_k^\star$. This subset also determines a confidence weight $w_k \leftarrow \sum_{j \in \mathscr{B}_k} PG[B_j(P_k, P^\star)]$ for this proposed position. The final position for $P^\star$ is then taken as the weighted average of the candidates $P_k^\star$.

Inter-part symmetry handling. Inter-part symmetries within a shape, which are detected in the analysis phase (see Section 5.3), are utilized in our heuristic with the purpose of symmetry preservation, since this is a natural constraint for man-made objects. First, if the user deforms a part that has symmetric counterparts, the system begins by applying a similar deformation to all its symmetric parts, so that they are consistently configured prior to optimization. Next, throughout the optimization, symmetric parts are deformed as a group: only a single part of the group is directly deformed and then the deformation for the remaining parts is automatically inferred from that. Note that same-label parts that are not symmetric also exhibit a measure of consistency in their configurations, but to a lesser extent than symmetric parts. Therefore, such parts are also handled as a group, i.e., their configurations are determined by the same set of fixed parts, although they are optimized separately.

Comparison to global optimization. In comparison with an expensive global approach, we still achieve two goals. First, we maximize the probability of the relations by moving each part to increase the overall probability gain. Although we do not ensure a global maximum, we reach a local maximum of relation probabilities. This is demonstrated by the following experiment: if we apply the global optimization to the result of the progressive solution in Figure 5.7(b), the part configuration does not change

significantly, showing that the probabilities are indeed located at a local maximum. Second, by starting the progressive solution from the user edit and following a local optimizer path, we ensure that we do not deviate much from the initial solution as only deformations that increase the overall probability are allowed.

In terms of complexity, one step of the propagation is much faster to solve (in the order of seconds), as there is only a single part to be optimized at each step. Thus, the aggregated time of all propagation steps is also much less than the time needed to perform the global optimization, making this heuristic well suited to an interactive tool. For example, the first deformation in Figure 5.7(b) takes 0.33 seconds with the progressive solution, while the global optimization takes 1.8 hours.

Comparison to iterative optimization. Note that an iterative solution to the optimization is also possible. In this case there are no fixed parts, and part positions are solved for according to the propagation order in each iteration, until convergence. Such a solution is similar to the ShapeUp framework by Bouaziz et al. [11], which iterates between projecting the shape vertices onto the user constraints, and optimizing their positions in a linear fashion, until convergence. We opted for a non-iterative solution, as it is faster since there are no iterations involved. It is also simpler and more stable as there are no convergence concerns. It is possible for an iterative scheme to get a better solution than a single step method, however, in practise we found the difference to be negligible in most cases. The ShapeUp system also operates at a lower level of abstraction, as user constraints are applied at the level of shape vertices. Our method works at the level of abstract primitives representing each shape part. The deformation implied by our solution, i.e. the translation, scaling and rotation applied in order to map the original part primitive to the optimized part primitive, is applied to the enclosed part geometry to get the final optimized model after the user edit.

5.4.3 Coupled shape editing

Editing multiple shapes at the same time in a coupled manner can be faster and more productive in situations where a modeler would like to perform the same type of edits to a family of shapes. The meta-representation directly allows such coupled guided edits through the relation PDFs. We have developed a coupled shape editing tool to illustrate this concept.

In the coupled editing tool, the user starts by loading a family of shapes, as before, pre-analyzed to extract the meta-representation. By selecting one or two parts from any shape in the family, and specifying a relation, the user can view the corresponding PDF. Recall that the PDF is built from a set of training samples of relation values coming

from all the shapes in the family. The user can then edit the curve using a range of different manipulations. She can click anywhere on the curve to select a point, and then scale the training samples to the left and to the right of the selected point. This can be used to set all training samples for that relation to a specific value that is desired (see Figure 5.10 for an example where the angle between chair back and chair seat is set to a very small range of around 90 degrees).

This manipulation of the curve immediately defines a mapping between the initial training sample values and their newly specified values. We then iterate over all the shapes and set the value for that specified relation to the new value coming from the mapping induced by the new curve. This is done by rotating, translating and scaling the parts in question so that the value for the specified relation between them is set to the new value. This edit can break the relation between the rest of the parts for each shape, therefore we use the original contact relations to rotate and translate these parts until the contact relations are restored. Finally, since the meta-representation is no longer valid after the training shapes have been edited, in the last step we recompute the meta-representation based on the new part configurations for each shape in the family.

5.5 Evaluation

In this section, we describe the experiments performed to evaluate the meta-representation. We designed three tools that illustrate the different areas where the meta-representation may be of use.

For all our experiments, we used four datasets of man-made shapes. The largest dataset contains around 400 chairs, taken from the COSEG benchmark database [95]. We also used a smaller dataset containing around 40 chairs, and two datasets with around 20 bicycles and 20 planes [92]. Smaller datasets were chosen to demonstrate that even with a small number of training samples, the meta-representation can capture important and useful relations between shape parts. All the datasets were preprocessed, starting from segmentation and consistent labeling, to establish correspondence between parts, abstraction of shape parts into boxes, consistent ordering of box axes, building the set of relations, and learning the model PDFs.

5.5.1 Exploration of shape families

Figures 5.1 and 5.5 show examples of exploration enabled by the meta-representation. In Figure 5.1, the user selected the wings and fuselage of airplanes, and navigated through the relations until selecting the angle between the first axis of the two parts.

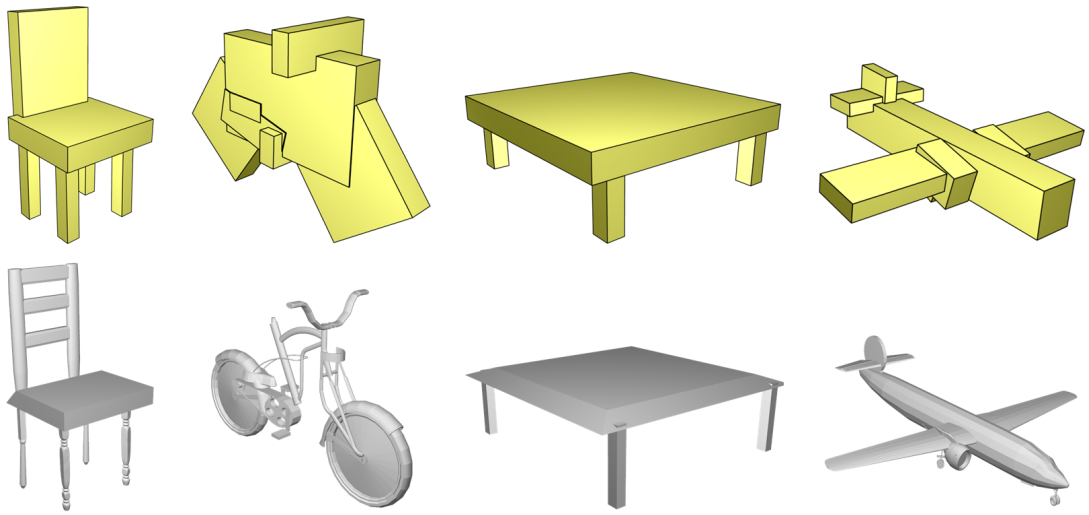


Figure 5.6: Top row: representative abstract configurations automatically obtained from the meta-representations of families of chairs, bikes, tables, and planes. Bottom row: corresponding representative shapes synthesized for the abstract configurations.

Note that these axes are consistently ordered. The distribution shows that there are two main modes in the set: one mode located at around 0 radians (corresponding to wings orthogonal to the fuselage) and another mode centered at 0.7 radians (wings bent by 40 degrees). By clicking on locations of the two modes, the system presents the corresponding shapes. In Figure 5.5 (a), the user selected a unary relation for the chair legs, corresponding to the extent of the vertical axis of the legs in relation to the size of the entire shape. With the tool, the user can identify the three modes that exist and display a few representative shapes. In (b), the scale difference between the second axis of the seat and back is selected (corresponding to the chair width). We observe that from the point of view of this relation, the chairs exhibit larger variation. The user can form an understanding of the groups by inspecting a few shapes for each mode.

These examples show an advantage of the per-relation exploration: instead of pre-selecting a single relation and clustering the shapes into a few representative groups, the user is able to explore the set according to the relation that is relevant for a given task (e.g., finding bar chairs with tall legs or jet airplanes with bent wings). This allows the user to learn about the different shape variations that exist in the set (regarding the specific relation selected), as well as how prevalent they are (as implied by the shape of the distribution). Additionally, nearest neighbors can be retrieved according to a specific relation, providing models in similar geometric configurations.

Representative abstraction: The meta-representation can also be utilized to enhance exploration by automatically creating a representative box configuration and shape for

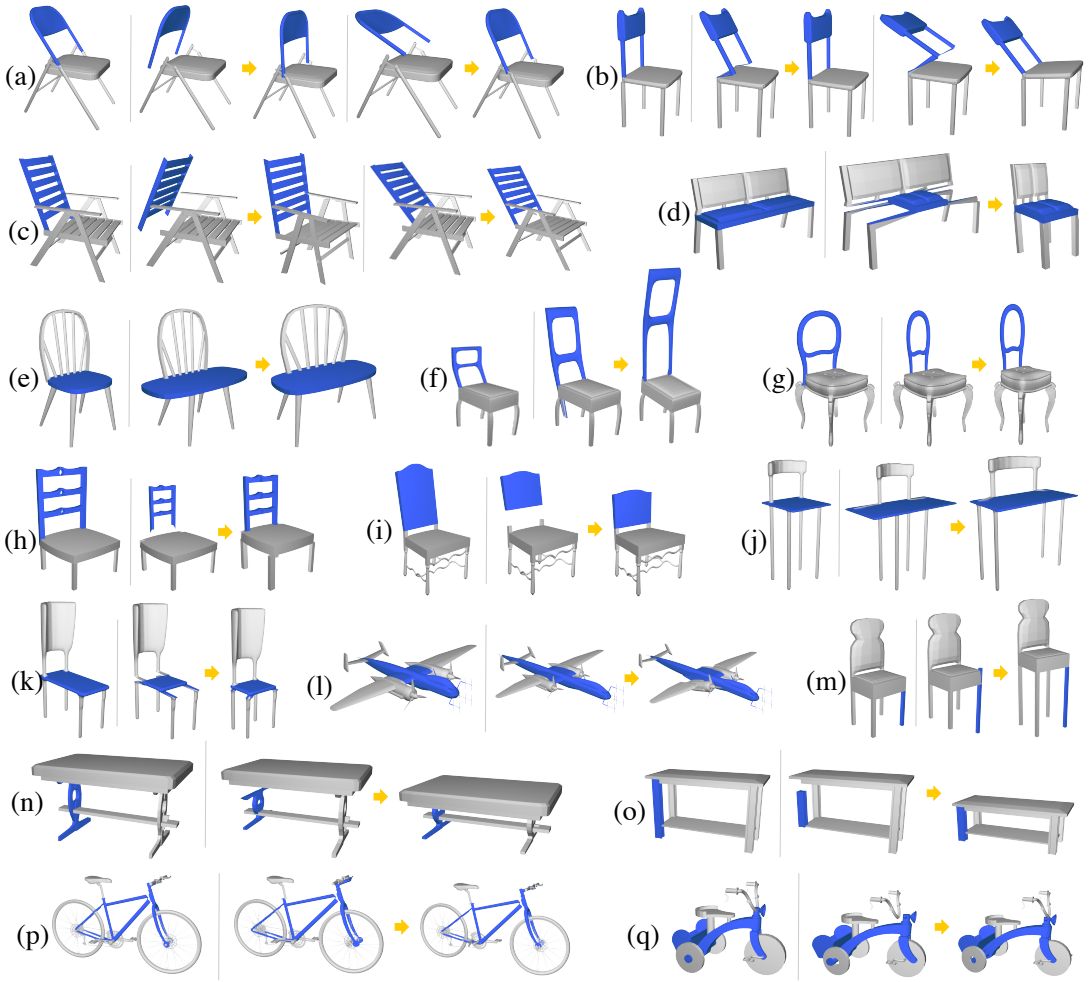


Figure 5.7: Gallery of editing results guided by the meta-representation: each example shows the original shape and one or more edits where the user rotated or scaled one part. The shapes optimized according to the meta-representation are shown after the arrows.

a given shape family. The abstraction is created as follows: given a set of shapes, we first analyze the existence and number of instances of each semantic part. Since each shape differs in this aspect, we only retain semantic parts that exist in σ or more of the shapes (we use $\sigma = 0.5$). The number of instances of a semantic part is based on what is most common across the set (e.g., common number of chair legs). We determine the size properties of each part by querying the PDF of each extent relation for the highest probability mode. In order to infer the relative rotations between the parts, we employ an approach similar to the propagation method described in Section 5.4.2. Here, however, as we aim to create an abstraction that is most indicative of the set, we consider pure probability in place of probability gain. Using the PDFs for all angle relations, we compute the highest probability mode for each relation, and each pair of parts, and use

Figure 5.8: Correcting a chair with a severe deformation using the meta-representation.

it to guide the propagation. Finally, we connect the parts based on an example shape for which the contact relations are maximized. This provides a configuration of boxes that represents a family of shapes. In order to create representative shapes, we find the parts that are closest in size and configuration to each box in the abstraction and combine them to form a shape, while deforming the parts to fit the abstract configuration. See Figure 5.6 for representative abstractions and shapes computed for four sets. Note that the representative abstraction is derived directly from the meta-representation and does not assume any seeding model.

5.5.2 Guided shape editing

Figure 5.7 displays results of guided editing obtained with the progressive solution described in Section 5.4.2, where the user deformed selected parts of the shapes and the guided editing tool then returned shapes optimized according to the meta-representation. Note that the examples in (a), (c), (e), (g), (h), and (i), were optimized with the meta-representation extracted from the small set of chairs, while the other edits were guided by the large set of 400 chairs.

We identify two improvements conveyed by the progressive optimization in these examples. First, if the configuration of parts created by the user is less common for the set, the system deforms the shape to a configuration with higher probability. This can be seen in the first example in (b), where although the user bent the back of the chair, the system rotated it back to a vertical position, as not many chairs have that specific inclined angle in the set. On the other hand, if the back is deformed to a larger angle, the system keeps the shape in this state, as there are more chairs with this configuration (also see Figure 5.1). Secondly, although the edited part becomes disconnected from

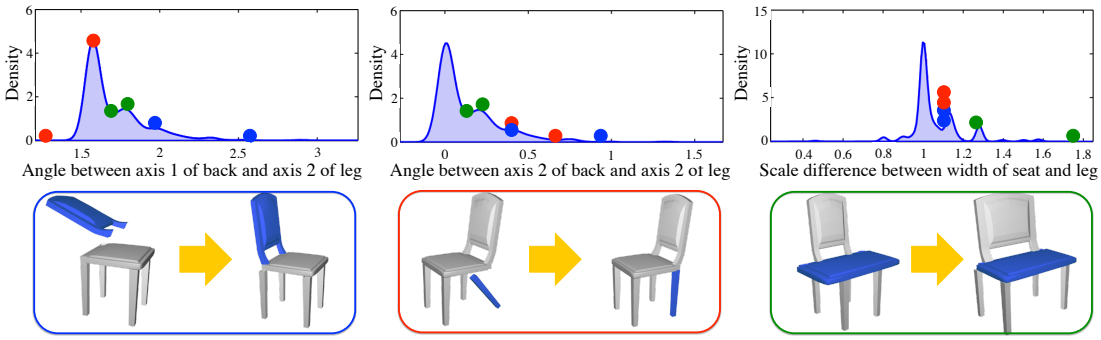


Figure 5.9: Guided editing tool. Bottom: a sequence of three edits. Top: three relations and the values corresponding to the parts involved in the edits are shown before and after optimization (blue is the first edit, red is the second, and green is the third).

the rest of the shape after the user edit, it is again connected to the shape as enforced by the contact point relations.

More importantly, if the editing constraints were derived from a single shape, we could incorrectly assume certain semantics, e.g., that the back and seat or wing and fuselage in Figure 5.1 need to remain orthogonal. In this regard, the meta-representation adds guidance with flexibility, by allowing more deformation freedom where the family supports it, or constraining the shape if it does not. Figure 5.8 further exemplifies this fact: although a chair is severely deformed, having lost its defining characteristics, the guided editing system is able to correct it, showing the effect of the *chair prior* that is present in the meta-representation.

Figure 5.9 shows a sequence of three editing operations applied on the same shape to illustrate the function of the guided editing tool. We can see how after each edit, multiple relation values are taken to lower probability states. These are then restored to higher probability states by the optimization.

5.5.3 Coupled shape editing

An example of coupled editing is shown in Figure 5.10. The user directly manipulates the distribution of angles between the first axes of chair seat and legs, to obtain a simplified curve with less variation in the angles. Next, all the shapes are automatically deformed to conform the set to the new distribution. Thus, all the inclined legs become straight (forming an angle of 90 degrees with the seat), as angles that deviate too much from 90 degrees have a probability of zero in the new distribution.

As discussed in Section 5.4.3, this is a useful application when the goal is to collectively edit a set so that it satisfies specific geometric configurations. The overall

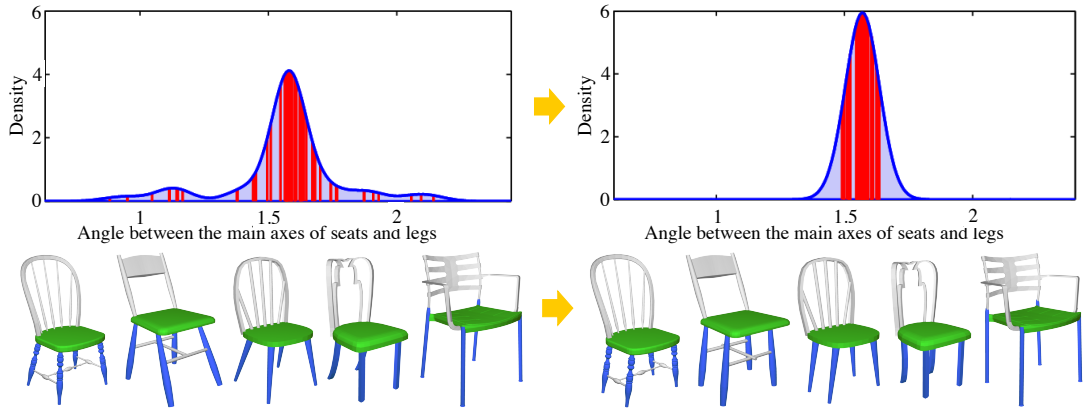


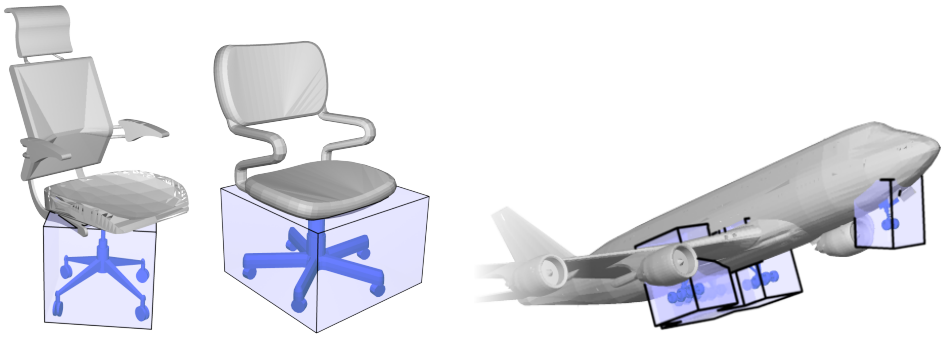
Figure 5.10: Coupled editing of a family of shapes obtained with the meta-representation: the distribution on the left (angle in radians between the main axes of seats and legs) is directly manipulated by a user, who changes the curve to acquire the more compact profile on the right. As a result, all the models in the set are automatically deformed to conform to the new distribution (bottom row).

variation that exists in a relation can be reduced by manipulating the distributions, or outlier models can be forced to conform to a specific range of values by removing their data points from the PDFs. Thus, the distributions act as a higher-level representation of the entire set, and the user is able to impact the geometry of several shapes by manipulating this representation.

5.5.4 Discussion

In this Chapter we introduced a meta-representation to capture the essence of part configurations in a family of shapes. This is accomplished with a system of distributions of unary and binary relations of shape parts, which can then be used for applications such as repository exploration, guided shape editing, and coupled editing of a set of shapes. We addressed the shortcomings of the template-based model presented in Chapter 4, by using a probabilistic model of shape part relations, abstracted through oriented bounding boxes, instead of the simpler axis-aligned bounding box parameterization used in Chapter 4. Nevertheless, there are some limitations and room for future work on this meta-representation.

With the system of distributions learned from a family of shapes, we effectively model shape validity in terms of probability, where the probabilities are derived from the frequency of part configurations in the set. This is appropriate for an exploration tool, as we are interested in exploring common styles in the set. The distributions can also capture some of the shape semantics if the set contains a representative sample of shapes



(a) (b)

Figure 5.11: Inconsistencies in the alignment of different OBBs.

from the family. In this context, there are some directions for introducing additional semantics into the representation. One possibility is to allow the user to add *forbidden regions* to the distributions, implying that part configurations with the corresponding relation values should not exist in the set, potentially allowing asymmetric distributions. In this way, the user can provide additional semantic information, such as disallowing a chair back to bend forward, but still allowing it to bend backwards. Ultimately, the user could also design the distributions by manually drawing the PDFs to imply the semantics of the set.

Regarding our model, we introduced a criterion for selecting a satisfactory bandwidth for each distribution, however, user supervision could also help in determining the optimal bandwidth, yielding a more accurate system of distributions. Moreover, if a shape possesses $|\mathscr{P}|$ parts, the representation consists of $\binom{|\mathscr{P}|}{2} \times |\mathscr{R}|$ curves. Thus, another direction for future work is to design a more compact representation. This has to be balanced with the fact that, as we are able to obtain more data, we would also seek to learn the correlations of part relations, possibly requiring a more complex model.

Furthermore, although the OBB construction described in Section 5.3 is stable across many shapes, it does not always yield optimal boxes, leading to inconsistent results in the presence of ambiguities. For example, the legs of the swivel chairs in Figure 5.11(a) cannot be oriented consistently based only on their symmetries, while the boxes computed for the landing gears in Figure 5.11(b) are also incorrectly aligned due to the geometry of the parts' convex hulls. One possibility for circumventing these problems could be to incorporate an algorithm that learns to predict the consistent box orientation from the orientation of other parts in the same shape. Another alternative is to replace the part primitives with other types of proxies, such as curves or sheets that do not need alignment.

In terms of our implementation choices, different sets of relations can be used with the meta-representation, as we experimented only with one specific set. For shape editing, currently the ordering in which parts are deformed is determined by a probability gain criterion, not by a semantic ordering of parts. For example, the stabilizer of a plane can have more impact on wings than the fuselage. Thus, more sophisticated criteria can be developed to determine such an ordering, and this can have a significant effect on the resulting optimized shapes. Also, the unary relations are currently not taken into account when optimizing the shapes with the progressive solution, as they are not derived from the positions of other parts. Thus, these may also be added to the heuristic solution. Finally, one more direction for future work is to use the meta-representation directly for shape synthesis. This could be accomplished by appropriately introducing variation in the relation values of the shapes, according to an existing set of distributions or user-designed PDFs.

Chapter 6

Conclusions

The continuous increase in the size of shape collections presents opportunities as well as challenges for geometry processing research in the near future. In this thesis we have presented a number of contributions in discovering and modelling shape structure using data from large, unorganized and diverse shape collections. We have also demonstrated how these contributions impact different areas of geometry processing such as shape co-alignment, organization and exploration of shape collections, automatic synthesis of 3D content, and intelligent editing and deformation of shapes.

6.1 Summary

We now summarize the contributions presented in this thesis and their impact to different challenges in geometry processing.

First, in order to tackle the first challenge in deriving structure from a shape collection, we devised a fast algorithm for rigidly co-aligning a set of shapes. Our key observation was that only a small set of rotations are responsible for erroneous shape alignments, in contrast to state-of-the-art methods that uniformly sample the rotation space in search of the best alignments. Based on this observation we used rotational self-symmetries of shapes to pinpoint these rotations, resulting in an algorithm that is both several times faster than such uniform sampling methods, as well as more accurate on average. In order to test our hypothesis, namely that we can reduce the complexity of shape co-alignment without sacrificing accuracy using rotational self-symmetries, we evaluated our algorithm on a large benchmark with ten different shape families. We compared our method to a state-of-the-art method by Huang et al. [41] and we reported 2-16$\times$ speed improvement and accuracy that is comparable on average or even superior for certain shape families.

Second, as a contribution towards modelling shape structure, we presented an analysis approach that extracts a template-based hierarchical parameterization for a collection of shapes. Our parameterization is based on positions and sizes of template boxes that are fit to each shape. We demonstrated how the parameterization can be used to enable intuitive exploration of the collection and provide a high-level overview of the underlying shape space. Perhaps more importantly, the parameterization can be used for interactive synthesis of novel shape variations, just by clicking in the empty space.

In order to test our hypothesis, namely that by modelling shape structure in terms of coarse shape part positions and sizes we can explore a collection and synthesize diverse novel shapes in a fast manner, we evaluated our method through a large user study on four different shape families. We reported the plausibility and diversity of the synthesized results as judged by human participants, compared to results achieved with the state-of-the-art method of Chaudhuri et al. [15]. Our user study indicates that our proposed explorative synthesis interface, can be used by casual users to quickly explore large shape collections, understand the variations inside these collections and create more diverse shapes than the original datasets, while sacrificing some plausibility in the process (plausibility will drastically improve by enforcing contacts at the level of geometry as we explained). We also compared our method to a state-of-the-art shape interpolation method by Jain et al. [46], and demonstrated an example where our method creates more plausible shapes. This is because our method deforms shape parts to fit the interpolated shapes while Jain et al.'s [46] method does apply any deformations. Finally, we evaluated the quality of our shape space embedding via a reachability experiment, where we reported that our parameterization is not affected by excluding random shapes.

Lastly, we introduced a model of structure to capture the relations of part configurations in a family of shapes. We call this model, a meta-representation, as it characterizes a family of shapes in terms of the relations of its shapes' parts. The meta-representation is a system of geometric distributions of unary and binary relations of shape parts, including scale, rotation and translation, which are learned from sets of co-segmented shapes with known correspondence.

In order to test our hypothesis, namely that probabilistically modelling shape structure using shape part relations can be used to summarize shape families, we evaluated our meta-representation on several shape families. In particular, we showed three scenarios where the meta-representation can be exploited to accelerate shape processing; in exploring a shape repository, to locate important shape configurations; in guided editing of a shape, to preserve its familial traits; and in coupled editing of a set of shapes, to collectively deform them via directly editing the geometric distributions.

6.2 Future Work

Discovering and modelling shape structure from large shape collections poses a number of research challenges, some of which we have tackled in this thesis, while others can be the focus of future attempts.

Starting from the goal of deriving structure from shape collections, we observe that there are still open questions. In this thesis, we presented a method for co-aligning shapes that have a common up-vector. Although this is reasonable assumption for most online shape repositories, there are still cases where this assumption can be broken. The next logical step should therefore tackle this, with an algorithm that can efficiently co-align a set of shapes, under no up-vector assumptions, bearing in mind that as datasets keep growing, efficiency will become more essential. Combining our method and that of Fu et al. [29] for upright orientation of man-made shapes can be one solution to this. Extending our method to parameterize rotations in a 2D space, with a more complicated search for autocorrelation minima might be another solution. Incorporating some notion of semantics, or human supervision can possibly help in this search.

Co-segmentation and co-analysis methods in general do a good job for relatively simple shapes with a small number of parts that can be approximated by primitive shapes. However, they face problems with complex shapes composed of many small parts that are not visibly protruding, for example motorcycles with complex engines. We would like to have a method that can learn all possible different parts that exist in a shape family, no matter how small or inconspicuous they are. This can be a model describing interesting variations of shapes in terms of topology and geometry rather than geometry alone. Collecting statistics from such a model can also help us reason about shape functionality, by studying why shapes contain certain parts or why shapes lack some parts.

Regarding models of structure, as demonstrated by our work in Chapters 4 and 5 it is not trivial to evaluate success of a certain model or a specific method for modelling structure. Depending on the application, a user study with human participants is an appropriate way to judge the visual quality of the results, but there is still a need for more quantitative comparisons, in the same spirit as the alignment benchmark presented in Chapter 3 to evaluate alignment efficiency and accuracy. One possible route to explore would be the development of a benchmark for evaluating shape deformation based on ground truth expected results. This would allow us to compare several competing shape editing or shape synthesis methods similar to the ones presented in this thesis. Selecting

a metric appropriate for the evaluation of such methods remains the most difficult part in this process.

Structure can be used for a variety of shape analysis and processing tasks beyond the ones described in this thesis. An intriguing area which has emerged only recently is that of shape style, which is fundamentally connected to functionality. We are interested in studying what separates style from functionality, and what is a way to extract style from a shape. Whether subtracting functionality from a shape will give us its style is largely unknown, and even in this case, the goal to extract style is far from achieved, despite recent effort in this area. Structure can help answer such questions as we observe more and more samples of the same shape family. Machine learning techniques for extracting features, locating what features can differentiate style and classifying shapes into styles can be a first attempt at the problem. In this thesis we have focused on smaller scale man-made shapes, excluding buildings. Style however is more well-defined for buildings, in terms of architectural style rules connected to the story, age and functionality of each building. It would therefore be interesting to extend the scope of our current work to buildings, and study problems such as unsupervised clustering of buildings into architectural styles, extraction of building style features, and reconstruction of buildings from point clouds or photographs based on style priors.

Shape functionality is also a very promising area on its own. The ability to reason about shape functionality has been actively sought over recent years. Access to a dictionary of functional shape parts can allow novel approaches for computer automated design, where the designer would be instructing the system what is the intended function for an object, and the details of what parts to choose and how to put them together would be worked out automatically. Using motion captured human data along with very simple primitive parts can be a first step towards defining the expected functionality and exploring the space of possible designs. Reasoning about functionality of shapes can be of great help in scene understanding and other fundamental computer vision tasks.

Finally, one very exciting avenue of future work would be the joint analysis of image and shape collections. Image collections have a large body of research, while shape collections are only recently being considered as a research topic. It is thus reasonable to expect that the analysis of shape collections can benefit from already established methods, practises and data from huge image collections. For instance, searching for 3D models might possibly benefit from 2D image search, which has achieved high levels of accuracy by now. What is more interesting, is if the opposite is also true, i.e. whether data from shape collections can help provide solutions to challenges faced in image collections. For example, 3D data might enable novel exploration interfaces for images,

based on queries that are impossible with 2D information, e.g. show images of an object from a certain viewpoint. The challenges in allowing such a joint analysis of two different domains, linking 2D images to 3D shapes, are open to further investigation. At the same time, it would be interesting to study how shape materials and texture can assist in solving existing geometry processing problems. The texture of a model for example, if present, can provide valuable cues towards aligning a set of shapes, that are currently not taken into account, as explained in Section 3.4.8.

Our long term goal is to use data from large shape collections to develop novel representations and shape analysis algorithms for understanding, learning and using shape structure. In this effort, we would like to extend our current focus from man-made shapes to bigger-scale structures such as buildings, and more interestingly, to organic shapes that were not built by man. Eventually, the plan is to complete a framework for discovering and modelling structure of any type of object, that can be used to address challenges in all areas of geometry processing.

References

[1] ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans. *ACM Transactions on Graphics (SIGGRAPH) 22*, 3 (2003), 587–594.

[2] AVERKIOU, M., KIM, V. G., AND MITRA, N. J. Autocorrelation Descriptor for Efficient Co-alignment of 3D Model Collections. *Computer Graphics Forum 34*, 8 (2015). In Press.

[3] AVERKIOU, M., KIM, V. G., AND MITRA, N. J. Autocorrelation Descriptor for Efficient Co-alignment of 3D Model Collections - Project Website, 2015. [http://geometry.cs.ucl.ac.uk/projects/2015/coalignment/; accessed 21-July-2015].

[4] AVERKIOU, M., KIM, V. G., ZHENG, Y., AND MITRA, N. J. ShapeSynth: Parameterizing Model Collections for Coupled Shape Exploration and Synthesis. *Computer Graphics Forum (Eurographics) 33*, 2 (2014), 125–134.

[5] BALLARD, D. H. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition 13*, 2 (1981), 111–122.

[6] BESL, P. J., AND MCKAY, N. D. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence 14*, 2 (1992), 239–256.

[7] BOKELOH, M., WAND, M., KOLTUN, V., AND SEIDEL, H.-P. Pattern-aware Shape Deformation Using Sliding Dockers. *ACM Transactions on Graphics (SIGGRAPH Asia) 30* (2011), 123:1–123:10.

[8] BOKELOH, M., WAND, M., AND SEIDEL, H.-P. A Connection between Partial Symmetry and Inverse Procedural Modeling. *ACM Transactions on Graphics (SIGGRAPH) 29* (2010), 104:1–104:10.

[9] BOTSCH, M., KOBBELT, L., PAULY, M., ALLIEZ, P., AND LEVY, B. *Polygon Mesh Processing*. Taylor & Francis, 2010.

[10] BOTSCH, M., AND SORKINE, O. On Linear Variational Surface Deformation Methods. *IEEE Transactions on Visualization and Computer Graphics 14*, 1 (2008), 213–230.

[11] BOUAZIZ, S., DEUSS, M., SCHWARTZBURG, Y., WEISE, T., AND PAULY, M. Shape-Up: Shaping Discrete Geometry with Projections. *Computer Graphics Forum (Symposium on Geometry Processing) 31*, 5 (2012), 1657–1667.

[12] BROWN, B., AND RUSINKIEWICZ, S. Global Non-Rigid Alignment of 3D Scans. *ACM Transactions on Graphics (SIGGRAPH) 26*, 3 (2007).

[13] CHAOUCH, M., AND VERROUST-BLONDET, A. Alignment of 3D models. *Graphical Models 71*, 2 (2009), 63–76.

[14] CHAUDHURI, S., KALOGERAKIS, E., GIGUERE, S., AND FUNKHOUSER, T. AttribIt: Content Creation with Semantic Attributes. In *Proceedings of ACM Symposium on User Interface Software and Technology* (2013), pp. 193–202.

[15] CHAUDHURI, S., KALOGERAKIS, E., GUIBAS, L., AND KOLTUN, V. Probabilistic Reasoning for Assembly-Based 3D Modeling. *ACM Transactions on Graphics (SIGGRAPH) 30*, 4 (2011), 35:1–35:10.

[16] CHAUDHURI, S., AND KOLTUN, V. Data-Driven Suggestions for Creativity Support in 3D Modeling. *ACM Transactions on Graphics (SIGGRAPH Asia) 29*, 6 (2010), 183:1–183:10.

[17] CHEN, D.-Y., TIAN, X.-P., SHEN, Y.-T., AND OUHYOUNG, M. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum (Eurographics) 22*, 3 (2003), 223–232.

[18] CHEN, Y., AND MEDIONI, G. Object Modelling by Registration of Multiple Range Images. *Image and Vision Computing 10*, 3 (1992), 145–155.

[19] CHIU, S.-T. A Comparative Review of Bandwidth Selection for Kernel Density Estimation. *Statistica Sinica 6*, 1 (1996), 129–145.

[20] COMANICIU, D., AND MEER, P. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 5 (2002), 603–619.

[21] COQUILLART, S. Extended Free-form Deformation: A Sculpturing Tool for 3D Geometric Modeling. In *Proceedings of ACM SIGGRAPH* (1990), pp. 187–196.

[22] DE SILVA, V., AND TENENBAUM, J. B. Sparse Multidimensional Scaling Using Landmark Points. Tech. rep., Stanford University, 2004.

[23] EGGERT, D. W., LORUSSO, A., AND FISHER, R. B. Estimating 3-D rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications 9*, 5 (1997), 272–290.

[24] EITZ, M., RICHTER, R., BOUBEKEUR, T., HILDEBRAND, K., AND ALEXA, M. Sketch-based Shape Retrieval. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4 (2012), 31:1–31:10.

[25] ELAD, M., TAL, A., AND AR, S. Content based retrieval of VRML objects: an iterative and interactive approach. In *Proceedings of Eurographics Workshop on Multimedia* (2002), pp. 107–118.

[26] FISH*, N., AVERKIOU*, M., VAN KAICK, O., SORKINE-HORNUNG, O., COHEN-OR, D., AND MITRA, N. J. Meta-representation of Shape Families. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 34:1–34:11. *joint 1st authors.

[27] FISHER, M., AND HANRAHAN, P. Context-based Search for 3D Models. *ACM Transactions on Graphics (SIGGRAPH Asia) 29*, 6 (2010), 182:1–182:10.

[28] FISHER, M., SAVVA, M., AND HANRAHAN, P. Characterizing Structural Relationships in Scenes Using Graph Kernels. *ACM Transactions on Graphics (SIGGRAPH) 30*, 4 (2011), 34:1–34:12.

[29] FU, H., COHEN-OR, D., DROR, G., AND SHEFFER, A. Upright Orientation of Man-made Objects. *ACM Transactions on Graphics (SIGGRAPH) 27*, 3 (2008), 42:1–42:7.

[30] FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. Modeling by Example. *ACM Transactions on Graphics (SIGGRAPH) 23*, 3 (2004), 652–663.

[31] FUNKHOUSER, T., MIN, P., KAZHDAN, M., CHEN, J., HALDERMAN, A., DOBKIN, D., AND JACOBS, D. A Search Engine for 3D Models. *ACM Transactions on Graphics 22*, 1 (2003), 83–105.

[32] GAL, R., SORKINE, O., MITRA, N. J., AND COHEN-OR, D. iWIRES: An Analyze-and-Edit Approach to Shape Manipulation. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (2009), 33:1–33:10.

[33] GELFAND, N., MITRA, N. J., GUIBAS, L. J., AND POTTMANN, H. Robust Global Registration. In *Proceedings of Symposium on Geometry Processing* (2005), pp. 197–206.

[34] GOLOVINSKIY, A., AND FUNKHOUSER, T. Consistent Segmentation of 3D Models. *Computers & Graphics 33*, 3 (2009), 262–269.

[35] HABBECKE, M., AND KOBBELT, L. Linear Analysis of Nonlinear Constraints for Interactive Geometric Modeling. *Computer Graphics Forum (Eurographics) 31*, 2 (2012), 641–650.

[36] HASSNER, T., ZELNIK-MANOR, L., LEIFMAN, G., AND BASRI, R. Minimal-Cut Model Composition. In *Proceedings of IEEE Shape Modelling International* (2005), pp. 72–81.

[37] HORN, B. K. P. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A 4*, 4 (1987), 629–642.

[38] HU, R., FAN, L., AND LIU, L. Co-Segmentation of 3D Shapes via Subspace Clustering. *Computer Graphics Forum (Symposium on Geometry Processing) 31*, 5 (2012), 1703–1713.

[39] HUANG, Q., KOLTUN, V., AND GUIBAS, L. Joint Shape Segmentation with Linear Programming. *ACM Transactions on Graphics (SIGGRAPH Asia) 30*, 6 (2011), 125:1–125:12.

[40] HUANG, Q.-X., ADAMS, B., WICKE, M., AND GUIBAS, L. J. Non-rigid Registration Under Isometric Deformations. In *Proceedings of Symposium on Geometry Processing* (2008), pp. 1449–1457.

[41] HUANG, Q.-X., SU, H., AND GUIBAS, L. Fine-grained Semi-supervised Labeling of Large Shape Collections. *ACM Transactions on Graphics (SIGGRAPH Asia) 32*, 6 (2013), 190:1–190:10.

[42] HUANG, Q.-X., WANG, F., AND GUIBAS, L. Functional Map Networks for Analyzing and Exploring Large Shape Collections. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 36:1–36:11.

[43] HUANG, Q.-X., ZHANG, G.-X., GAO, L., HU, S.-M., BUTSCHER, A., AND GUIBAS, L. An Optimization Approach for Extracting and Encoding Consistent Maps in a Shape Collection. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 167:1–167:11.

[44] HUANG, S.-S., SHAMIR, A., SHEN, C.-H., ZHANG, H., SHEFFER, A., HU, S.-M., AND COHEN-OR, D. Qualitative Organization of Collections of Shapes via Quartet Analysis. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 71:1–71:10.

[45] HUBER, D. *Automatic Three-dimensional Modeling from Reality*. PhD thesis, Robotics Institute, 2004.

[46] JAIN, A., THORMÄHLEN, T., RITSCHEL, T., AND SEIDEL, H.-P. Exploring Shape Variations by 3D-Model Decomposition and Part-based Recombination. *Computer Graphics Forum (Eurographics) 31*, 2 (2012), 631–640.

[47] JOLLIFFE, I. T. *Principal Component Analysis*. Springer-Verlag, 1986.

[48] KALOGERAKIS, E., CHAUDHURI, S., KOLLER, D., AND KOLTUN, V. A Probabilistic Model for Component-Based Shape Synthesis. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4 (2012), 55:1–55:11.

[49] KALOGERAKIS, E., HERTZMANN, A., AND SINGH, K. Learning 3D Mesh Segmentation and Labeling. *ACM Transactions on Graphics (SIGGRAPH) 29*, 4 (2010), 102:1–102:12.

[50] KAZHDAN, M. *Shape Representations and Algorithms for 3D Model Retrieval*. PhD thesis, Princeton, 2004.

[51] KAZHDAN, M. An Approximate and Efficient Method for Optimal Rotation Alignment of 3D Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*, 7 (2007), 1221–1229.

[52] KAZHDAN, M., CHAZELLE, B., DOBKIN, D., FUNKHOUSER, T., AND RUSINKIEWICZ, S. A Reflective Symmetry Descriptor for 3D Models. *Algorithmica 38*, 1 (2003), 201–225.

[53] KIM, V. G., LI, W., MITRA, N., DIVERDI, S., AND FUNKHOUSER, T. Exploring Collections of 3D Models using Fuzzy Correspondences. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4 (2012), 54:1–54:11.

[54] KIM, V. G., LI, W., MITRA, N. J., CHAUDHURI, S., DIVERDI, S., AND FUNKHOUSER, T. Learning Part-based Templates from Large Collections of 3D Shapes. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 70:1–70:12.

[55] KIM, V. G., LIPMAN, Y., AND FUNKHOUSER, T. Blended Intrinsic Maps. *ACM Transactions on Graphics (SIGGRAPH) 30*, 4 (2011), 79:1–79:12.

[56] KIM, Y. M., MITRA, N. J., HUANG, Q., AND GUIBAS, L. Guided Real-Time Scanning of Indoor Objects. *Computer Graphics Forum (Pacific Graphics) 32* (2013), 177–186.

[57] KIM, Y. M., MITRA, N. J., YAN, D.-M., AND GUIBAS, L. Acquiring 3D Indoor Environments with Variability and Repetition. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 138:1–138:11.

[58] KLEIMAN, Y., FISH, N., LANIR, J., AND COHEN-OR, D. Dynamic Maps for Exploring and Browsing Shapes. *Computer Graphics Forum (Symposium on Geometry Processing) 32*, 5 (2013), 187–196.

[59] KOO, B., LI, W., YAO, J., AGRAWALA, M., AND MITRA, N. J. Creating Works-like Prototypes of Mechanical Objects. *ACM Transactions on Graphics (SIGGRAPH Asia) 33*, 6 (2014), 217:1–217:9.

[60] KRAEVOY, V., JULIUS, D., AND SHEFFER, A. Shuffler: Modeling with Interchangeable Parts. In *Proceedings of Pacific Graphics* (2007), pp. 129–138.

[61] LAGA, H., MORTARA, M., AND SPAGNUOLO, M. Geometry and Context for Semantic Correspondences and Functionality Recognition in Man-made 3D Shapes. *ACM Transactions on Graphics 32*, 5 (2013), 150:1–150:16.

[62] LAMDAN, Y., AND WOLFSON, H. J. Geometric Hashing: A General And Efficient Model-based Recognition Scheme. In *Proceedings of International Conference on Computer Vision* (1988), pp. 238–249.

[63] LEE, J., AND FUNKHOUSER, T. Sketch-based Search and Composition of 3D Models. In *Proceedings of Eurographics Workshop on Sketch-based Interfaces and Modeling* (2008), pp. 97–104.

[64] LEORDEANU, M., AND HEBERT, M. Efficient MAP Approximation for Dense Energy Functions. In *Proceedings of International Conference on Machine Learning* (2006), pp. 545–552.

[65] LEORDEANU, M., HEBERT, M., AND SUKTHANKAR, R. An Integer Projected Fixed Point Method for Graph Matching and MAP Inference. In *Proceedings of Conference on Neural Information Processing Systems* (2009), pp. 1114–1122.

[66] LI, G., LIU, L., ZHENG, H., AND MITRA, N. J. Analysis, Reconstruction and Manipulation using Arterial Snakes. *ACM Transactions on Graphics (SIGGRAPH Asia) 29*, 6 (2010), 152:1–152:10.

[67] LIN, J., COHEN-OR, D., ZHANG, H. R., LIANG, C., SHARF, A., DEUSSEN, O., AND CHEN, B. Structure-Preserving Retargeting of Irregular 3D Architecture. *ACM Transactions on Graphics (SIGGRAPH Asia) 30*, 6 (2011), 183:1–183:10.

[68] LIN, J., JIN, X., AND WANG, C. C. L. Sketch Based Mesh Fusion. In *Proceedings of International Conference on Advances in Computer Graphics* (2006), pp. 90–101.

[69] MENG, M., XIA, J., LUO, J., AND HE, Y. Unsupervised co-segmentation for 3D shapes using iterative multi-label optimization. *Computer-Aided Design 45*, 2 (2013), 312–320.

[70] MITRA, N. J., WAND, M., ZHANG, H., COHEN-OR, D., AND BOKELOH, M. Structure-Aware Shape Processing. In *Proceedings of Eurographics State-of-the-art Reports* (2013).

[71] NAN, L., XIE, K., AND SHARF, A. A Search-Classify Approach for Cluttered Indoor Scene Understanding. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 137:1–137:10.

[72] NGUYEN, A., BEN-CHEN, M., WELNICKA, K., YE, Y., AND GUIBAS, L. An Optimization Approach to Improving Collections of Shape Maps. *Computer Graphics Forum (Symposium on Geometry Processing) 30*, 5 (2011), 1481–1491.

[73] OVSJANIKOV, M., LI, W., GUIBAS, L., AND MITRA, N. J. Exploration of Continuous Variability in Collections of 3D Shapes. *ACM Transactions on Graphics (SIGGRAPH) 30*, 4 (2011), 33:1–33:10.

[74] RUSINKIEWICZ, S., AND LEVOY, M. Efficient Variants of the ICP Algorithm. In *Proceedings of International Conference on 3D Digital Imaging and Modeling* (2001), pp. 145–152.

[75] RUSTAMOV, R., OVSJANIKOV, M., AZENCOT, O., BEN-CHEN, M., CHAZAL, F., AND GUIBAS, L. Map-Based Exploration of Intrinsic Shape Differences and Variability. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 72:1–72:12.

[76] SCHNEIDER, P. J., AND EBERLY, D. H. *Geometric Tools for Computer Graphics*. Morgan Kaufmann, 2003.

[77] SCHULZ, A., SHAMIR, A., LEVIN, D. I. W., SITTHI-AMORN, P., AND MATUSIK, W. Design and Fabrication by Example. *ACM Transactions on Graphics (SIGGRAPH) 33*, 4 (2014), 62:1–62:11.

[78] SEDERBERG, T. W., AND PARRY, S. R. Free-form Deformation of Solid Geometric Models. In *Proceedings of ACM SIGGRAPH* (1986), pp. 151–160.

[79] SHAMIR, A. A Survey on Mesh Segmentation Techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556.

[80] SHAO, T., XU, W., YIN, K., WANG, J., ZHOU, K., AND GUO, B. Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching. *Computer Graphics Forum (Pacific Graphics) 30*, 7 (2011), 2011–2020.

[81] SHAPIRA, L., SHALOM, S., SHAMIR, A., COHEN-OR, D., AND ZHANG, H. Contextual Part Analogies in 3D Objects. *International Journal of Computer Vision 89*, 2-3 (2010), 309–326.

[82] SHARF, A., BLUMENKRANTS, M., SHAMIR, A., AND COHEN-OR, D. SnapPaste: An Interactive Technique for Easy Mesh Composition. *The Visual Computer 22*, 9 (2006), 835–844.

[83] SHEN, C.-H., FU, H., CHEN, K., AND HU, S.-M. Structure Recovery by Part Assembly. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 180:1–180:11.

[84] SIDI, O., VAN KAICK, O., KLEIMAN, Y., ZHANG, H., AND COHEN-OR, D. Unsupervised Co-Segmentation of a Set of Shapes via Descriptor-Space Spectral Clustering. *ACM Transactions on Graphics (SIGGRAPH Asia) 30*, 6 (2011), 126:1–126:10.

[85] SILVERMAN, B. W. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[86] SLOAN, P.-P. J., ROSE III, C. F., AND COHEN, M. F. Shape by Example. In *Proceedings of Symposium on Interactive 3D Graphics* (2001), pp. 135–143.

[87] SORKINE, O., AND ALEXA, M. As-Rigid-As-Possible Surface Modeling. In *Proceedings of Symposium on Geometry Processing* (2007), pp. 109–116.

[88] SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. Laplacian Surface Editing. In *Proceedings of Symposium on Geometry Processing* (2004), pp. 175–184.

[89] TALTON, J. O., GIBSON, D., YANG, L., HANRAHAN, P., AND KOLTUN, V. Exploratory Modeling with Collaborative Design Spaces. *ACM Transactions on Graphics (SIGGRAPH Asia) 28*, 5 (2009), 167:1–167:10.

[90] TERZOPOULOS, D., PLATT, J., BARR, A., AND FLEISCHER, K. Elastically Deformable Models. In *Proceedings of ACM SIGGRAPH* (1987), pp. 205–214.

[91] VAN KAICK, O., TAGLIASACCHI, A., SIDI, O., ZHANG, H., COHEN-OR, D., WOLF, L., AND HAMARNEH, G. Prior Knowledge for Shape Correspondence. *Computer Graphics Forum (Eurographics) 30*, 2 (2011), 553–562.

[92] VAN KAICK, O., XU, K., ZHANG, H., WANG, Y., SUN, S., SHAMIR, A., AND COHEN-OR, D. Co-Hierarchical Analysis of Shape Structures. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 69:1–69:10.

[93] VAN KAICK, O., ZHANG, H., HAMARNEH, G., AND COHEN-OR, D. A Survey on Shape Correspondence. *Computer Graphics Forum 30*, 6 (2011), 1681–1707.

[94] VRANIC, D. V., SAUPE, D., AND RICHTER, J. Tools for 3D-object retrieval: Karhunen-Loeve transform and spherical harmonics. In *Proceedings of IEEE Workshop on Multimedia Signal Processing* (2001), pp. 293–298.

[95] WANG, Y., ASAFI, S., VAN KAICK, O., ZHANG, H., COHEN-OR, D., AND CHEN, B. Active Co-Analysis of a Set of Shapes. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 157:1–157:10.

[96] WEBER, J., AND PENN, J. Creation and Rendering of Realistic Trees. In *Proceedings of ACM SIGGRAPH* (1995), pp. 119–128.

[97] XIE, X., XU, K., MITRA, N. J., COHEN-OR, D., SU, Q., GONG, W., AND CHEN, B. Sketch-to-Design: Context-based Part Assembly. *Computer Graphics Forum 32*, 8 (2013), 233–245.

[98] XU, K., CHEN, K., FU, H., SUN, W.-L., AND HU, S.-M. Sketch2Scene: Sketch-based Co-retrieval and Co-placement of 3D Models. *ACM Transactions on Graphics (SIGGRAPH) 32*, 4 (2013), 123:1–123:15.

[99] XU, K., LI, H., ZHANG, H., COHEN-OR, D., XIONG, Y., AND CHENG, Z.-Q. Style-Content Separation by Anisotropic Part Scales. *ACM Transactions on Graphics (SIGGRAPH Asia) 29*, 6 (2010), 184:1–184:10.

[100] XU, K., ZHANG, H., COHEN-OR, D., AND CHEN, B. Fit and Diverse: Set Evolution for Inspiring 3D Shape Galleries. *ACM Transactions on Graphics (SIGGRAPH) 31*, 4 (2012), 57:1–57:10.

[101] XU, W., WANG, J., YIN, K., ZHOU, K., VAN DE PANNE, M., CHEN, F., AND GUO, B. Joint-aware Manipulation of Deformable Models. *ACM Transactions on Graphics (SIGGRAPH) 28*, 3 (2009), 35:1–35:9.

[102] YU, Y., ZHOU, K., XU, D., SHI, X., BAO, H., GUO, B., AND SHUM, H.-Y. Mesh Editing with Poisson-based Gradient Field Manipulation. *ACM Transactions on Graphics (SIGGRAPH) 23*, 3 (2004), 644–651.

[103] YUMER, M. E., AND KARA, L. B. Co-Abstraction of Shape Collections. *ACM Transactions on Graphics (SIGGRAPH Asia) 31*, 6 (2012), 166:1–166:11.

[104] ZHENG, S., PRISACARIU, V. A., AVERKIOU, M., CHENG, M.-M., MITRA, N. J., SHOTTON, J., TORR, P. H., AND ROTHER, C. Object Proposals Estimation in Depth Image Using Compact 3D Shape Manifolds. In *Proceedings of German Conference on Pattern Recognition* (2015). In press.

[105] ZHENG, Y., COHEN-OR, D., AVERKIOU, M., AND MITRA, N. J. Recurring Part Arrangements in Shape Collections. *Computer Graphics Forum (Eurographics) 33*, 2 (2014), 115–124.

[106] ZHENG, Y., COHEN-OR, D., AND MITRA, N. J. Smart Variations: Functional Substructures for Part Compatibility. *Computer Graphics Forum (Eurographics) 32*, 2pt2 (2013), 195–204.

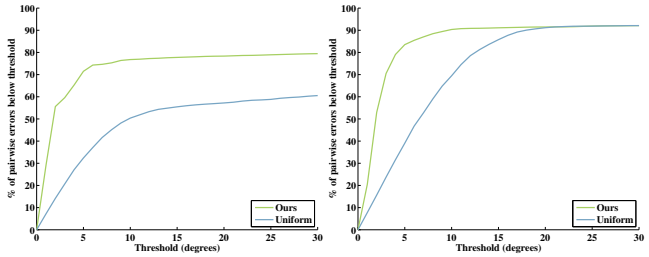
[107] ZHENG, Y., FU, H., COHEN-OR, D., AU, O. K.-C., AND TAI, C.-L. Component-wise Controllers for Structure-Preserving Shape Manipulation. *Computer Graphics Forum (Eurographics) 30*, 2 (2011), 563–572.

Appendix A

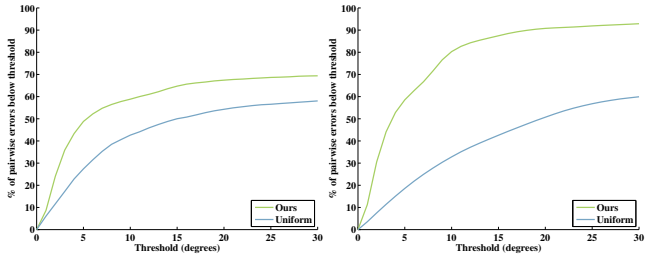
Efficient co-alignment of shape collections results

In this Appendix, we present the full set of alignment results and per-dataset comparisons for our efficient co-alignment method.

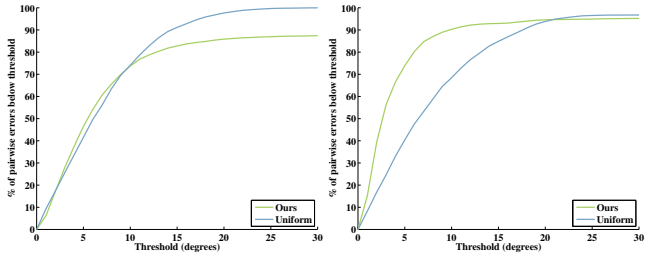
Comparisons of our method and UNIFORM method for all datasets can be found in Figure A.1. Comparisons of our method with clustering and our method without clustering for all datasets can be found in Figure A.2. Comparison of our unsupervised pipeline and our supervised pipeline averaged over all datasets can be found in Figure A.3. Comparisons of our unsupervised pipeline and our supervised pipeline for all datasets can be found in Figure A.4. Renderings of all the shapes in our datasets, before co-alignment (odd rows - in gray) and after co-alignment can be found in Figures A.5-A.15. Renderings of all the shapes in the noisy cars dataset that we used to test our method's performance on outlier shapes can be found in Figure A.7.



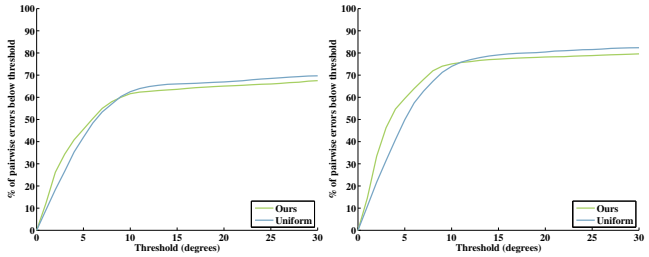
(a) Bikes **(b)** Cars



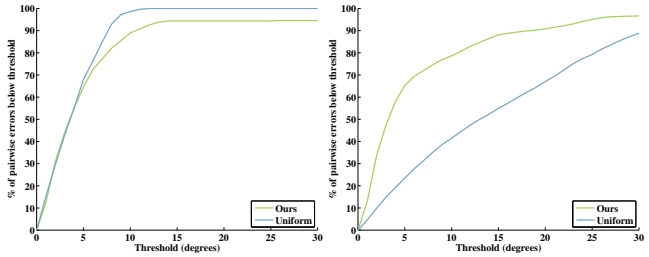
(c) Chairs **(d)** Chairs (big)



(e) Cups **(f)** Helicopters

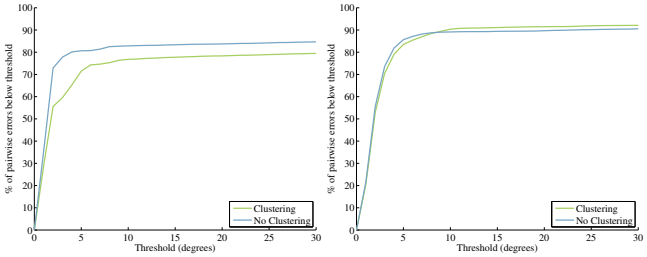


(g) Planes **(h)** Ships

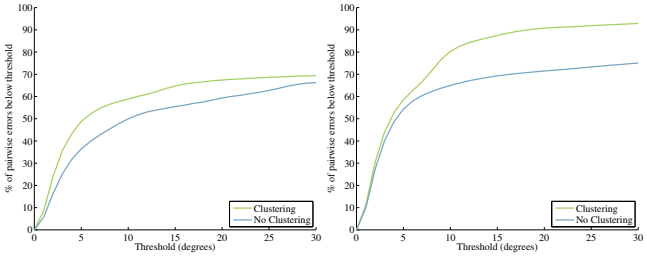


(i) Snowmobiles **(j)** Sofas

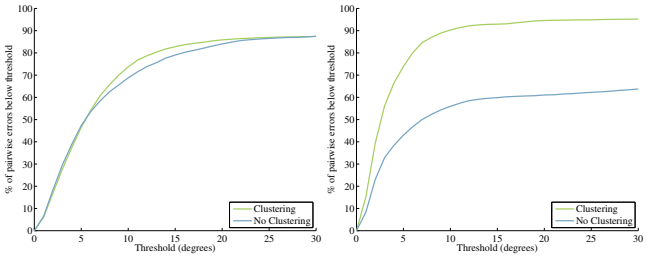
Figure A.1: Comparison of our method (green) and UNIFORM method (blue), for all datasets



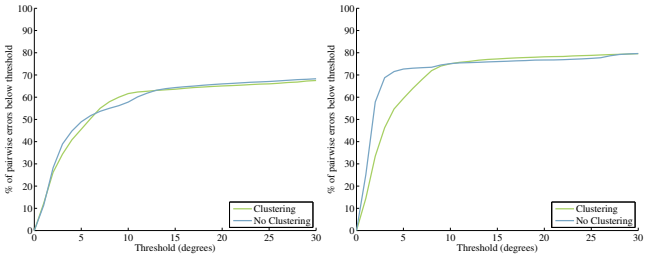
(a) Bikes **(b)** Cars



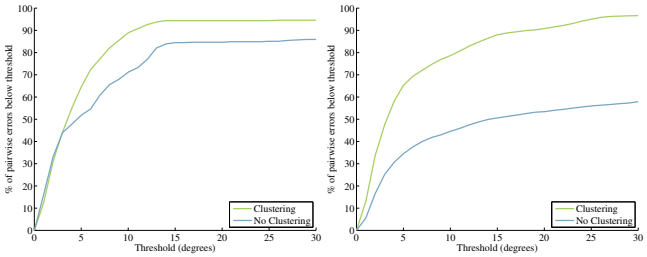
(c) Chairs **(d)** Chairs (big)



(e) Cups **(f)** Helicopters



(g) Planes **(h)** Ships



(i) Snowmobiles **(j)** Sofas

Figure A.2: Comparison of our method with clustering (green) and our method without clustering (blue), for all datasets

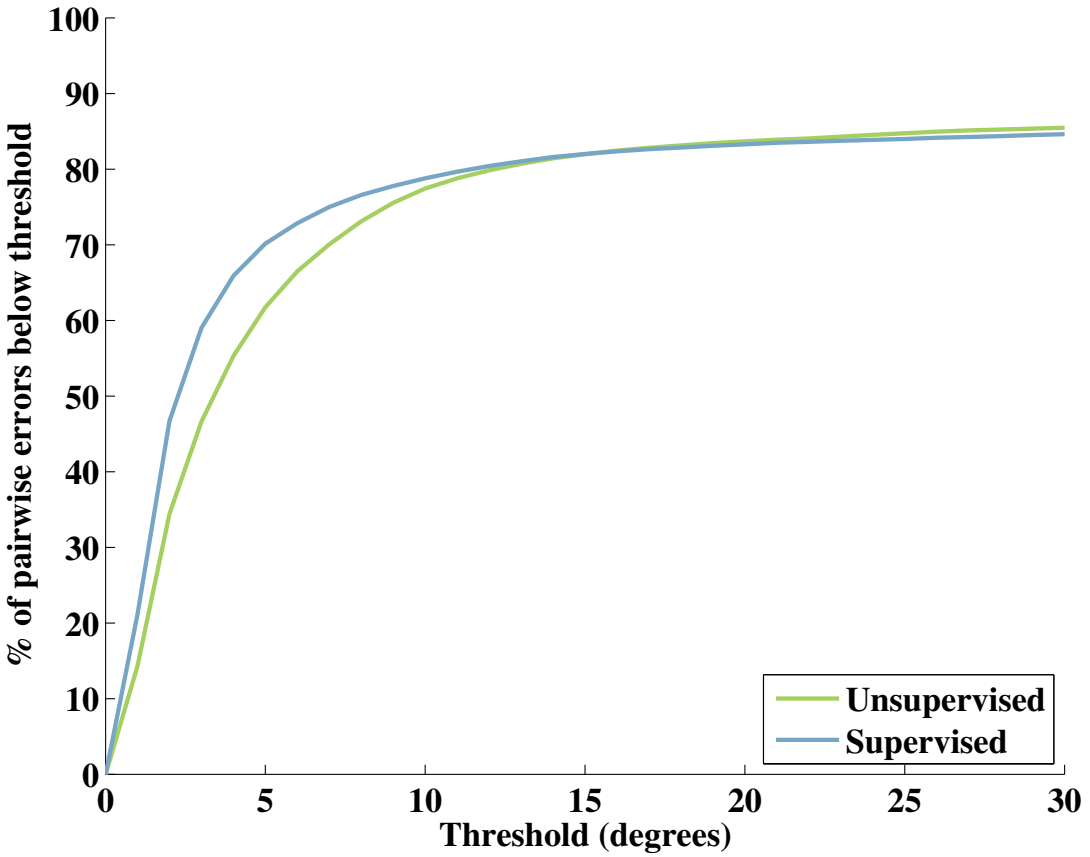
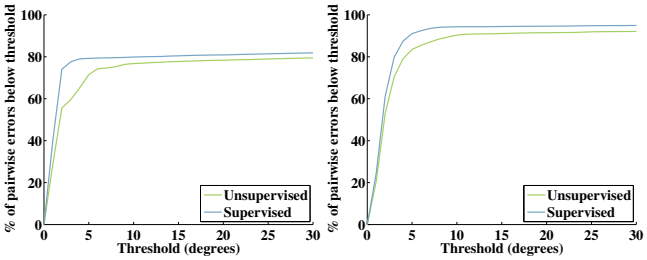
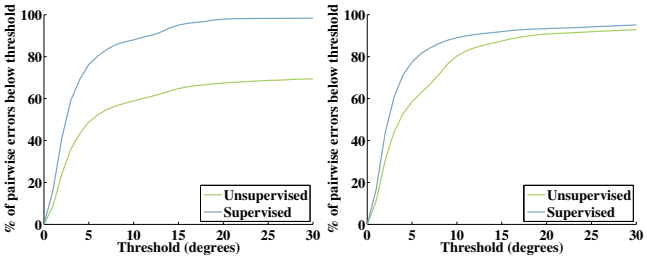


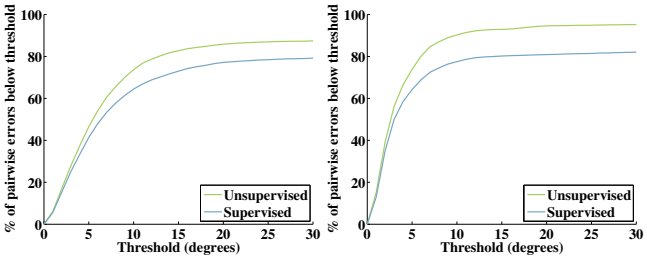
Figure A.3: This figure illustrates the accuracy of our full unsupervised alignment pipeline (green) in comparison to the accuracy achieved if a human aligns shapes between clusters manually (blue). We plot the fraction of models aligned within a prescribed angle threshold, averaged over our 10 datasets.



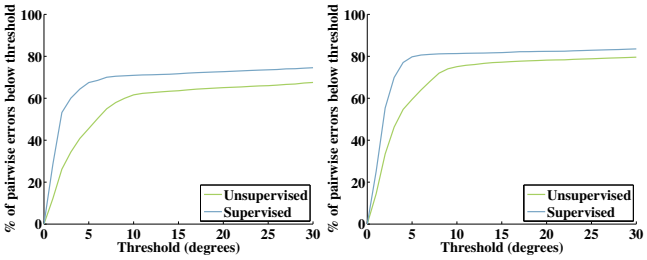
(a) Bikes **(b)** Cars



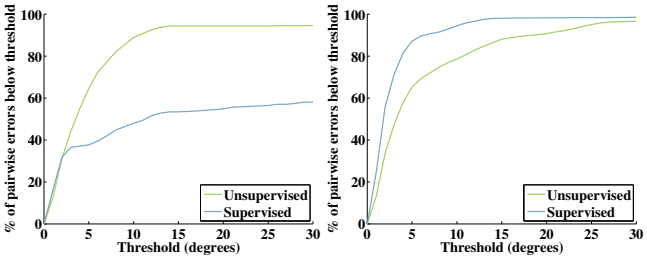
(c) Chairs **(d)** Chairs (big)



(e) Cups **(f)** Helicopters

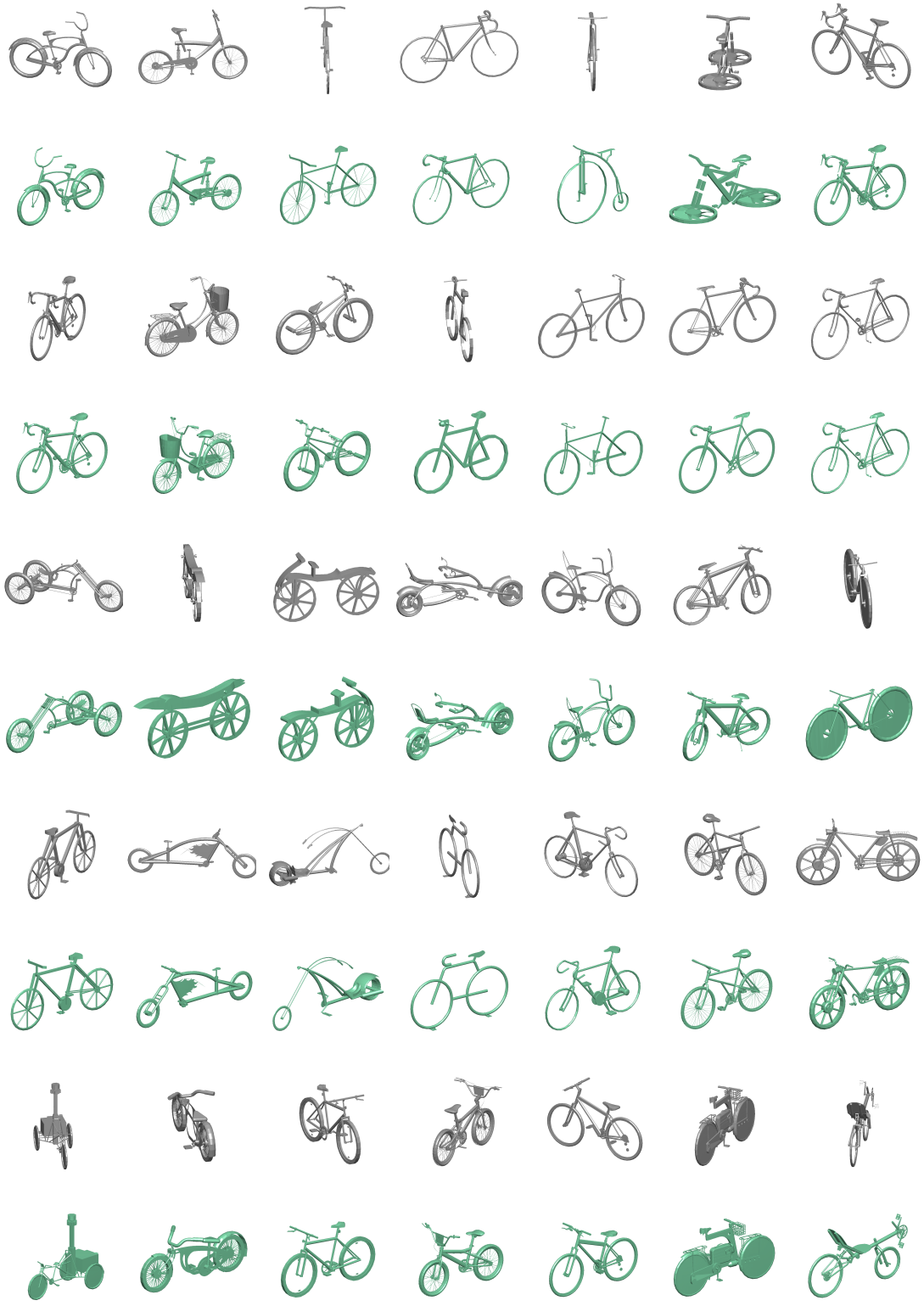


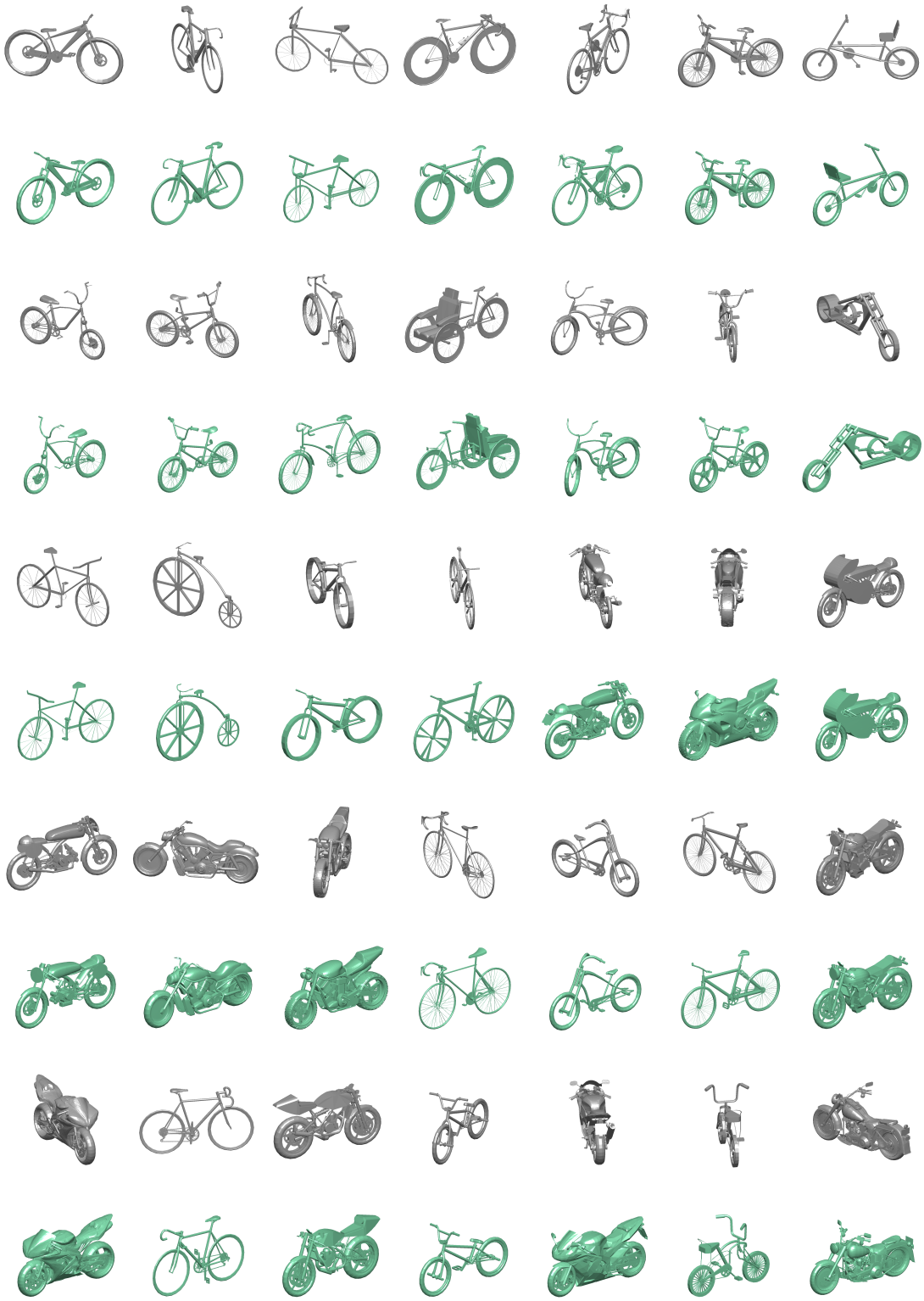
(g) Planes **(h)** Ships



(i) Snowmobiles **(j)** Sofas

Figure A.4: Comparison of our unsupervised pipeline (green) and our supervised pipeline (blue), for all datasets





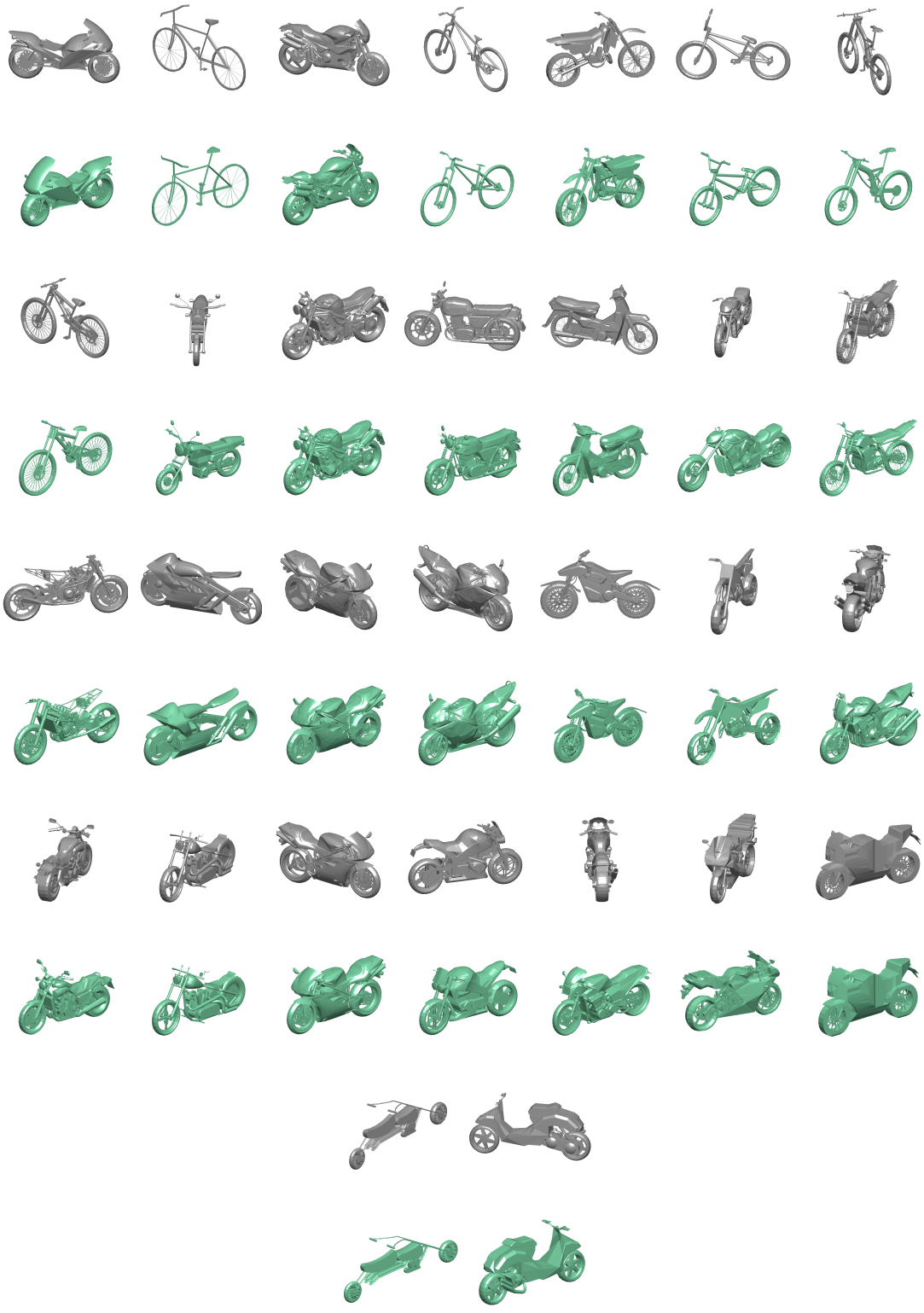
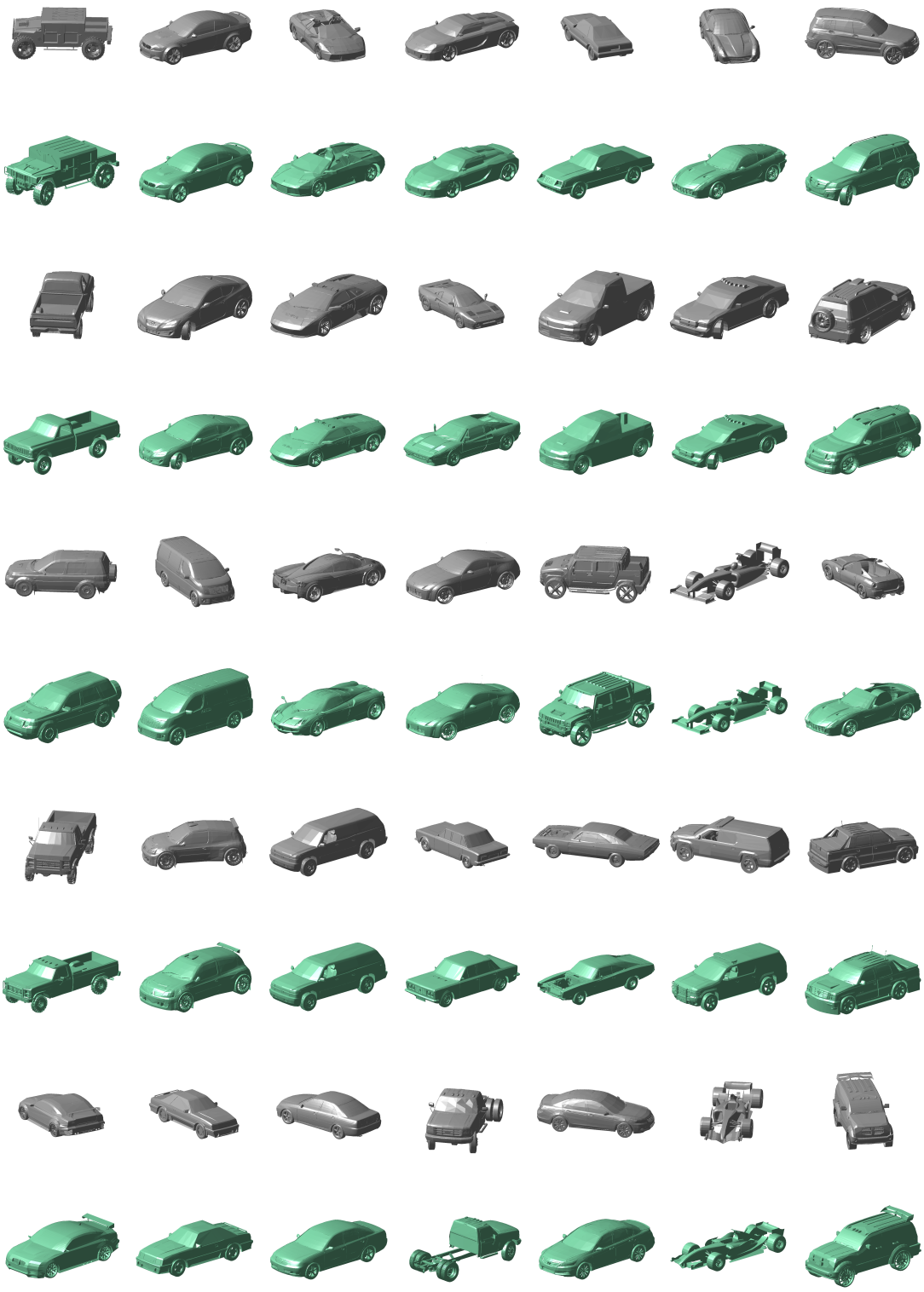
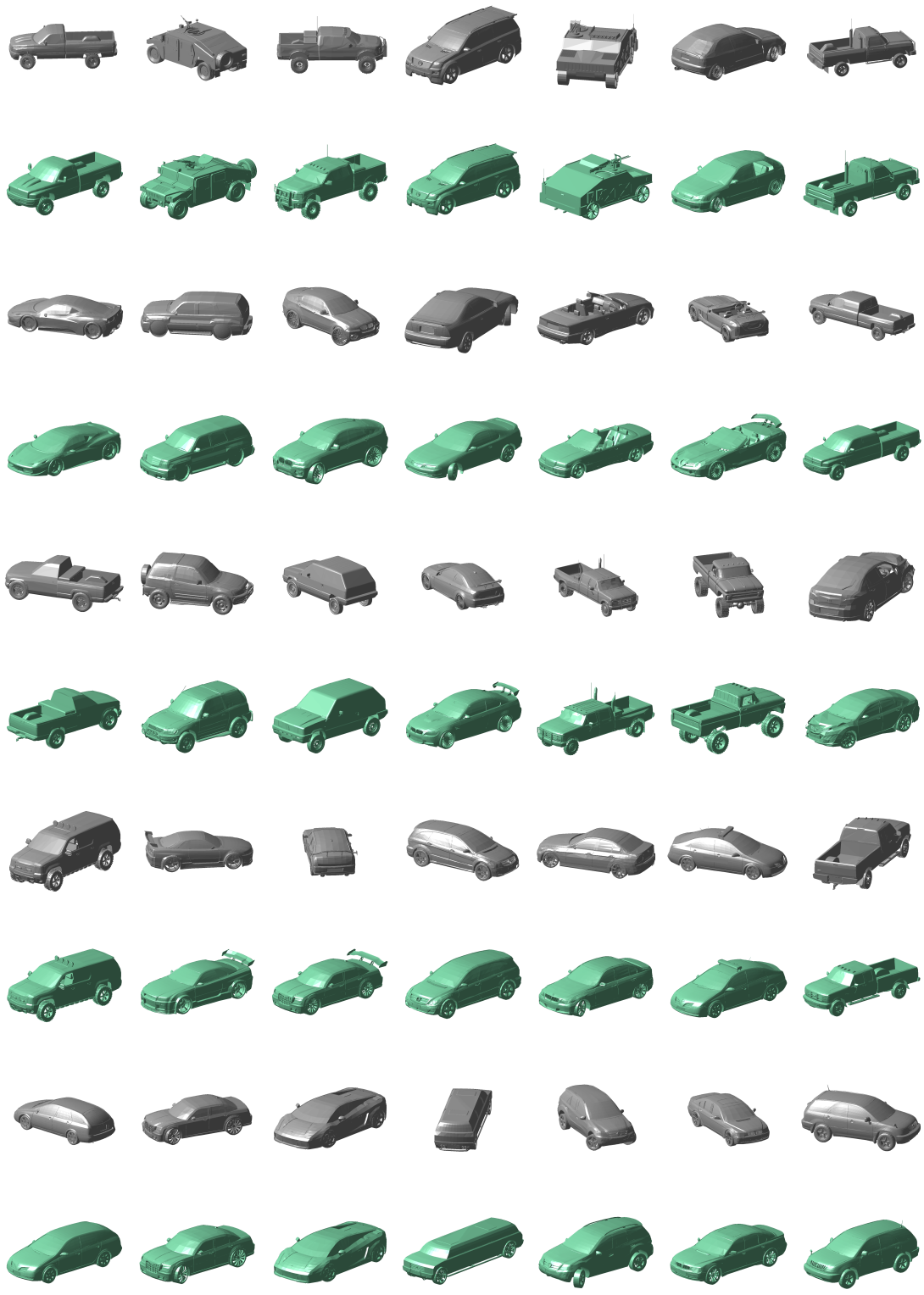


Figure A.5: Bikes dataset before (odd rows - in gray) and after (even rows - in green) alignment.





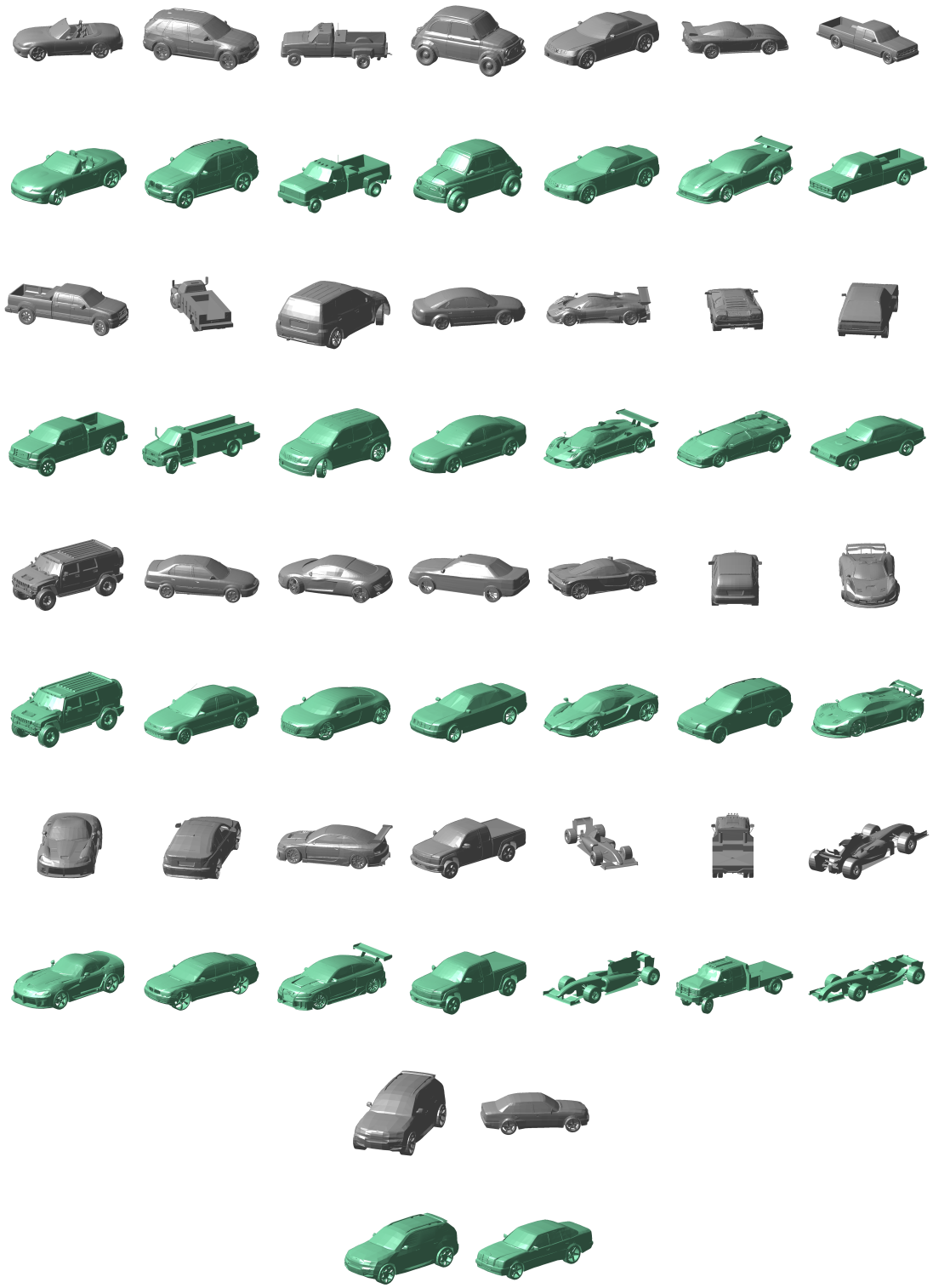
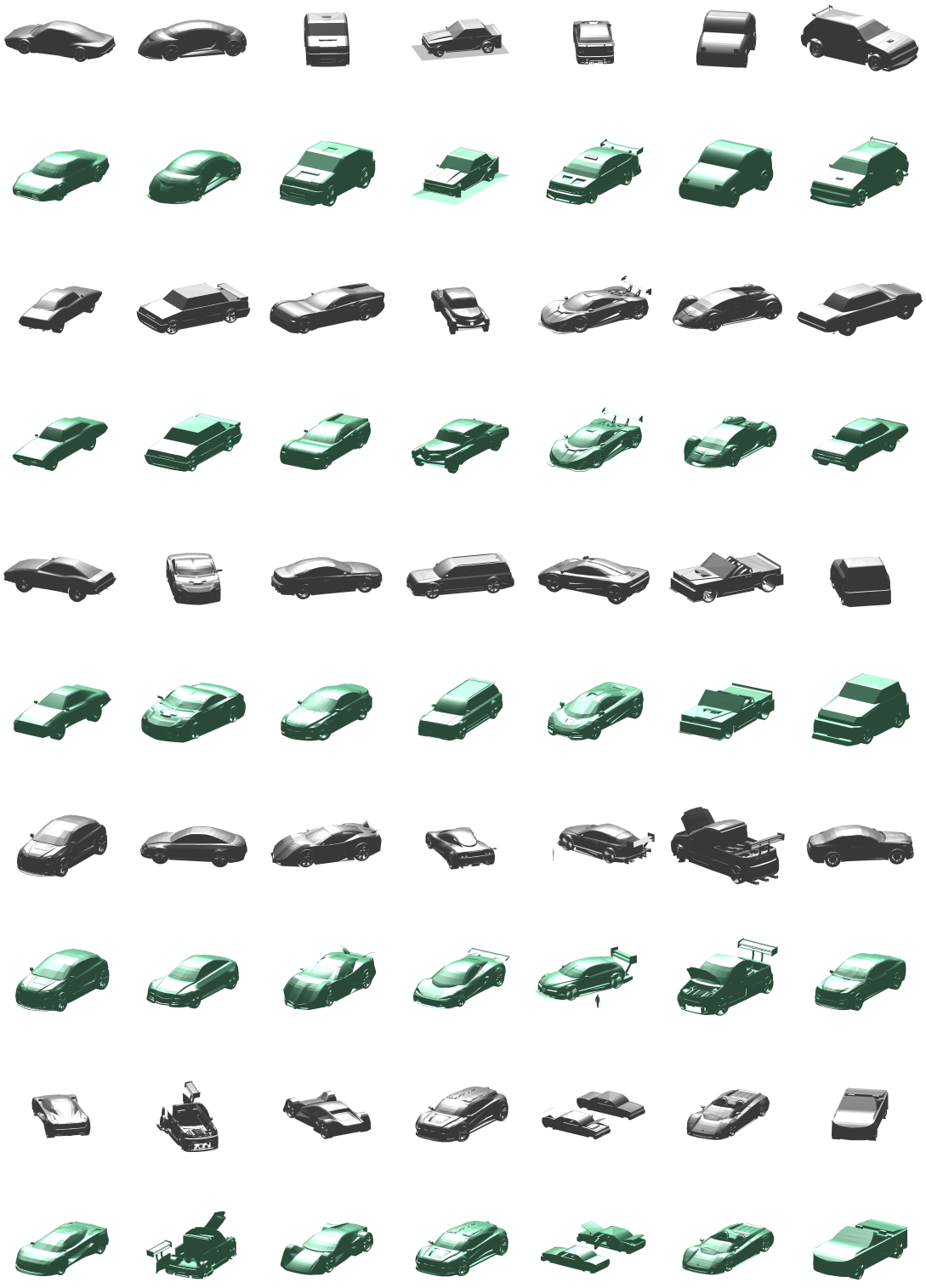
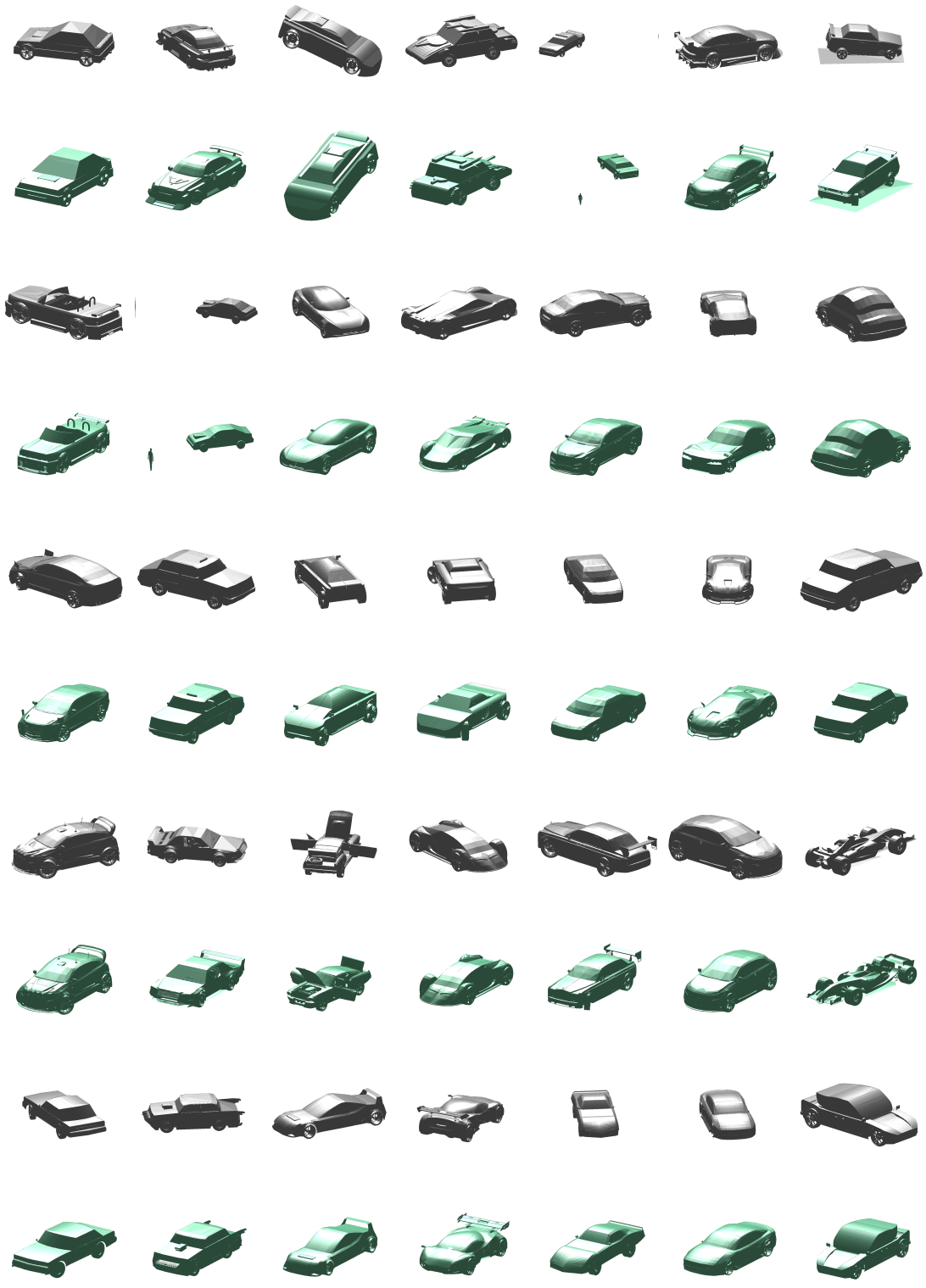
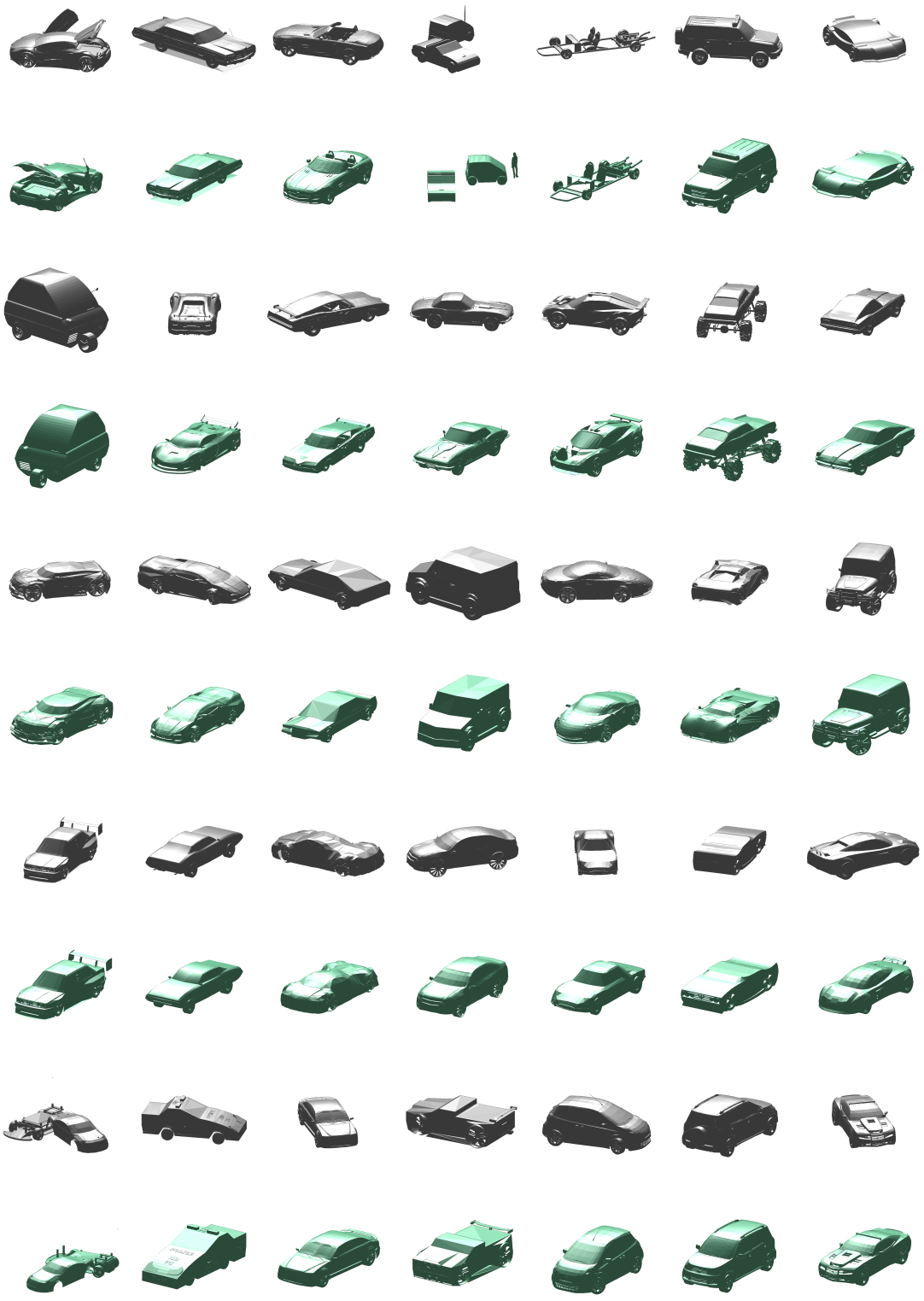


Figure A.6: Cars dataset before (odd rows - in gray) and after (even rows - in green) alignment.







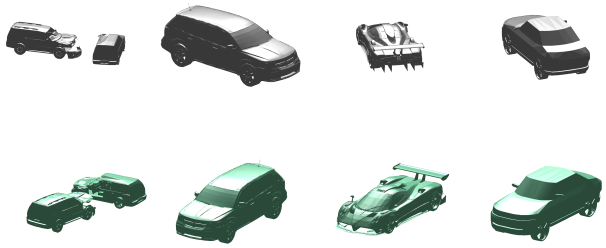
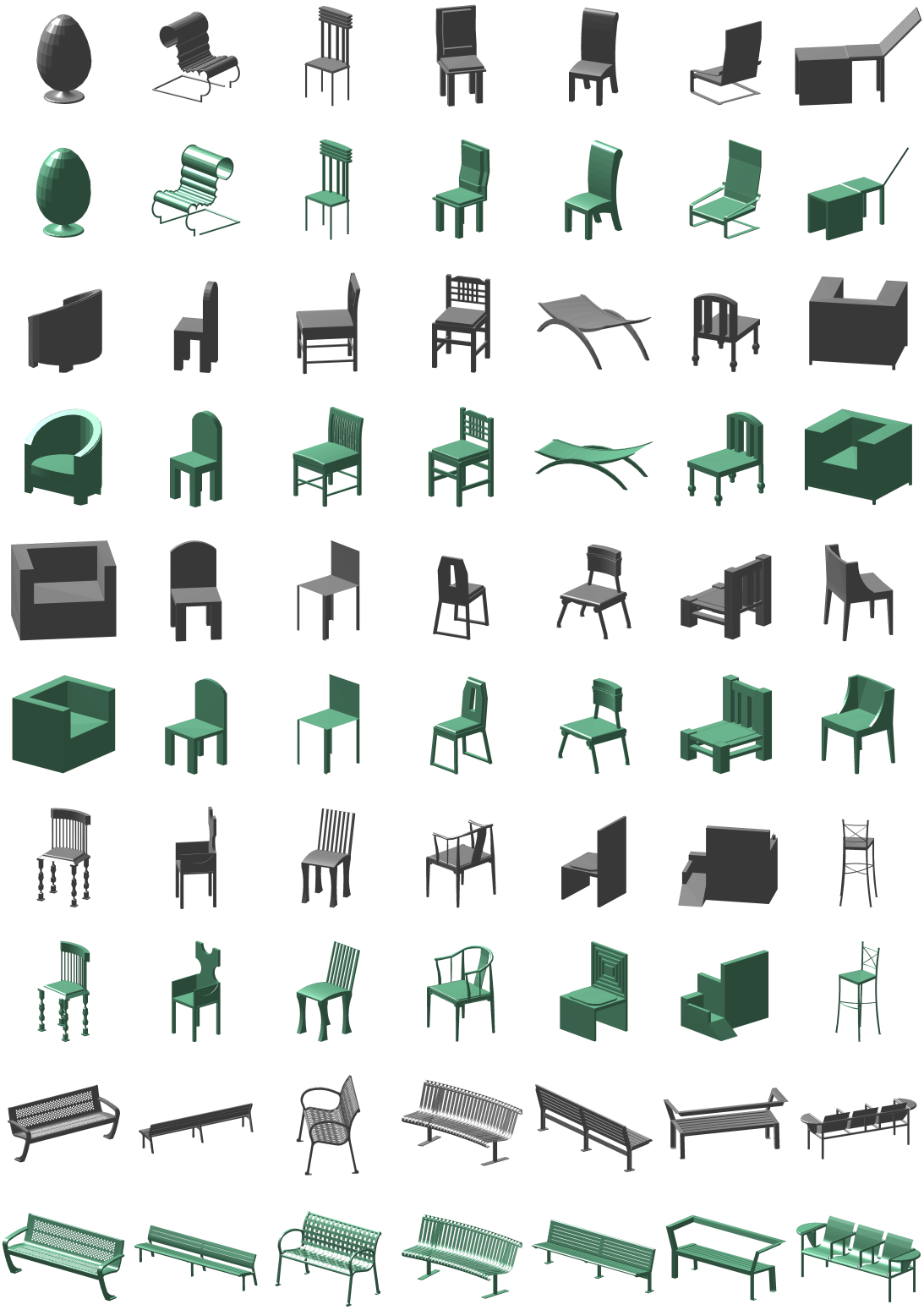
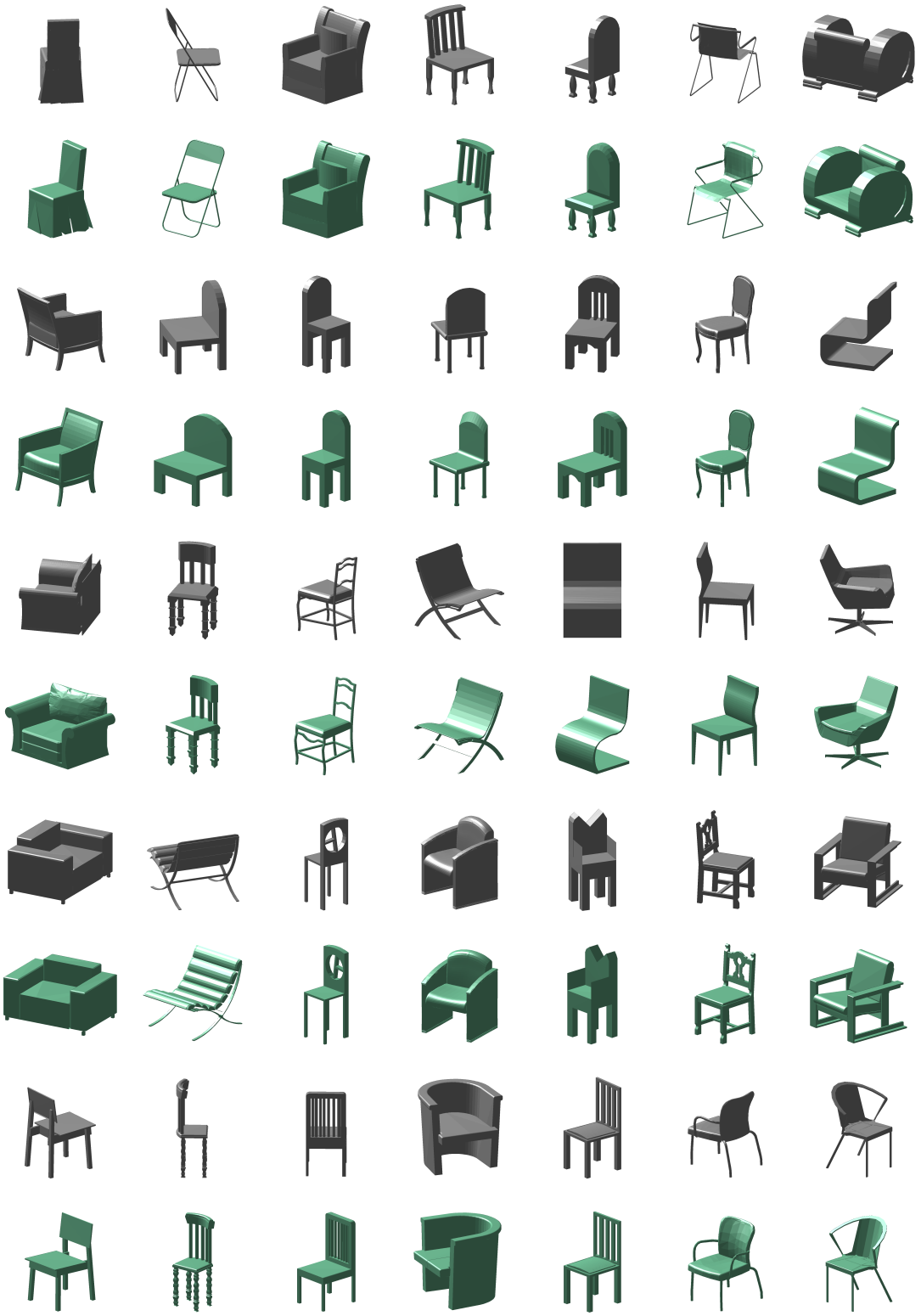


Figure A.7: Noisy cars dataset before (odd rows - in gray) and after (even rows - in green) alignment.

Figure A.8: Chairs dataset before (odd rows - in gray) and after (even rows - in green) alignment.





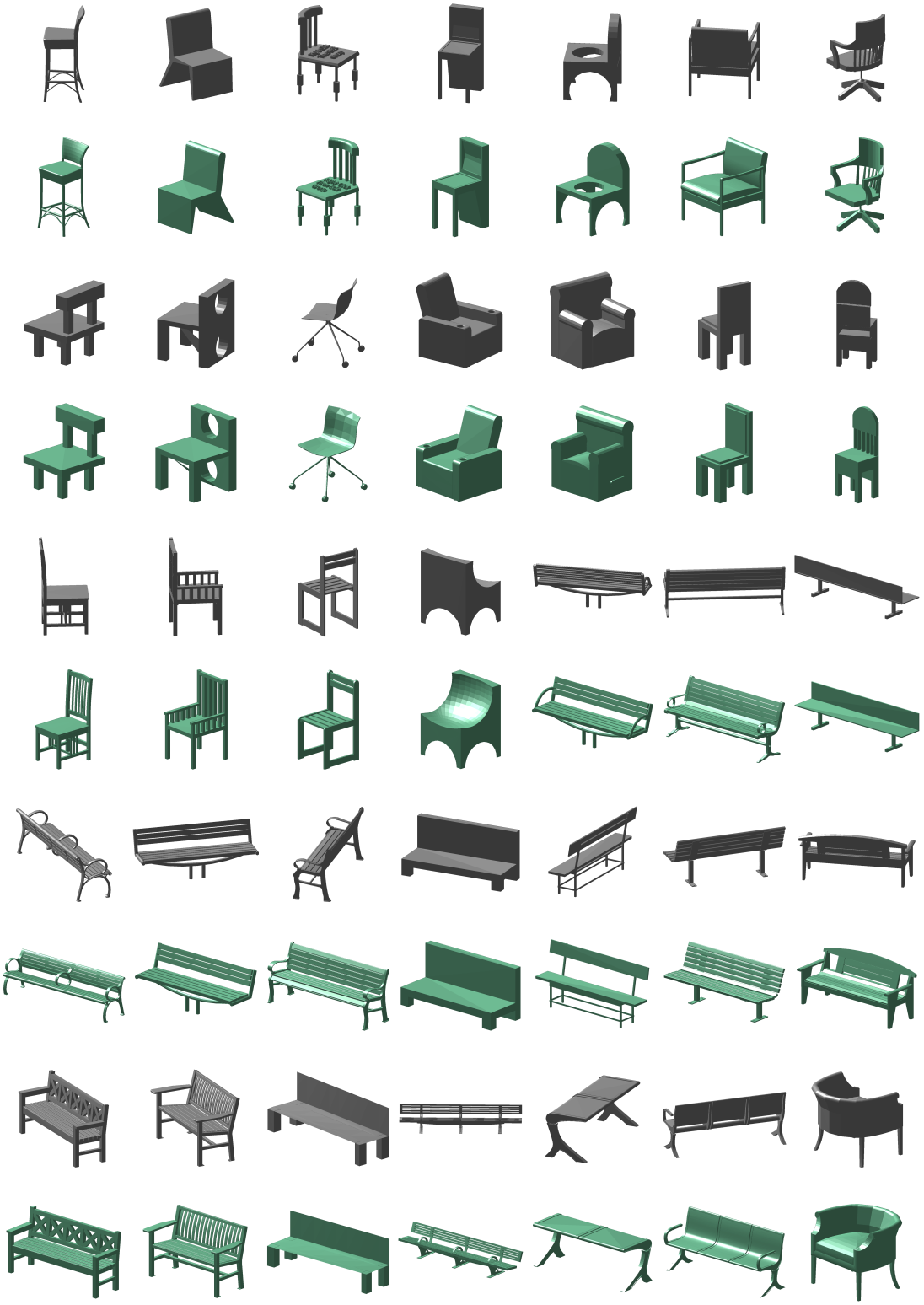


Figure A.9: Chairs (big) dataset before (odd rows - in gray) and after (even rows - in green) alignment.

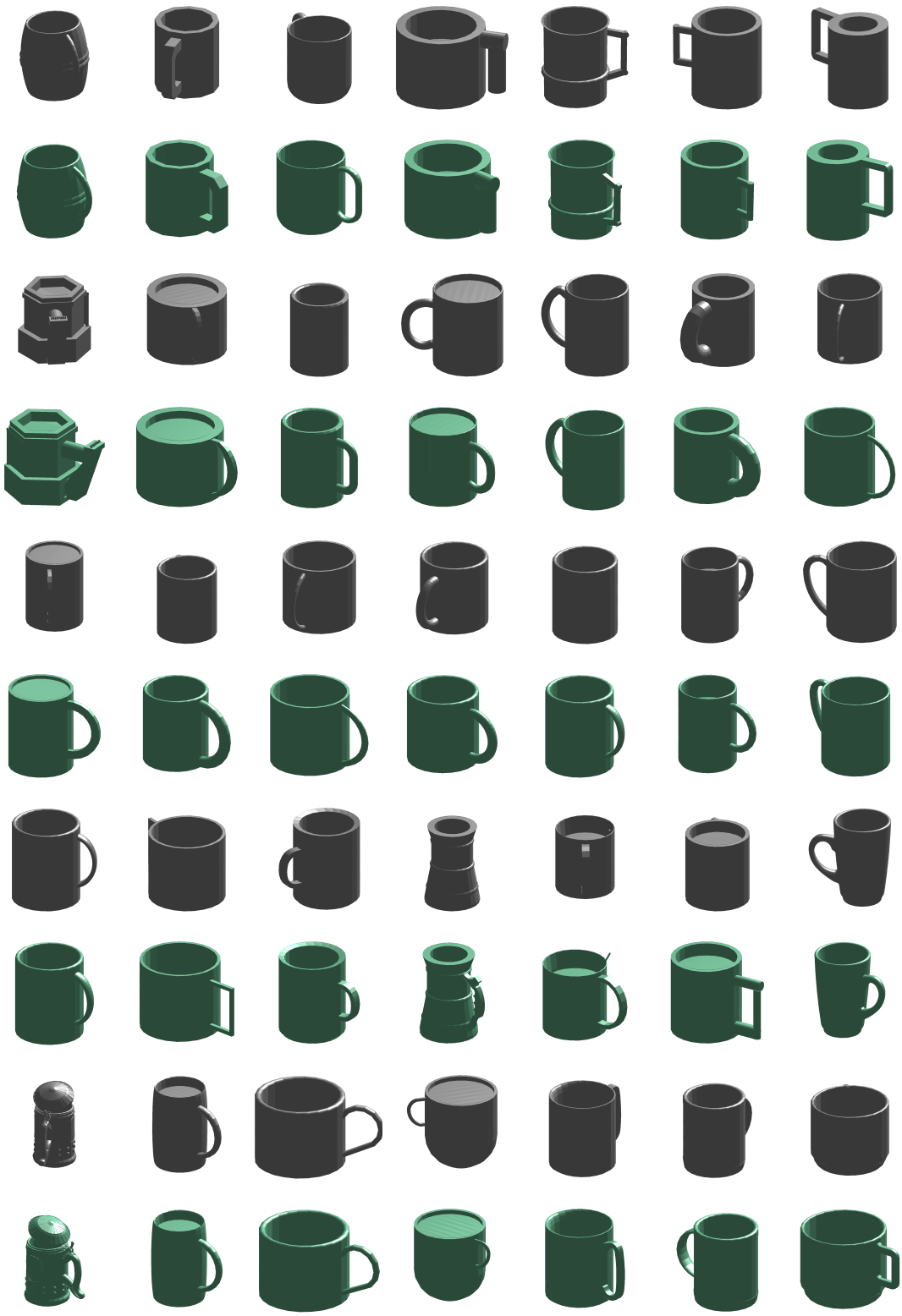
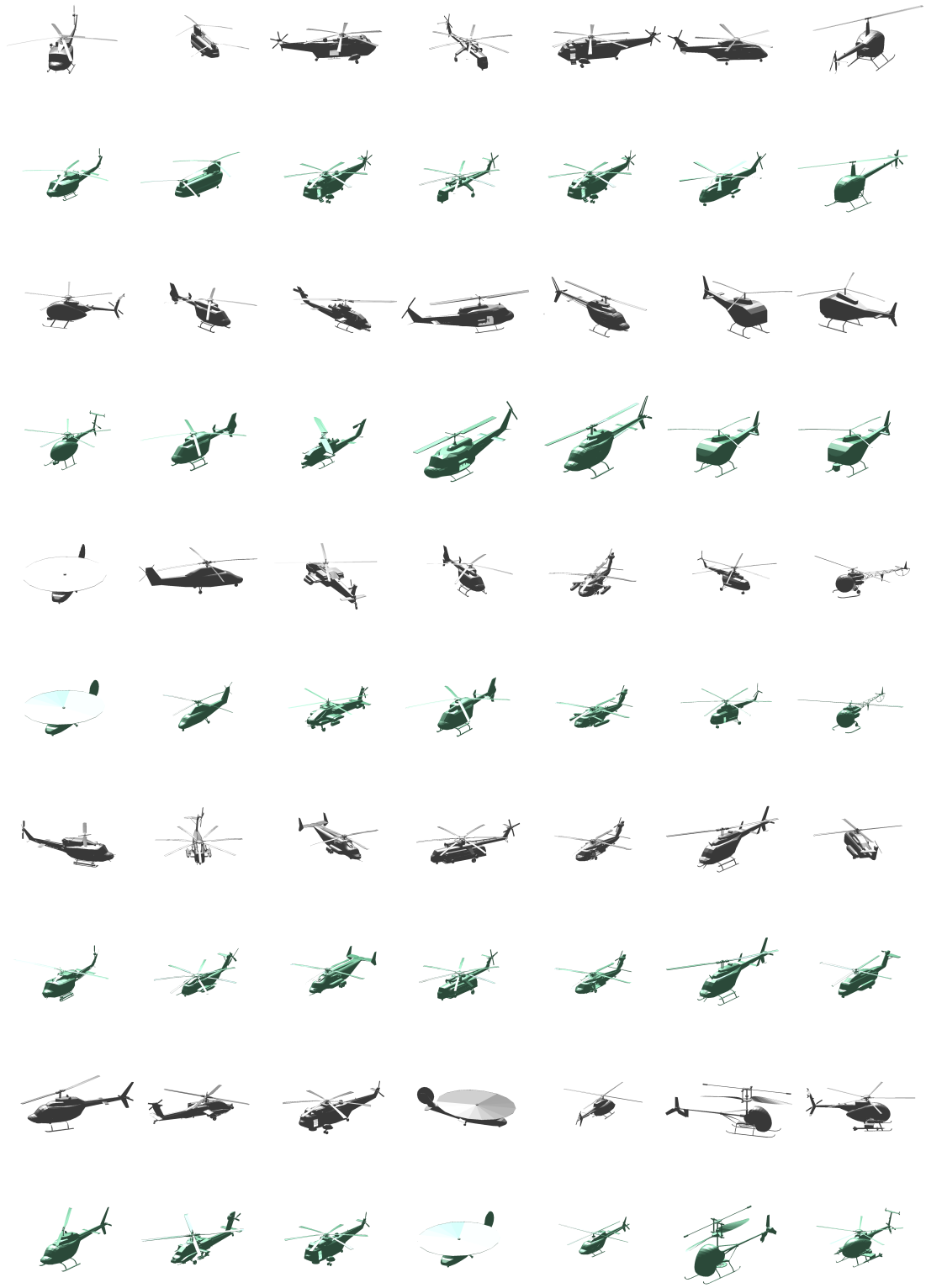
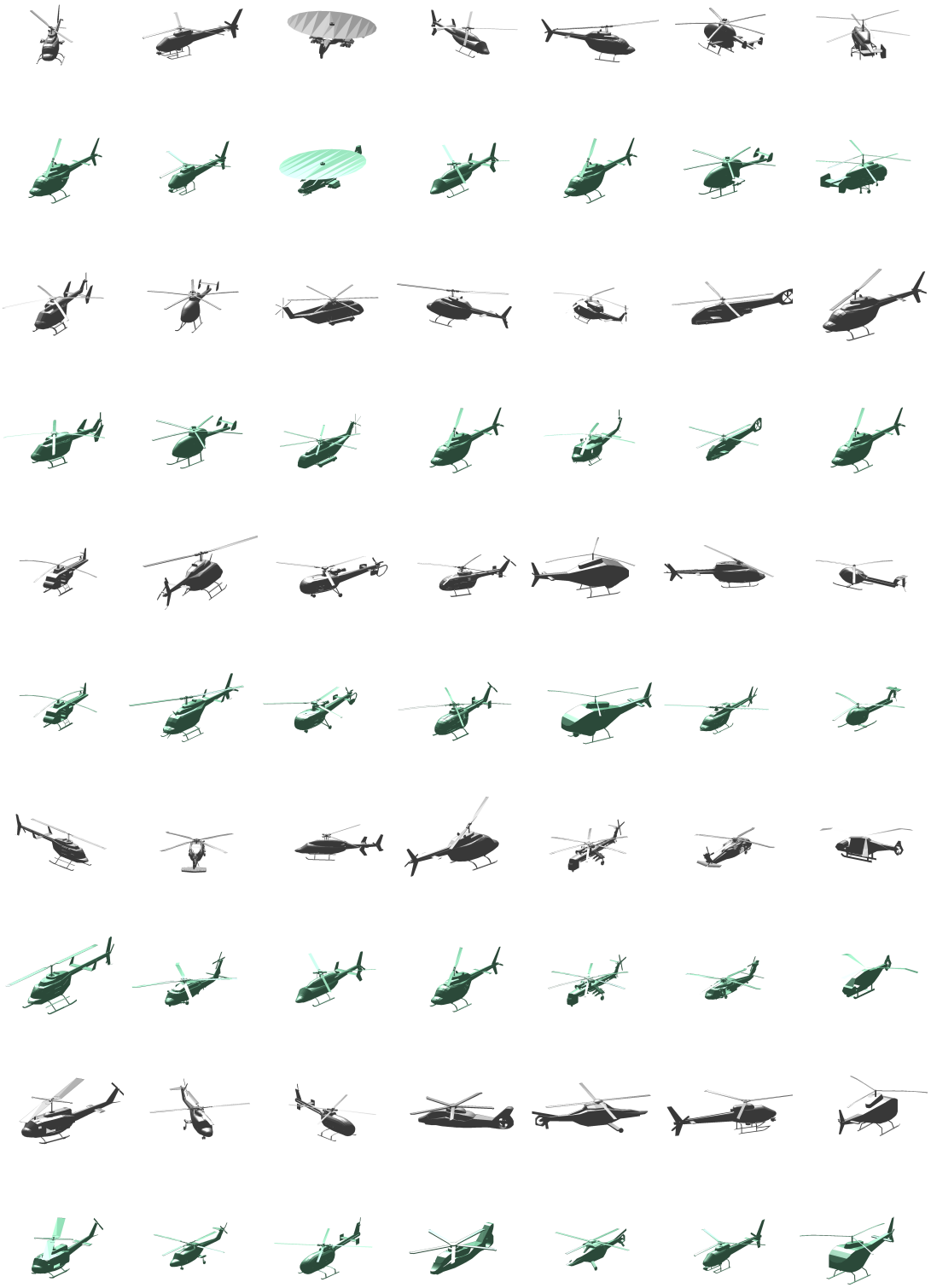


Figure A.10: Cups dataset before (odd rows - in gray) and after (even rows - in green) alignment.





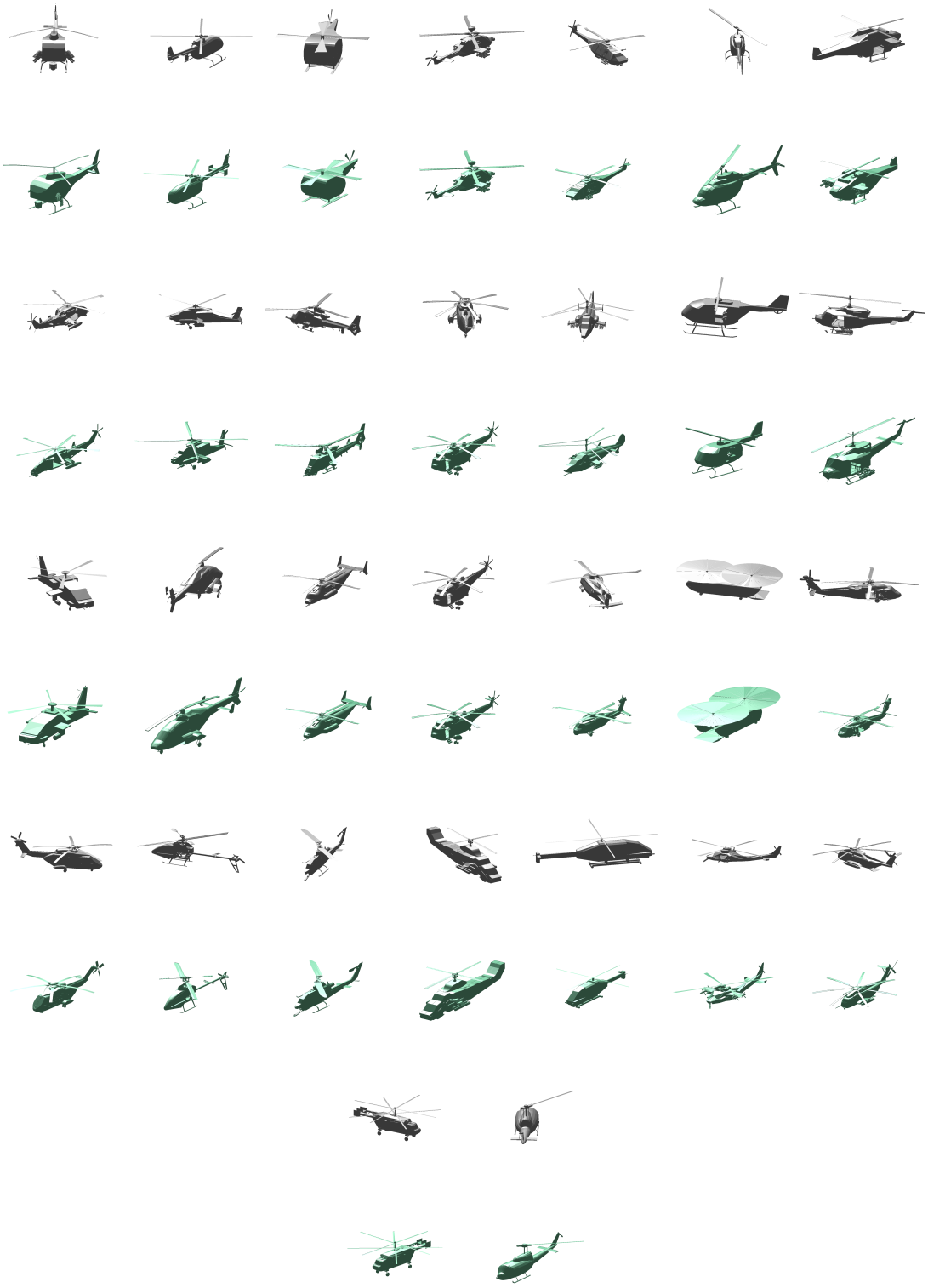
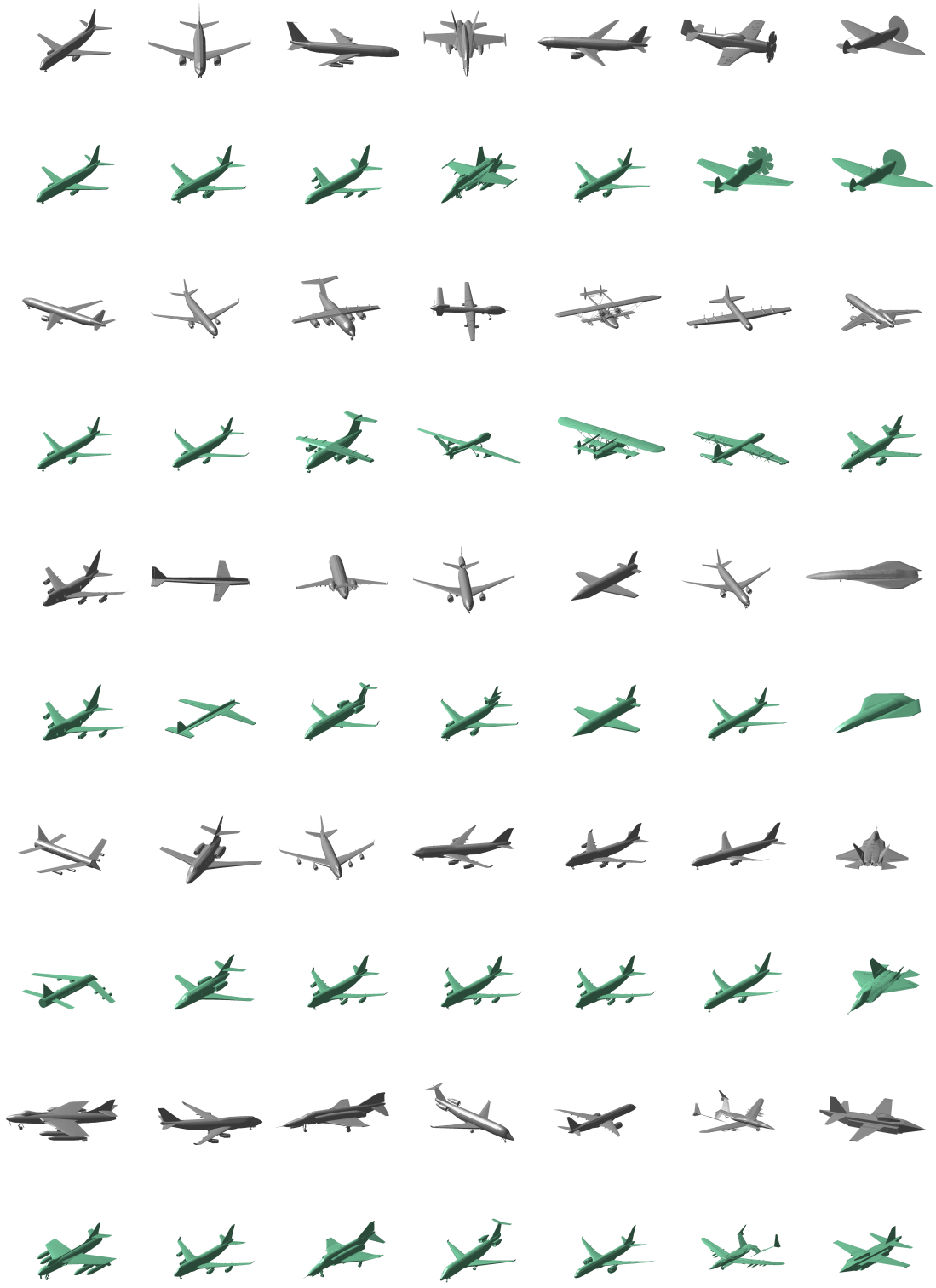
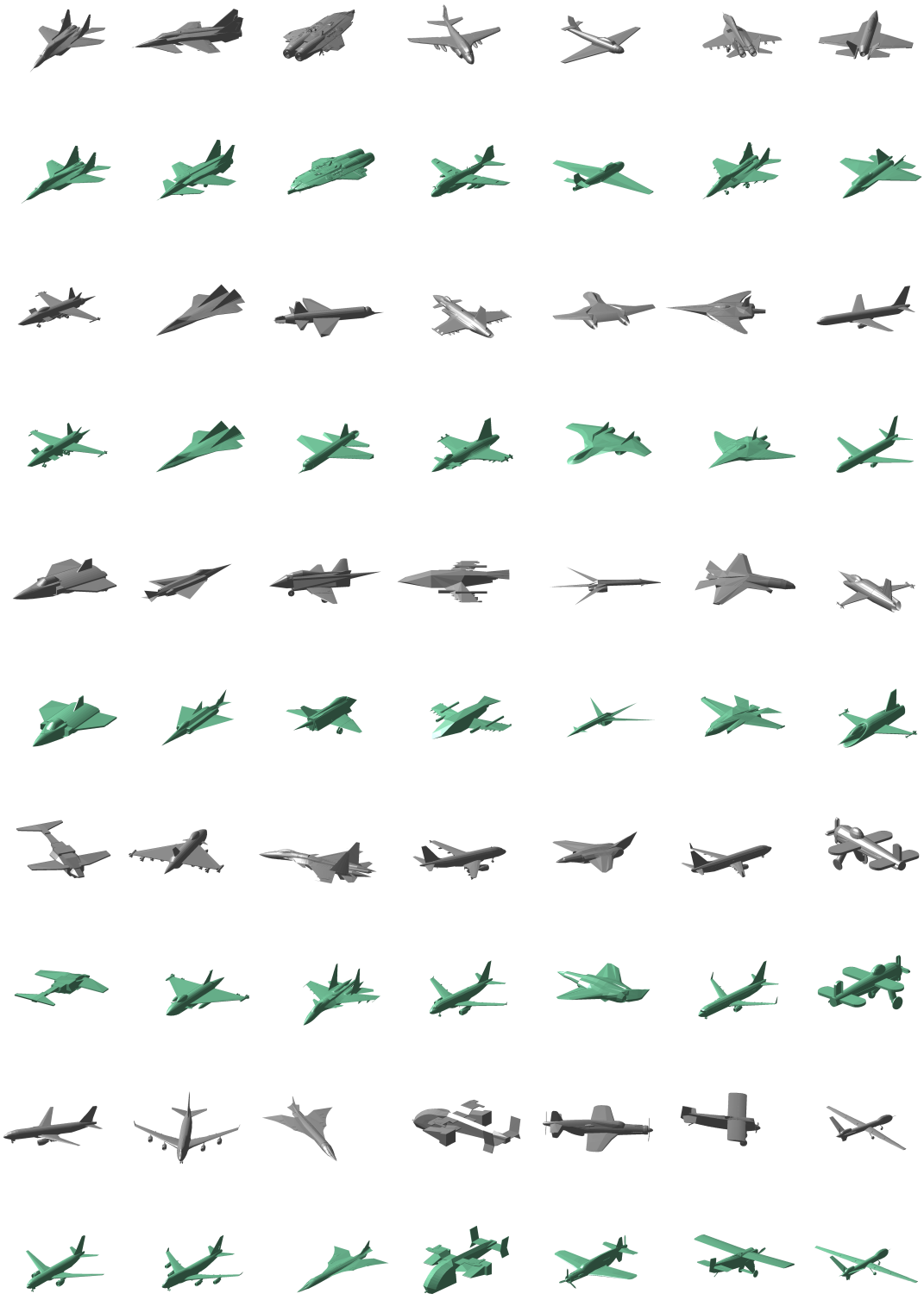


Figure A.11: Helicopters dataset before (odd rows - in gray) and after (even rows - in green) alignment.





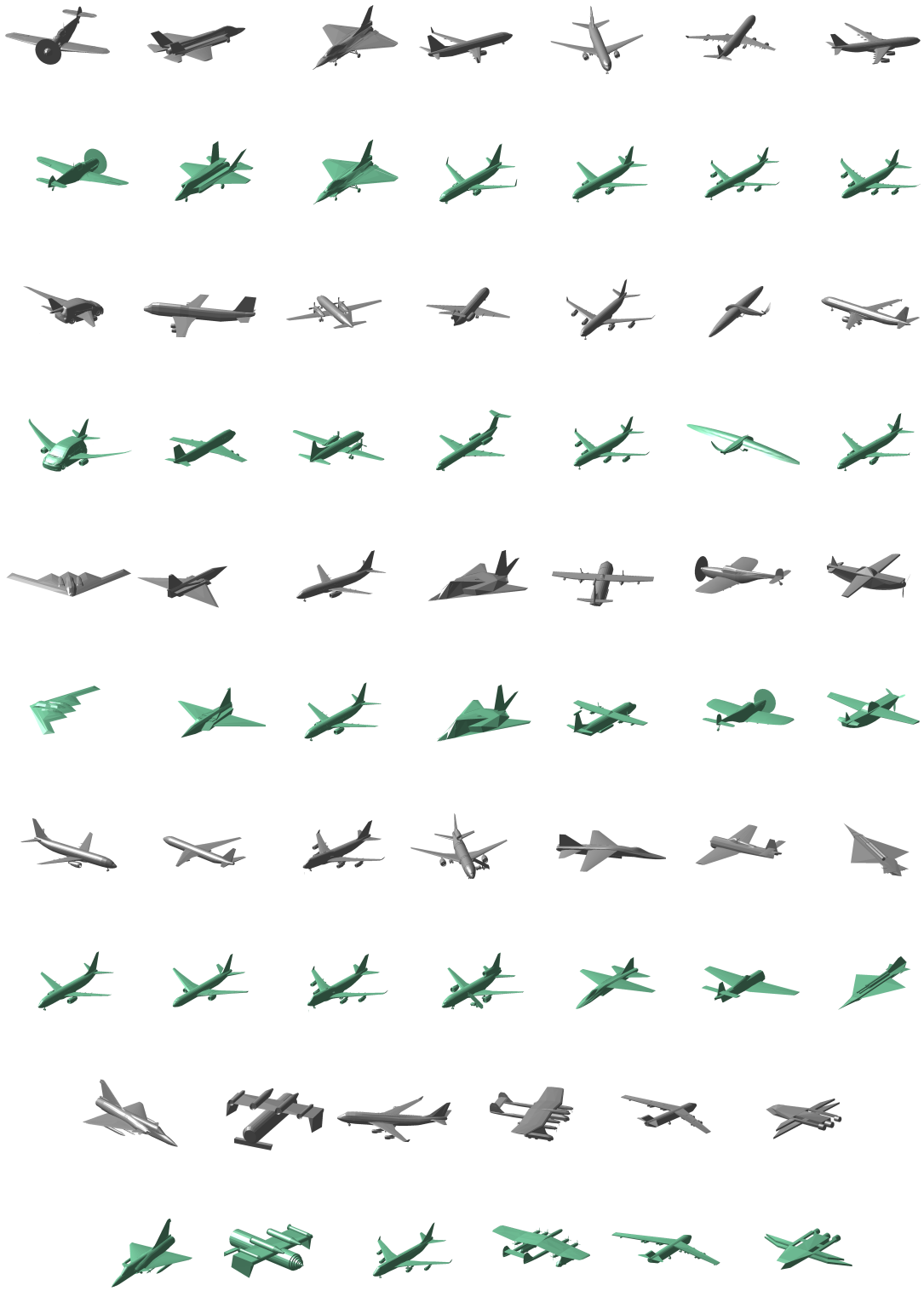
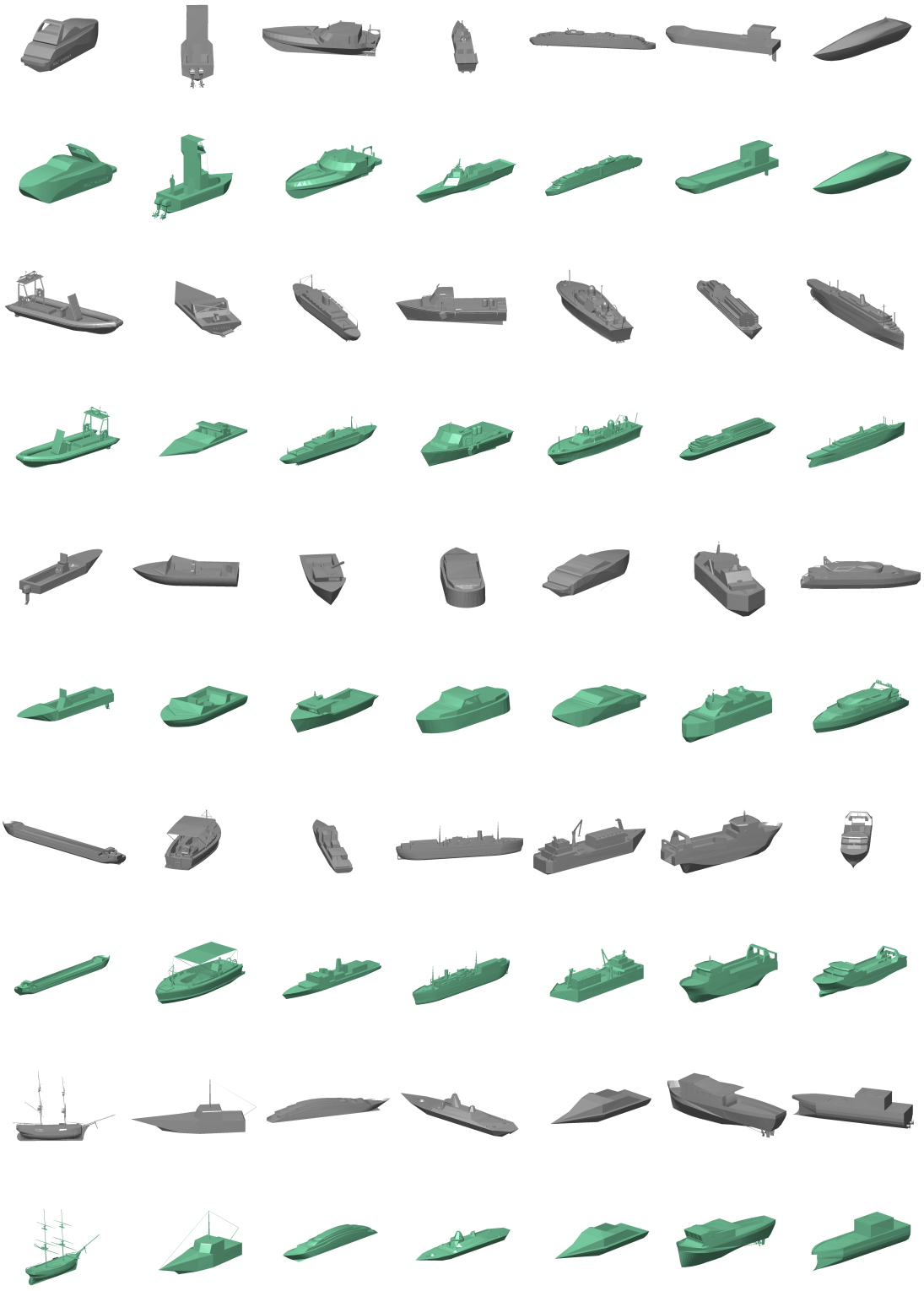
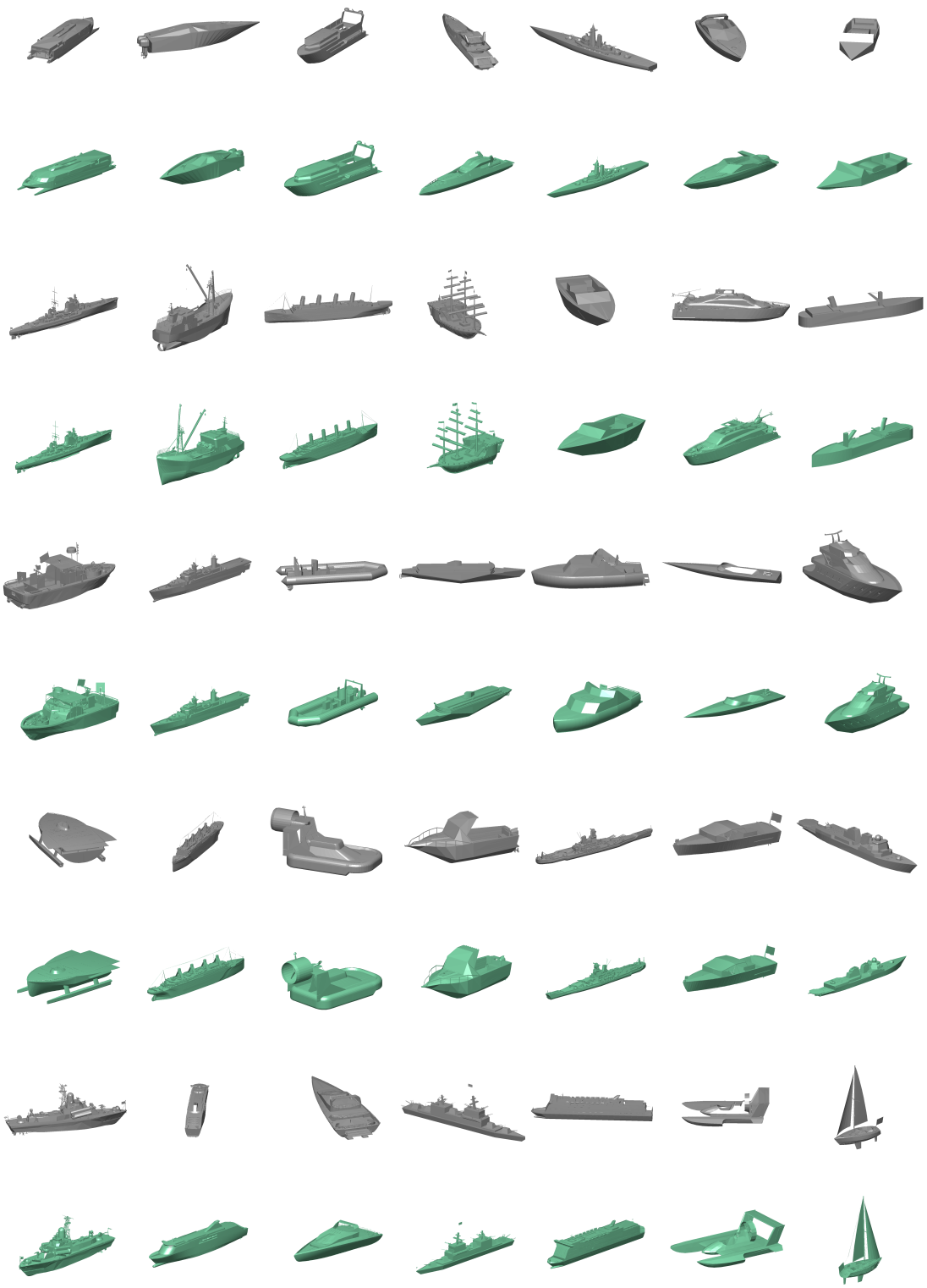


Figure A.12: Planes dataset before (odd rows - in gray) and after (even rows - in green) alignment.





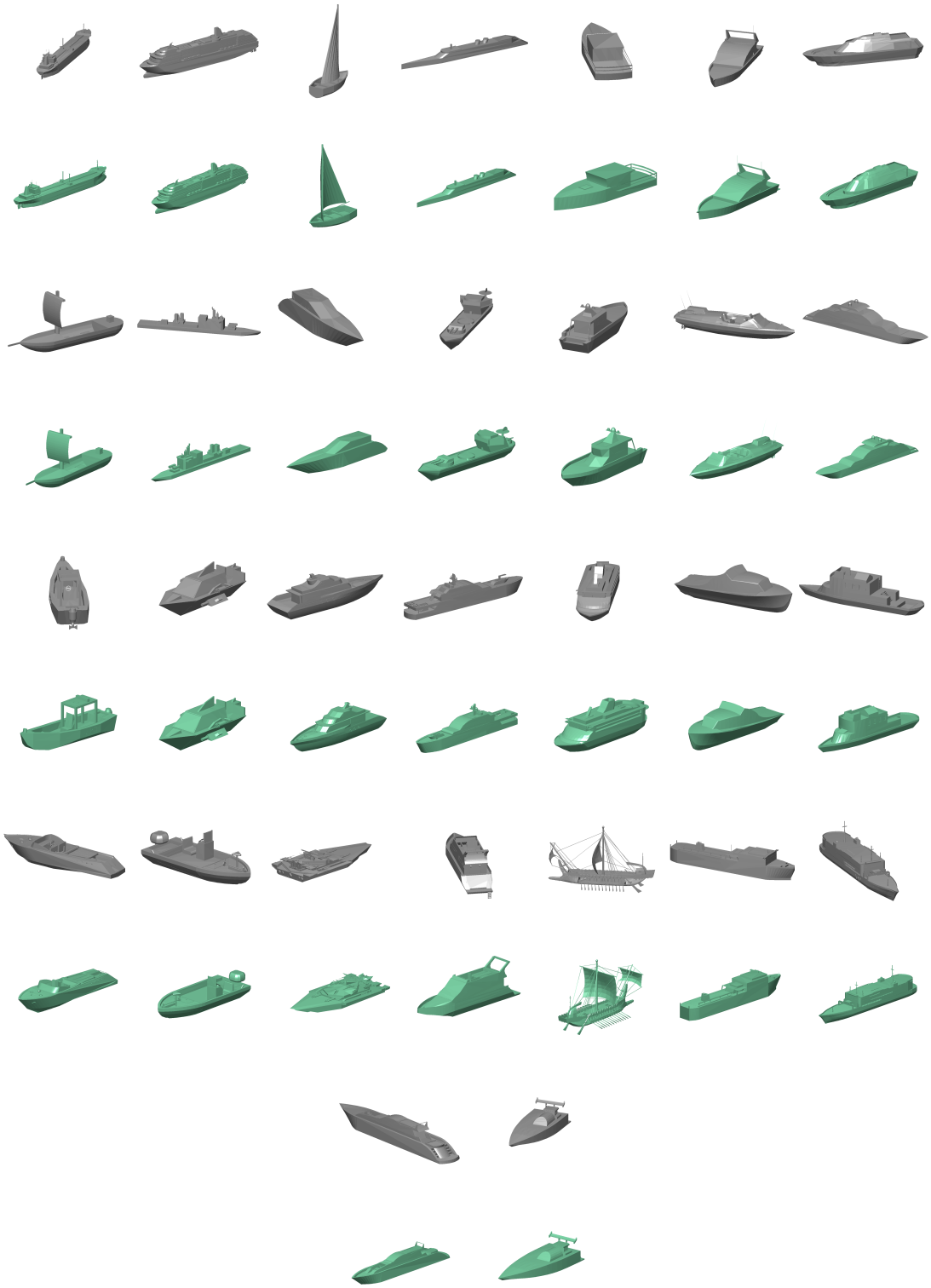
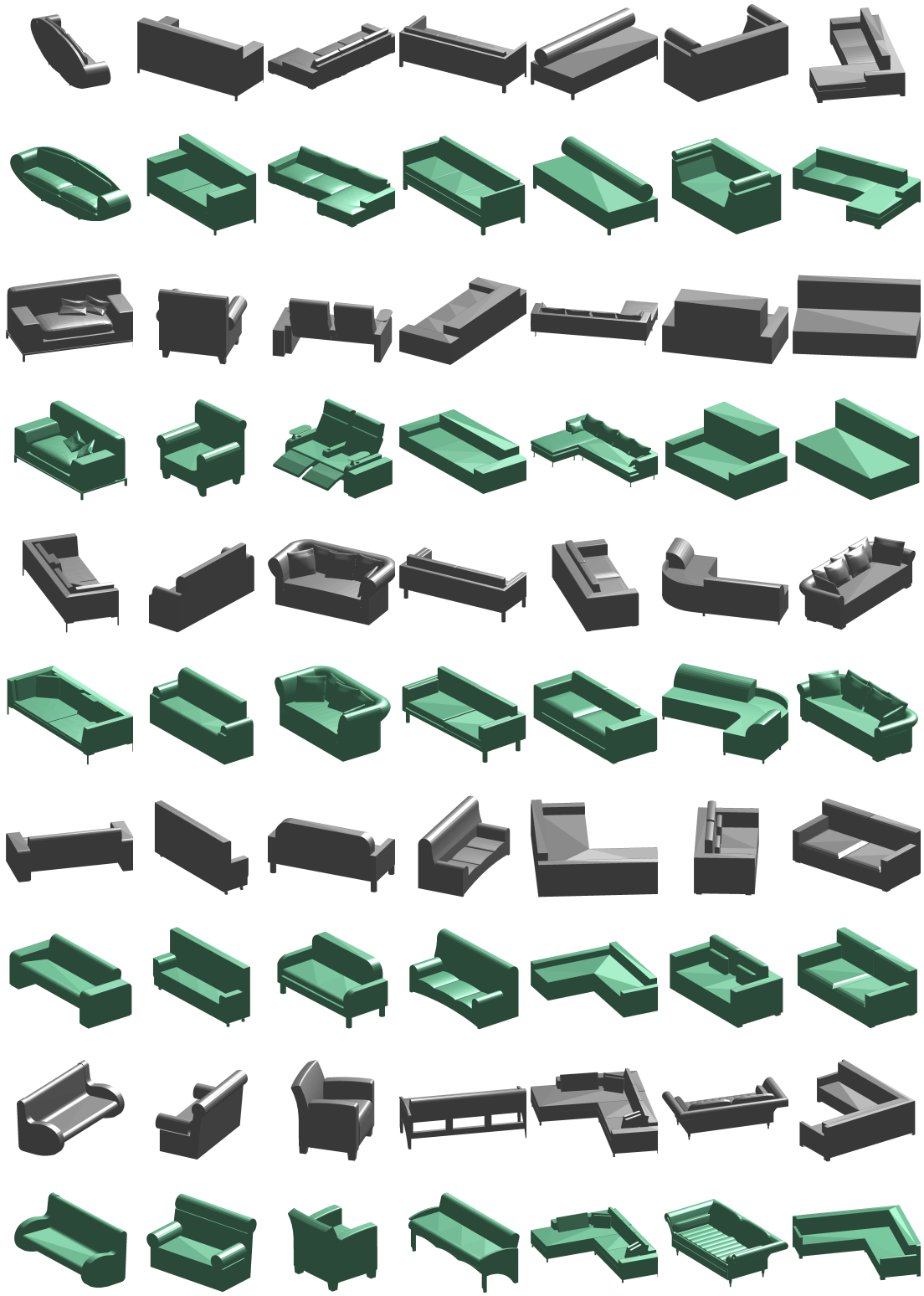
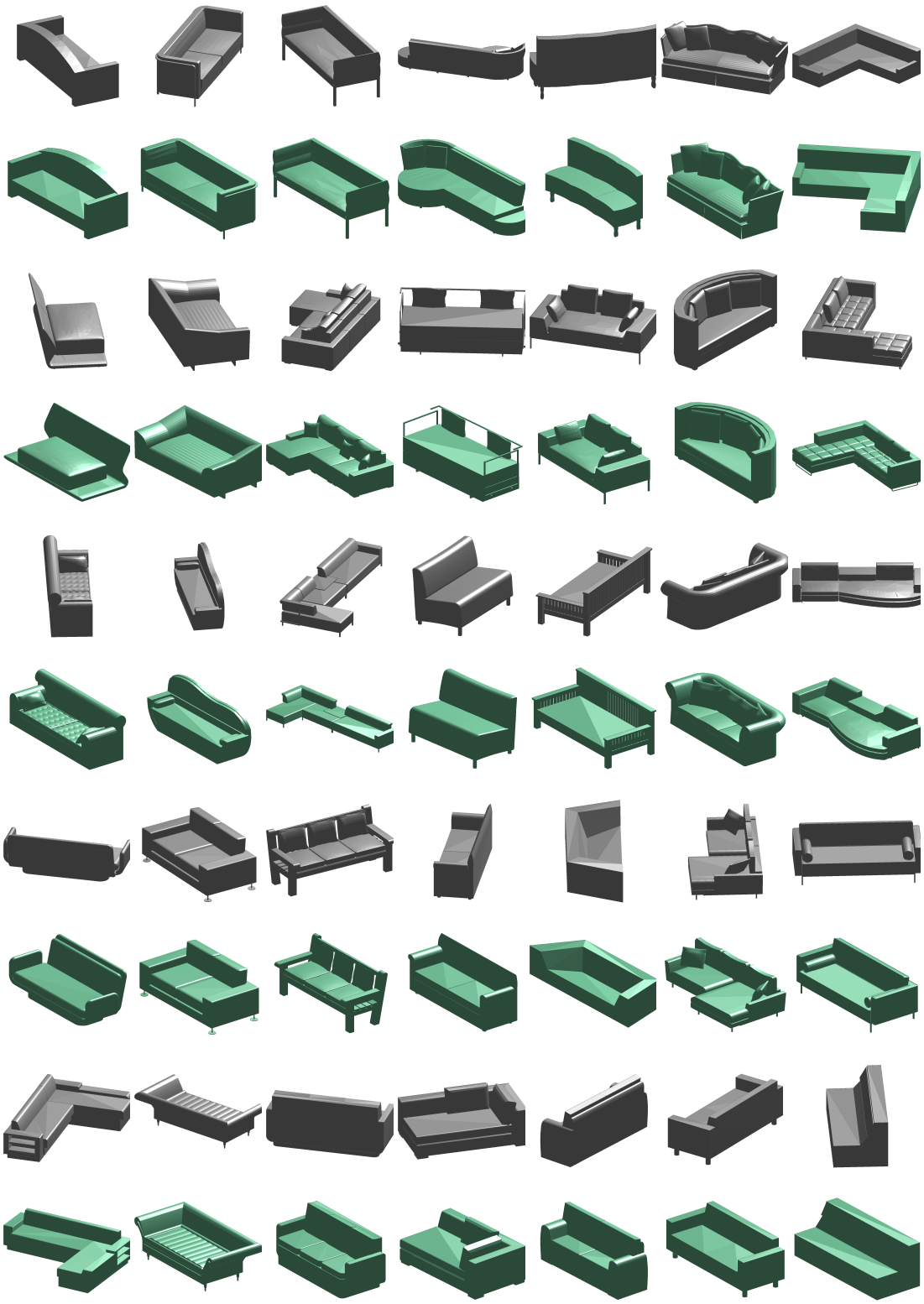


Figure A.13: Ships dataset before (odd rows - in gray) and after (even rows - in green) alignment.

Figure A.14: Snowmobiles dataset before (odd rows - in gray) and after (even rows - in green) alignment.





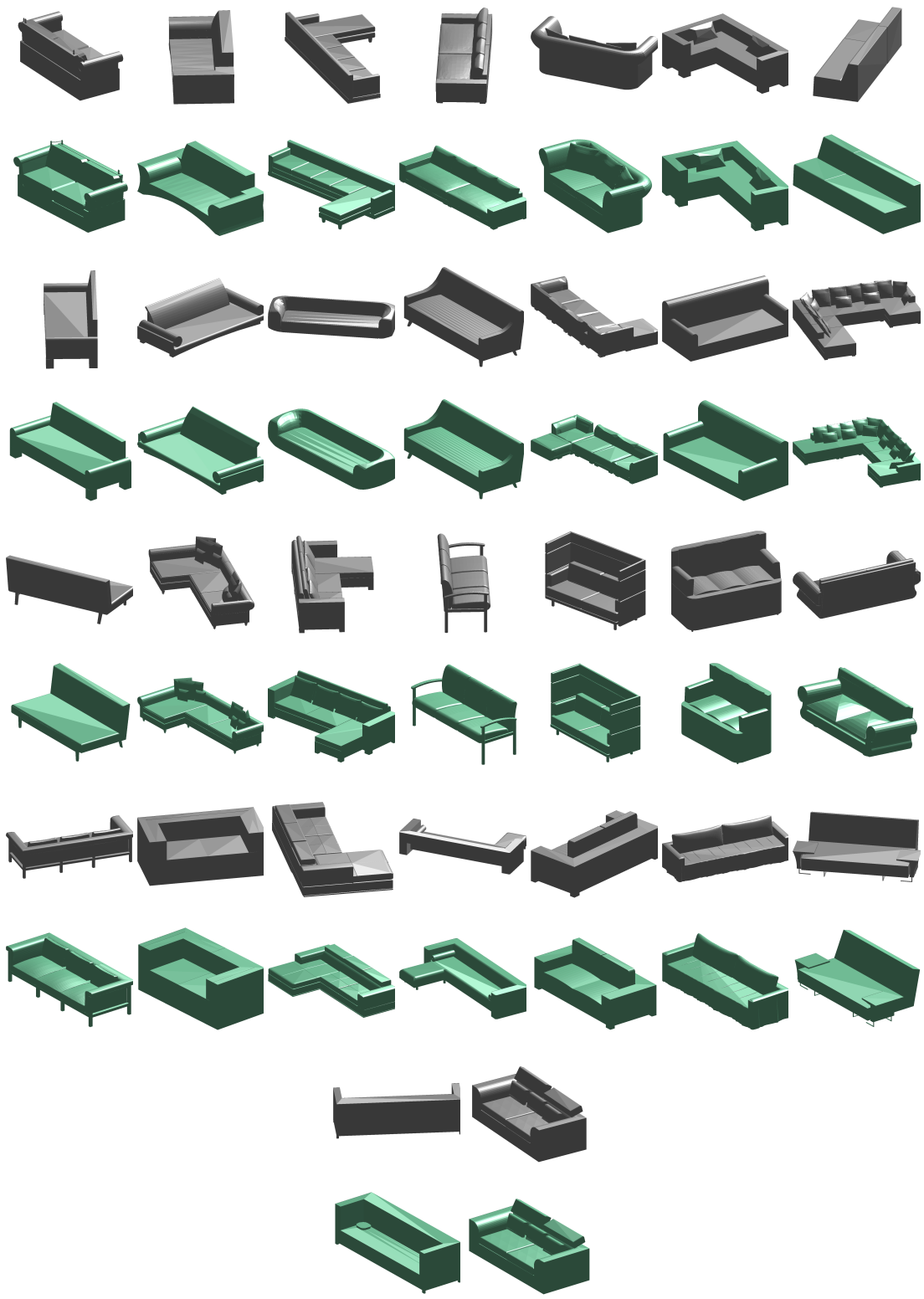


Figure A.15: Sofas dataset before (odd rows - in gray) and after (even rows - in green) alignment.

Appendix B

Template-based shape parameterization results

In this Appendix, we present the full set of results created by participants in our user study for evaluating our template-based shape parameterization method.

Full results for user experiments on Trimble 3D Warehouse datasets for our template-based parameterization method are presented in Figures B.1, B.2, B.3 (our method), and results from the user study comparing to Chaudhuri et al. [15] are presented in Figures B.7, B.9 (our method), and Figures B.8, B.10 (Chaudhuri et al. [15]).

(a) User: 1

(b) User: 2

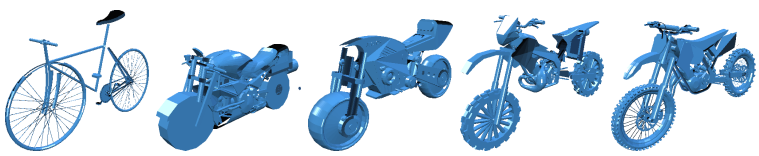
(c) User: 3

(d) User: 4

(e) User: 5

(f) User: 6

(g) User: 7



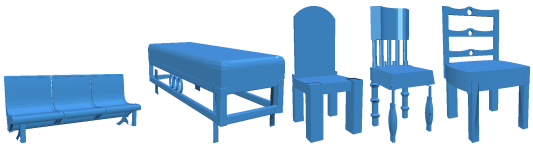
(h) User: 8

Figure B.1: Dataset: bike, Our method.

(a) User: 1

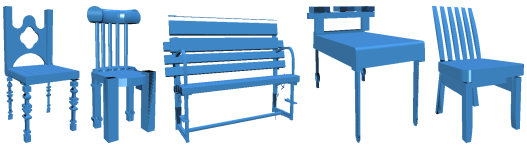
(b) User: 2

(c) User: 3



(d) User: 4

(e) User: 5



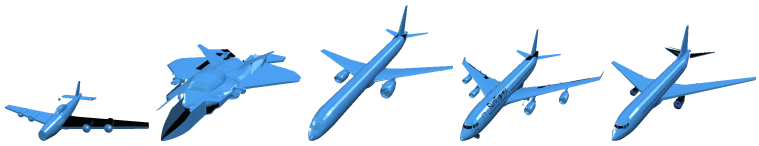
(f) User: 6

(g) User: 7

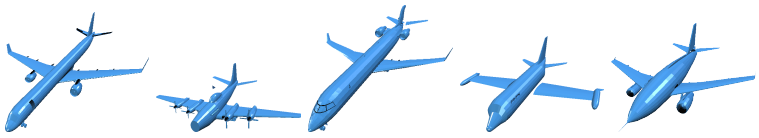
(h) User: 8

Figure B.2: Dataset: chair, Our method.

(a) User: 1



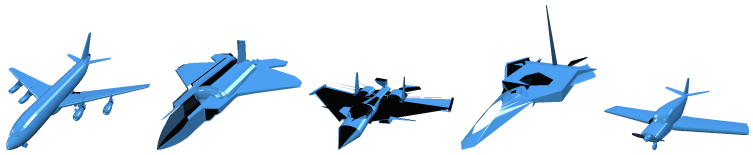
(b) User: 2



(c) User: 3

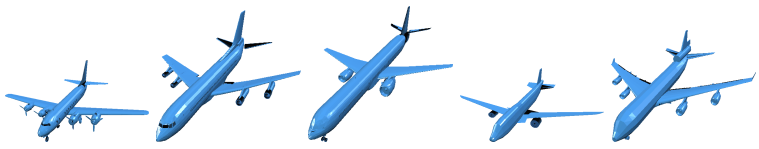
(d) User: 4

(e) User: 5



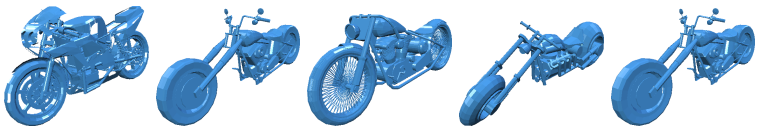
(f) User: 6

(g) User: 7

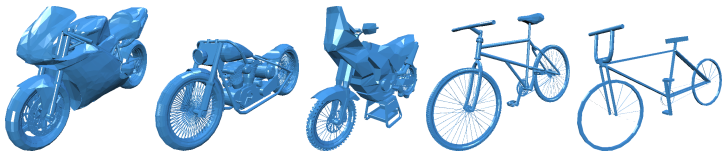


(h) User: 8

Figure B.3: Dataset: plane, Our method.

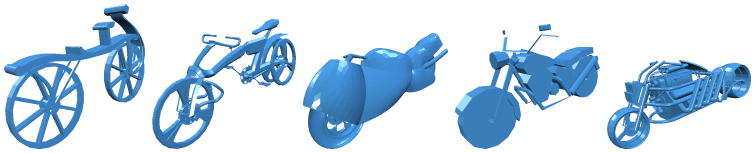


(a) Random Set: 1

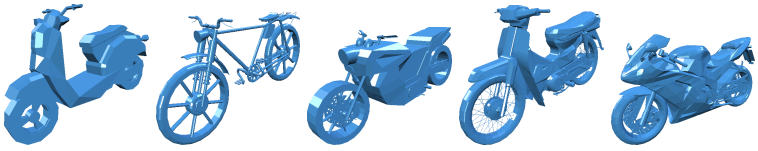


(b) Random Set: 2

(c) Random Set: 3



(d) Random Set: 4



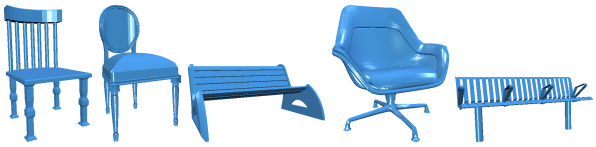
(e) Random Set: 5

(f) Random Set: 6

(g) Random Set: 7

(h) Random Set: 8

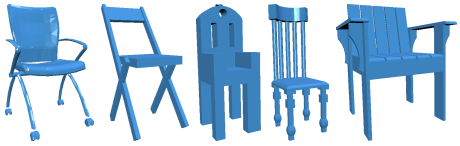
Figure B.4: Dataset: bike, Baseline.



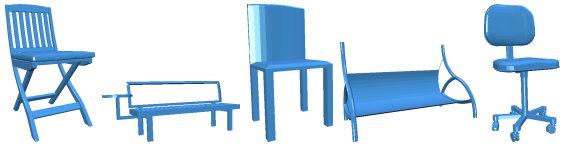
(a) Random Set: 1

(b) Random Set: 2

(c) Random Set: 3



(d) Random Set: 4



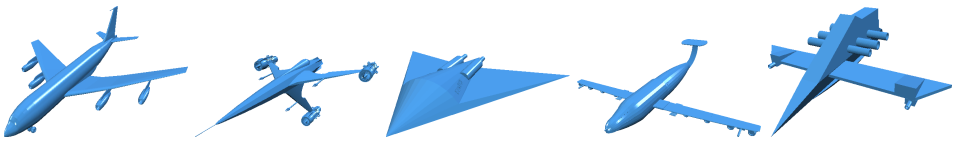
(e) Random Set: 5

(f) Random Set: 6

(g) Random Set: 7

(h) Random Set: 8

Figure B.5: Dataset: chair, Baseline.

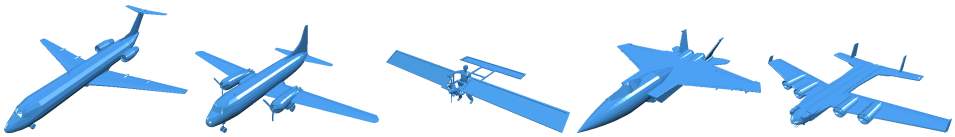


(a) Random Set: 1

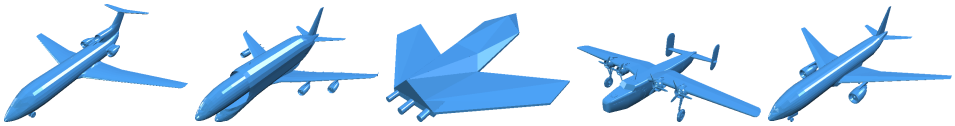
(b) Random Set: 2

(c) Random Set: 3

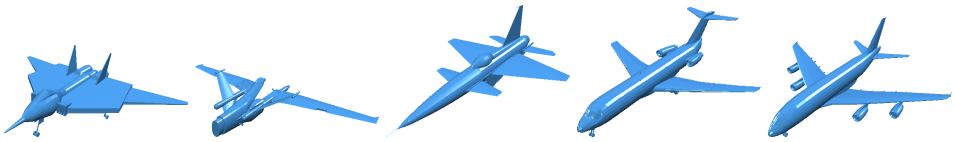
(d) Random Set: 4



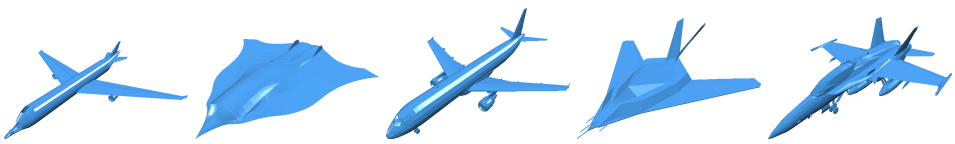
(e) Random Set: 5



(f) Random Set: 6



(g) Random Set: 7



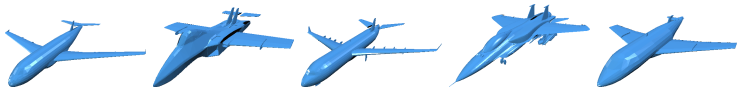
(h) Random Set: 8

Figure B.6: Dataset: plane, Baseline.

(a) User 9

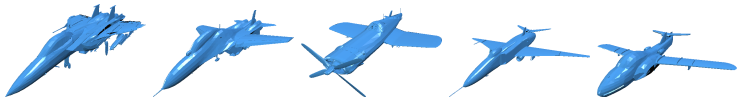
(b) User 10

(c) User 11

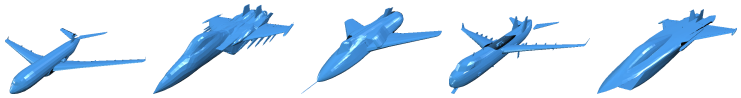


(d) User 12

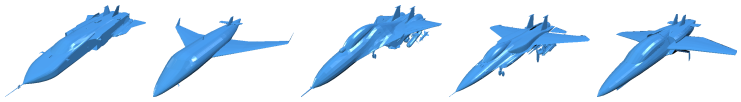
(e) User 13



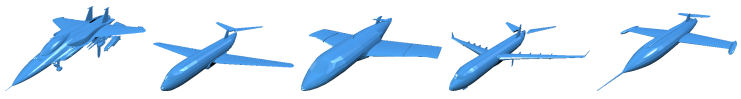
(f) User 14



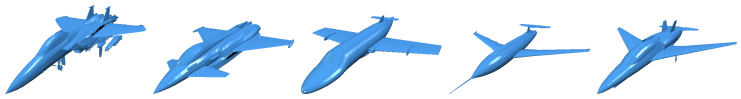
(g) User 15



(h) User 16

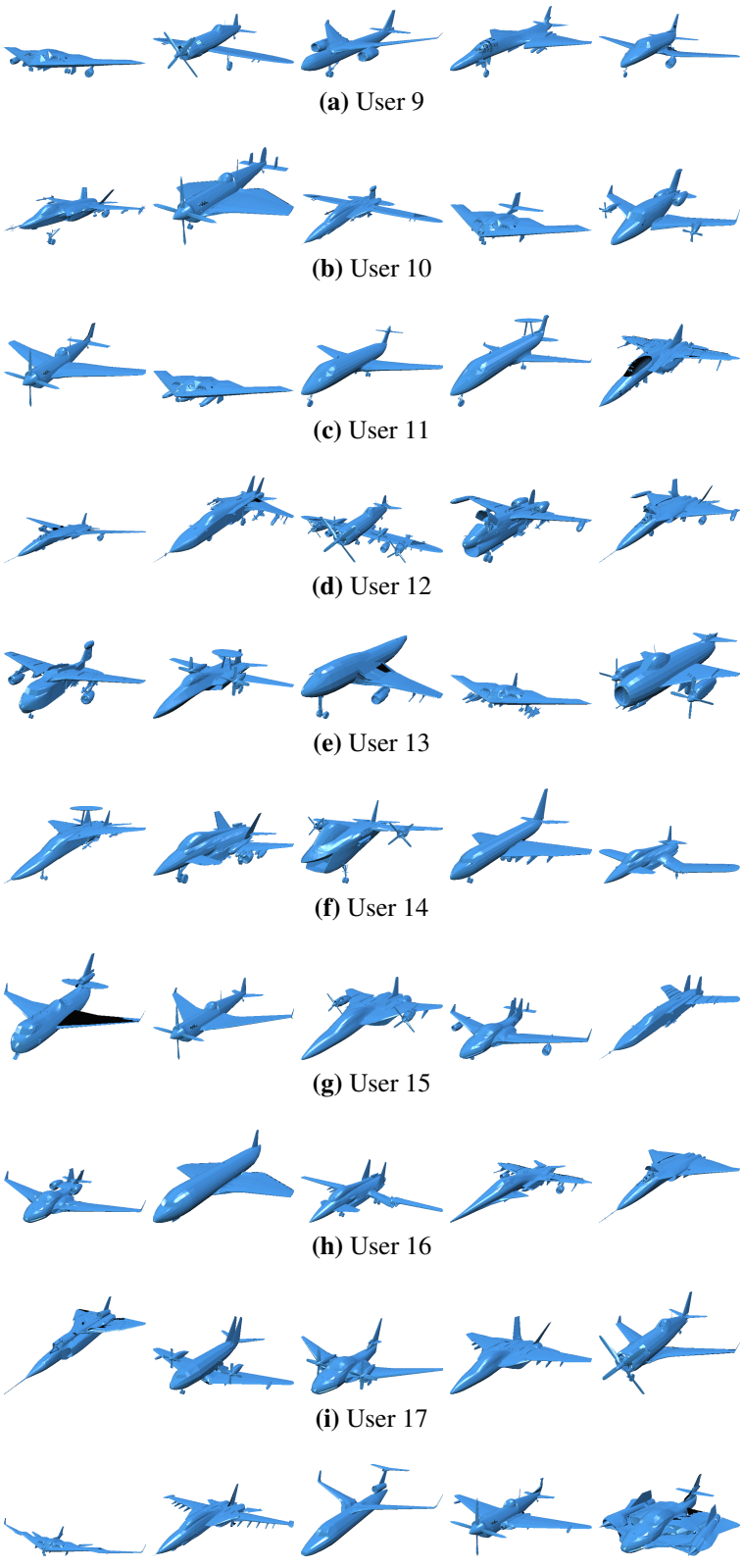


(i) User 17



(j) User 18

Figure B.7: Comparison: 100 airplanes, Task: **T1**, Method: Our method



(a) User 9

(b) User 10

(c) User 11

(d) User 12

(e) User 13

(f) User 14

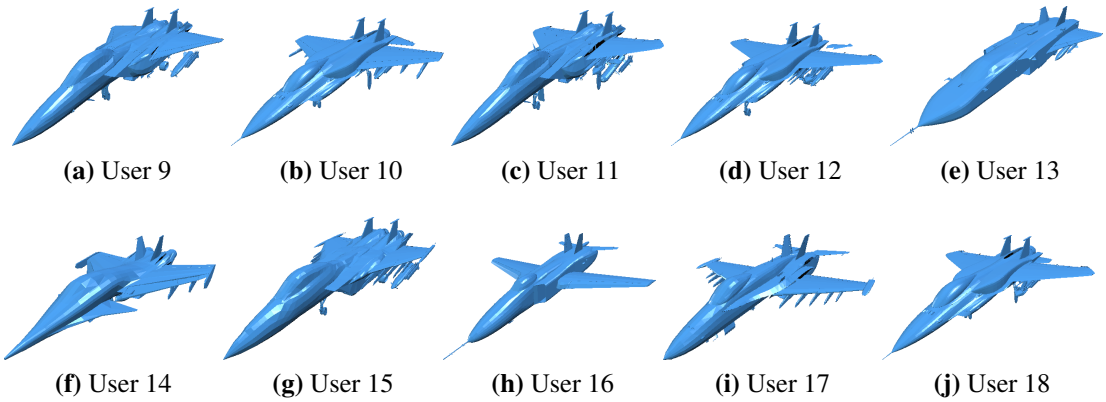
(g) User 15

(h) User 16

(i) User 17

(j) User 18

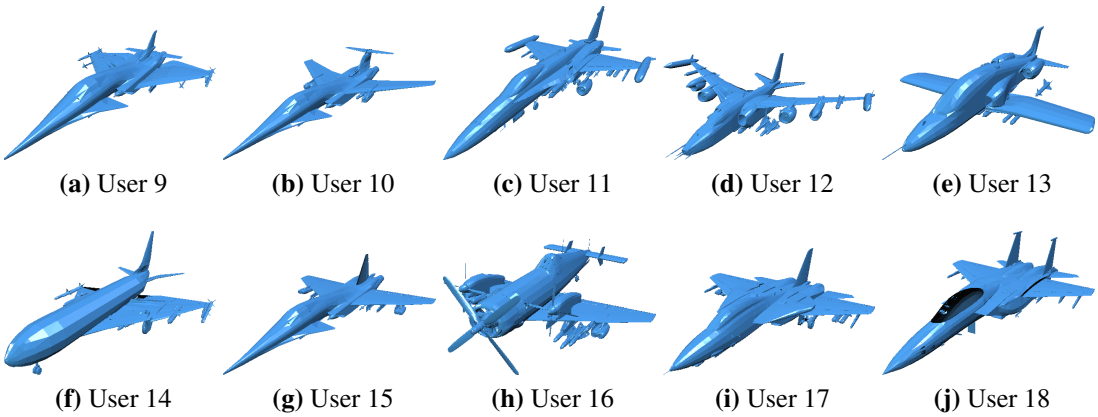
Figure B.8: Comparison: 100 airplanes, Task: **T1**, Method: Chaudhuri et al. [2011]



(a) User 9 **(b)** User 10 **(c)** User 11 **(d)** User 12 **(e)** User 13

(f) User 14 **(g)** User 15 **(h)** User 16 **(i)** User 17 **(j)** User 18

Figure B.9: Comparison: 100 airplanes, Task: **T2**, Method: Our method



(a) User 9 **(b)** User 10 **(c)** User 11 **(d)** User 12 **(e)** User 13

(f) User 14 **(g)** User 15 **(h)** User 16 **(i)** User 17 **(j)** User 18

Figure B.10: Comparison: 100 airplanes, Task: **T2**, Method: Chaudhuri et al. [2011]