CITY UNIVERSITY OF HONG KONG
香港城市大學

Sketch-based Shape and Structure Analysis in Design and Fabrication
在設計和製造中基於草圖的形狀和結構分析

Submitted to
School of Creative Media
創意媒體學院
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
哲學博士學位

by

YU Deng
于鄧

August 2023
二零二三年八月

Abstract

Sketching is a universal and intuitive tool for humans to render and interpret the visual world and is extensively used by designers in the product design and digital fabrication process. Since human viewers can easily envision the missing 3D information from a sparse, abstract, and imprecise sketch, they tend to use sketches to represent complex shapes based on notable advantages such as flexibility, concision, and efficiency. However, inferring the desired content from an input sketch or multi-view sketches is still highly challenging for machines due to the ill-posed nature. With the successful developments of deep learning techniques, such as implicit representation, image-to-image translation, and metric learning, we have more opportunities and general tools to solve challenging problems in sketch-based shape and structure analysis tasks. This dissertation explores the topic of sketch-based shape and structure analysis with the above advanced frameworks in three aspects: imperfect inputs, interaction with external physical factors, and multi-view inputs.

To enable people to perform the sketch-based shape and structural analysis in the design and fabrication process, we propose three deep learning-based techniques to assist users in three tasks, namely, beautifying imperfect freehand sketches, simulating shape structural stress for a single sketch under a user-specified force, and inferring correspondence from multi-view sketches.

Although sketches are widely studied and used in various sketch-based applications, existing algorithms are still struggling to directly make use of these freely drawn sketches that are usually drawn in an imprecise and abstract format, in particular, sketches created for depicting man-made objects with diverse geometry and non-trivial topology. We present a novel freehand sketch beautification method, which takes as input a freely drawn sketch of a man-made object and automatically beautifies it both geomet-

rically and structurally. Beautifying a sketch is challenging because of its highly abstract and heavily diverse drawing manner. Existing methods are usually confined to the distribution of their limited training samples and thus cannot beautify freely drawn sketches with rich variations. To address this challenge, we adopt a divide-and-combine strategy. Specifically, we first parse an input sketch into semantic components, beautify individual components by a learned part beautification module based on part-level implicit manifolds, and then reassemble the beautified components through a structure beautification module. With this strategy, our beautification method can go beyond the training samples and handle novel freehand sketches by learning both the possible part geometries and the plausible combinations of individual components. We demonstrate the effectiveness of our sketch beautification system with extensive experiments and a perceptive study.

In the process of product design and digital fabrication, structural analysis of a designed prototype is a fundamental and essential step. However, such a step is usually invisible and agnostic to designers in the early sketching phase. This limits users' ability to contemplate a shape's physical properties and structural soundness. To bridge this gap, we present *Sketch2Stress* that allows users to perform structural analysis of desired objects at the sketching stage. *Sketch2Stress* takes as input a sketch and a point map to specify the location of a user-assigned external force. It automatically predicts a normal map and a corresponding structural stress map distributed over the user-sketched underlying object. In this way, *Sketch2Stress* empowers designers to easily examine the stress sustained everywhere and identify potential problematic regions over their sketched object. Furthermore, combined with the predicted normal map, users are able to conduct a region-wise structural analysis efficiently by aggregating the stress effects of multiple forces in the same direction. We demonstrate the effectiveness and practicality of *Sketch2Stress* system with extensive experiments and user studies.

The above two works focus on the analysis and processing of single-view sketches. However, interpreting missing 3D information from the single-view sketch only is still challenging for existing computer algorithms, especially the sketch-based shape reconstruction approaches. Therefore, multi-

view inputs are often needed and used in the aforementioned algorithms to reduce the inherent ambiguity in single-view sketches and recover the underlying 3D geometry faithfully. The third technique aims at automatically computing the semantic shape correspondence among multi-view freehand sketches created for the same objects. Correspondence matching is a fundamental but still an open problem in the research community. This problem is more challenging for multi-view sketches since the visual features of corresponding points can be very sparse and vary significantly across different views. To solve this problem, we present *SketchDesc* to learn a novel local sketch descriptor from data. We further contribute a training dataset by generating the pixel-level correspondence for the multi-view line drawings synthesized from 3D shapes. To handle the sparsity and ambiguity of sketches, we design a novel multi-branch neural network that integrates a patch-based representation and a multi-scale strategy to learn the pixel-level correspondence among multi-view sketches. Through extensive experiments on hand-drawn sketches and multi-view line drawings rendered from multiple 3D shape datasets, we demonstrate the effectiveness of *SketchDesc*.

CITY UNIVERSITY OF HONG KONG
Qualifying Panel and Examination Panel

Surname: YU
First Name: Deng
Degree: Doctor of Philosophy
College/Department: School of Creative Media

The Qualifying Panel of the above student is composed of:

Supervisor(s):

Prof. FU Hongbo	School of Creative Media City University of Hong Kong

Co-supervisor(s):

Prof. LAU Chung Man Manfred	School of Creative Media City University of Hong Kong

Qualifying Panel Member(s):

Prof. LIAO Jing	Department of Computer Science City University of Hong Kong

This thesis has been examined and approved by the following examiners:

Prof. FU Hongbo	School of Creative Media City University of Hong Kong
Prof. ZHU Kening	School of Creative Media City University of Hong Kong
Prof. CHU Hung-Kuo	Department of Computer Science National Tsing Hua University
Dr. YU Qian	School of Software Beihang University

Acknowledgement

First and foremost, I want to thank my supervisor, Prof. Hongbo Fu, for guiding me through my PhD study, consolidating my theoretical knowledge that paves the way for my dissertation, reminding me to think further, teaching me how to do research in a scientific way, and many more during my PhD life. I am lucky to be co-supervised by Prof. Manfred Lau, who always brings up sparkling points and polishes my rough ideas. I sincerely appreciate the suggestions, inspiration, and encouragement of my supervisors whom I respect most.

Many thanks to my thesis examining committee members, Prof. Kening Zhu, Prof. Qian Yu, and Prof. Hung-Kuo Chu for their patience in reviewing and constructive comments. I would like to thank the collaborators of my various projects that form the fundamental bases of my dissertation, in particular, Dr. Youyi Zheng from the State Key Lab of CAD&CG, Zhejiang University, Dr. Lei Li and Prof. Chiew-Lan Tai from HKUST, Prof. Yi-Zhe Song from the SketchX Lab, University of Surry, Dr. Lin Gao from the Institute of Computing Technology, Chinese Academy of Sciences, and Chufeng Xiao from the School of Creative Media.

I also thank my lab mates for fruitful discussions on research and memorable moments in daily life. Special thanks to Kin Chung Kwan, Yilan Chen, Pui Chung Wong, Wanchao Su, Hui Ye, Jie Zhou, Linzi Qu, Qiaochu Wang, Xuanyu Wang, Jingyuan Liu, Zeyu Hu, and Xuyang Bai. Many thanks to my dear friends and PhD classmates for their tremendous helps during my tough time: Yaozhun Huang, Ann Mak, Hang Ji, Yujia Zhang, Xiao Wang, Mengjun Lin, Bin Chen, Lingyan Ruan, Chanjun Mu, Mengjun Li, Shaoyu Cai, Pingchuan Ke, Taizhou Chen, Lifeng Huang, Miaomiao Qi, Haokai Song, and Xiaobo Ke.

Finally, I would like to give my greatest thanks to my family, especially my parents and my elder sister. They are the ones who unconditionally love

me, support me, relieve me, and motivate me. Thanks for everything you gave to me.

Contents

List of Figures

List of Tables

1 Introduction

Sketching is an important tool in product design and digital fabrication. Putting sketching assistance into consideration, e.g., quickly setting up an indoor scene with a sketched office-style long table alongside two sides chairs in a meeting room, can greatly facilitate the expression and improve the believability of the creation of designers by specifying the general environment and curving the essential objects in a depicted scenario. However, the further step, shape and structure analysis of the sketched object under the external environmental forces is still missing in the research community and remains challenging in creating structure-sound sketch prototypes and refining the original sketch design iteratively to direct the subsequent fabrication steps in the real production process.

Despite the abstract nature of sketches, human viewers can implicitly beautify sketches and easily envision their underlying 3D objects in the real world, whether from single or multiple views. On the one hand, humans have an innate perception of similarity and discrepancy in the shape and structure of sketched objects, e.g., humans can easily recognize geometry and structure discrepancies within an arrangement of sketched objects, like, distinguishing swivel chairs from four-leg chairs. This inspires us to explore machine learning approaches to understand the perceptual shape and structure similarity in human sketches. On the other hand, designers extensively use sketching to curve the shape and structure of desired product prototypes. However, they have no way of analyzing the structural soundness of their created sketch prototypes immediately, even for experienced designers, not to mention refining their original design at the sketching stage. This is because structural analysis of designed prototypes is the latter fundamental step after the sketching step in the conventional product design and digital fabrication process [52], as shown in Figure 1.1, and aims to verify whether designed products can withstand realistic forces

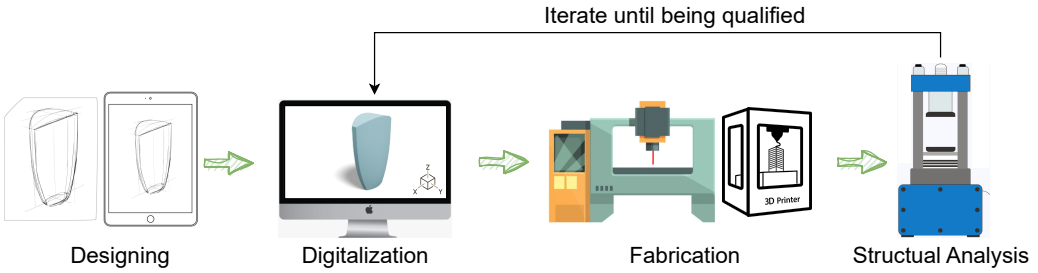


Fig. 1.1. Illustration of the general steps in the product design and digital fabrication process. Designers first utilize sketches to express their ideas and creation in the designing step. After a digital 3D model is created in the digitalization step, the fabrication step further fabricates it into the physical product. The structural analysis step then verifies whether this final product can pass the compression testing and further refines the product structure in the previous digitalization and fabrication steps iteratively. The separation between the designing step and its afterwards steps makes it challenging for designers to analyze the structure-soundness of their designed product during sketching.

in the wild world or under specified scenarios, e.g., the soundness of chair legs to undertake the weight of an extremely heavy person.

As the physical stress undertaken by designed prototypes needs to be mathematically/physically precise considering the inputs of structure, functions, and external forces, this step is commonly operated on 3D shapes [121, 113, 155, 58, 126]. Since 3D shapes are the most common and informative representations encoding highly detailed geometric features and miscellaneous shape structures, they are frequently used to simulate complex motions and display lifelike interactions with physical factors. But adapting the structural analysis task from informative 3D objects to sparse 2D sketches is nontrivial due to the ill-posed nature of the 2D sketch and the high-dimension and complex representation of the interactive external forces (locations, magnitudes, directions). This problem is more challenging when input sketches are rough and imprecise. Inspired by advanced techniques, such as implicit representation [88, 76], image-to-image translation [49, 132], and metric learning [122, 74, 123], we provide our solutions specifically designed for the aforementioned tricky problems in sketch-based shape and structure analysis, such as beautifying imperfect freehand sketches, interactively analyzing the shape structure of sketches

under specified external force configurations, and inferring semantic shape correspondence from multi-view sketches.

We present three deep learning techniques to assist non-professional users in sketch-based shape and structure analysis in different aspects: A sketch beautification system helps interactively beautify an input freely drawn sketch of a man-made object both geometrically and structurally; *Sketch2Stress* guides users in analyzing the structural weakness of their sketched object under external forces simulating physical factors and further refining their original drawings iteratively to more structurally-sound sketch prototypes; *SketchDesc* predicts the semantic shape correspondence for different kinds of multi-view sketches;

In our sketch beautification system (Chapter 3), we first decompose an input sketch to part sketches and introduce a novel implicit representation for part sketches, allowing for smooth interpolation between two samples with large geometry discrepancies. We then adopt a retrieval-and-interpolation way to beautify the geometry of part sketches and design a novel spatial transformation network to further refine the global structure of the entire input sketch. In *Sketch2Stress* (Chapter 4), we first decouple the high-dimension and complex forces to the constant force magnitude and direction based on the estimation of a 2.5D normal map. Then we utilize an effective data-driven framework to approximate the mathematically/physically precise stress by constructing a novel large-scale sketch-force-stress dataset and proposing a new two-branch (for force location and direction) generation pipeline. Finally, we provide a new sketch-based interface for novice users to support sketch-based weakness analysis and structural refinement (for adjusting the original design). In *SketchDesc* (Chapter 5), we transform the correspondence learning problem to the metric learning problem, where we utilize the triplet loss [99] to shrink the distance between feature descriptors of the corresponding points among multi-view sketches and expand the distance between ones that are not corresponding. In this way, we present a pixel-wise sketch descriptor that can easily find the corresponding points among the multi-view sketch inputs.

In the design and fabrication process, our proposed systems focus on several specified types of sketches. Figure 1.2 and Table 1.1 clarify and display

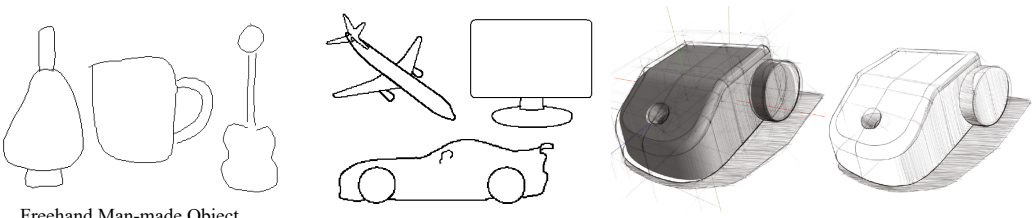


Fig. 1.2. Different types of sketches utilized by our proposed systems in the design and fabrication process.

System	Roles	Usage Scenario	Input Sketch
Sketch Beautification	Beautify User's Freehand Man -made Object sketches	Single-view	Freehand Man-made Object Sketch
Sketch2Stress	Analyze User's Sketched Struc -tures under Specified Forces	Single-view	Freehand/Product Sketch
SketchDesc	Extend Single-view Applications to Multi-view Scenario	Multi-view	Freehand/Product/Synthetic Sketch

Tab. 1.1. Roles of our proposed systems in the sketch-based shape and structural analysis in design and fabrication.

the types of input sketches used by our different algorithms, namely, freehand man-made object sketches, synthetic sketches, and product sketches, respectively. Typically, freehand sketches are created by novice users with non-professional drawing skills and design experience. The specified freehand man-made object sketches are also created by novice users but are further utilized to depict 3D man-made objects that can be semantically segmented partly. Synthetic sketches, on the other hand, are 2D edge maps directly rendered from existing 3D shapes using non-photorealistic rendering (NPR) algorithms such as canny edge detection [11]. As for product sketches [38], they are used by professional designers to capture the essential geometry and structure of real products in the product design and digital fabrication process.

We use Table 1.1 and Figure 1.3 to illustrate the roles and connections related to the topic of sketch-based shape and structure analysis and demonstrate their contributions to the current design and fabrication process. As shown in Table 1.1, our sketch beautification system aims to assist users in beautifying their freely sketched man-made objects both geometrically and structurally during the design and fabrication process. And our *Sketch2Stress* enable users to perform structural analysis on their created

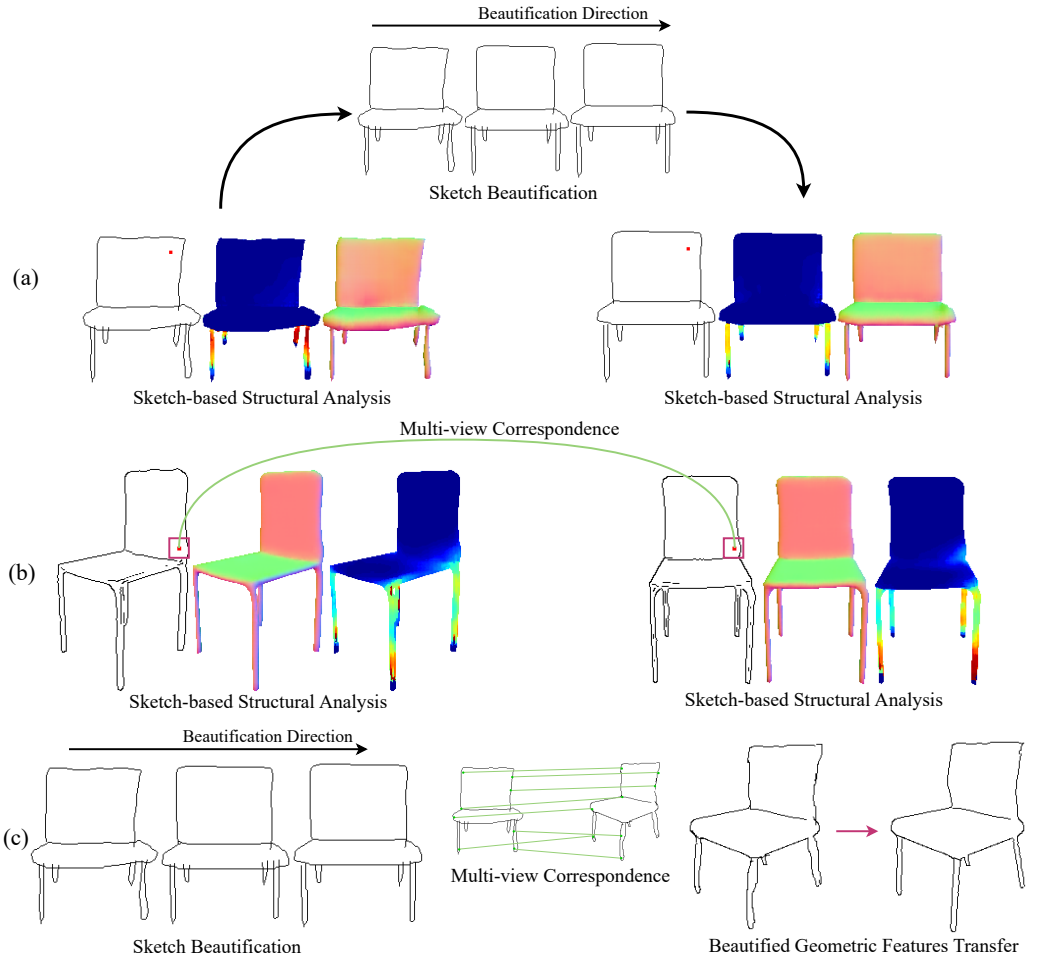


Fig. 1.3. Illustration of connections in our proposed three systems: the relationship between sketch beautification system and *Sketch2stress* (a), the benefits of using our *SketchDesc* to *Sketch2Stress* (b) and sketch beautification system (c), respectively.

sketch structures under external environmental forces. Both the sketch beautification system and *Sketch2Stress* are used in the single-view scenarios. However, in the design process, multi-view sketches are more useful and representative than single-view sketches. Therefore, our *SketchDesc* computes the semantic correspondence among multi-view sketches and further extends the single-view applications to the multi-view scenario (see Figure 1.3 (b) and (c)). Figure 1.3 (a) shows the structural stress analysis results of a user's freehand sketch (the left part) and its beautified version (the right part) using our sketch beautification system (the upper middle). It is evident that our sketch beautification system can further benefit both our *Sketch2Stress* and *SketchDesc* by improving the user's im-

perfect freehand sketch into a better format with refined part geometry and global structure. Furthermore, our *Sketch2Stress* can reflect such improvement on the structure after beautification (see fewer high-stress regions with warmer colors in the right-side structural stress map). Figure 1.3 (b) further displays a multi-view structural analysis scenario by incorporating *SketchDesc* into *Sketch2Stress* to associate the corresponding force points applied to the multi-view sketches of the same objects. As demonstrated in Figure 1.3 (c), combining the sketch beautification system with our *SketchDesc* can further transfer the beautified local geometric features to other views by the multi-view correspondence computed by *SketchDesc*.

1.1 Beautification for Imperfect Sketches of Man-made Objects

Due to frequently-observed defects in people's freehand sketches, such as poorly drawn curves, imperfect straight lines, detached and inter-penetrated parts, and unclosed shape boundaries, there still is a gap between the user's freehand sketches and what the existing sketch-based algorithms [24, 72] really wanted. Although the beautification problem has been studied for decades, ranging from the primitives of sketched geometric objects [128] to strokes in handwritings [158], systematic analysis of beautifying sketches for man-made objects has rarely been studied. The key challenge here is instantiating poorly drawn conceptual geometries and refining imprecise structures simultaneously. Addressing the beautification problem of man-made object sketches can inspire and facilitate various downstream sketch-based applications such as sketch-based modeling [107, 24, 119], sketch-based retrieval [13, 21], and other sketch understanding tasks [147].

As sketches are widely utilized in manufacture designing [62, 38, 61], animation drafting [116, 105] and HCI (Human-Computer Interaction) [145], many algorithms have been presented to process freely drawn sketches targeting vectorization [86], rough sketch cleanup [105], and simplification [127, 68]. These methods are common in two places: first, they usually produce more local modifications on input sketches but seldom consider or

touch the global structures; also, the input of these methods is nearly ready-to-use sketches with perfect strokes that are straight, mutually parallel, or curved with perpendicular angles. Hence, the aforementioned approaches are inapplicable to our task of beautifying the artifacts in the man-made object sketches with diverse local distortion and global inconsistency. As for the drawing assistance to the overall structures, prior approaches [2, 26, 60] are mainly designed in a heuristic and interactive way that provides a holistic scaffold or shadow guidance, and updates their guidance based on drawn strokes during the drawing process. Alternatively, Fišer et al. [32] proposed a rule-based stroke beautification approach, which, however, is still conditioned on the former strokes and confined to the stroke sequences. Therefore, these methods cannot be applied as post-processing to the existing hand-drawn sketches without the drawing order of individual strokes.

We observe that the traditional data-driven methods [105, 106] usually lack robustness and fail to produce satisfactory beautification results for freely-drawn sketches of man-made objects, especially when the inputs are created with large variations. It is because their well-trained inference models are heavily confined to the distribution of the limited training samples. We are primarily inspired by CompoNet[98], which jumps out of the bound of the empirical distribution of the observed data and enriches the distribution diversity of generated samples by learning both novel part synthesis and plausible part compositions. Unlike their generation task that tries to synthesize infinite outputs from finite inputs, our beautification task can be regarded as an opposite counterpart progress that aims to beautify infinite freehand inputs to finite outputs.

In our sketch beautification approach, we treat a sketched object as a combination of sketched parts or part sketches and further transform the sketch beautification problem into two sub-problems: part sketch beautification and global structure beautification. Given a freehand input sketch, we expect to beautify the local geometry defects in the part beautification module and address the overall structure issues in the structure beautification module. To better instantiate the user's conceptual sketches, we make several strict constraints for this task. First, during the process of part beautification, one should follow the user's original drawing intention as much

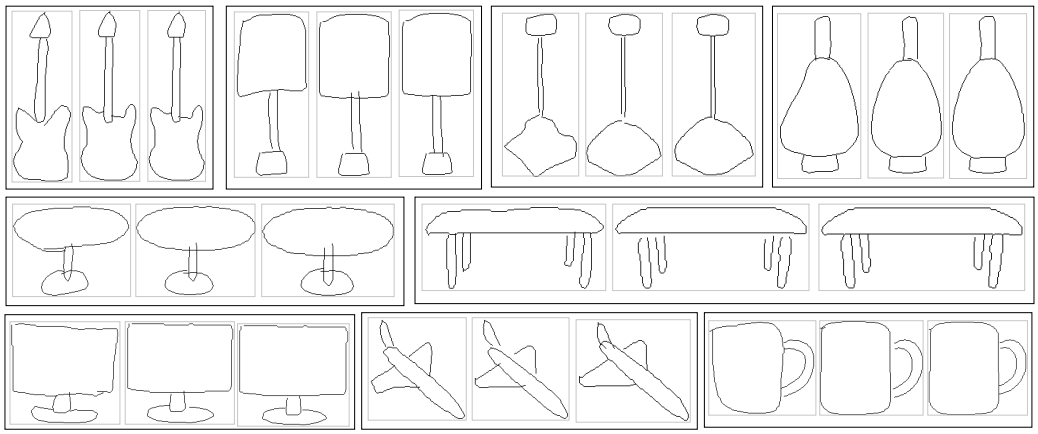


Fig. 1.4. We present a novel technique for beautifying freehand sketches of man-made objects. Each triplet contains an input sketch (left), the sketch after part beautification (middle), and the final result after structure refinement (right).

as possible. Second, during the process of structure beautification, one should maintain the user's original structure as much as possible. Therefore, directly replacing the user's input part sketch with a better part sketch retrieved from the database is not considered in our sketch beautification method. Besides, the large transformation and structure vibrations are not what we aim for. Therefore, we perform a mild refinement on the input sketch in terms of both geometry and structure (see Figure 1.4).

In the part beautification stage, we argue that existing deep sketch representations (i.e., CNN features from rasterized pixel maps [149] or RNN features from stroke sequences [41]) are insufficient to perform beautification on users' conceptual freehand sketches, as shown in Figure 1.5. CNN representations are widely utilized in natural image synthesis and can easily generate plausible and novel samples by interpolating existing images [138] or the disentangled semantic attributes [53]. However, different from natural images, sketches are usually too sparse, with few valid elements or points that cannot be smoothly interpolated with CNN representations. As for RNN representations, existing methods cannot guarantee a precise reconstruction of input sketches, not to mention the more challenging interpolation task. In our sketch beautification approach, we propose a novel implicit sketch representation, which is not only able to represent sketches effectively but also can construct a continuous and smooth manifold to

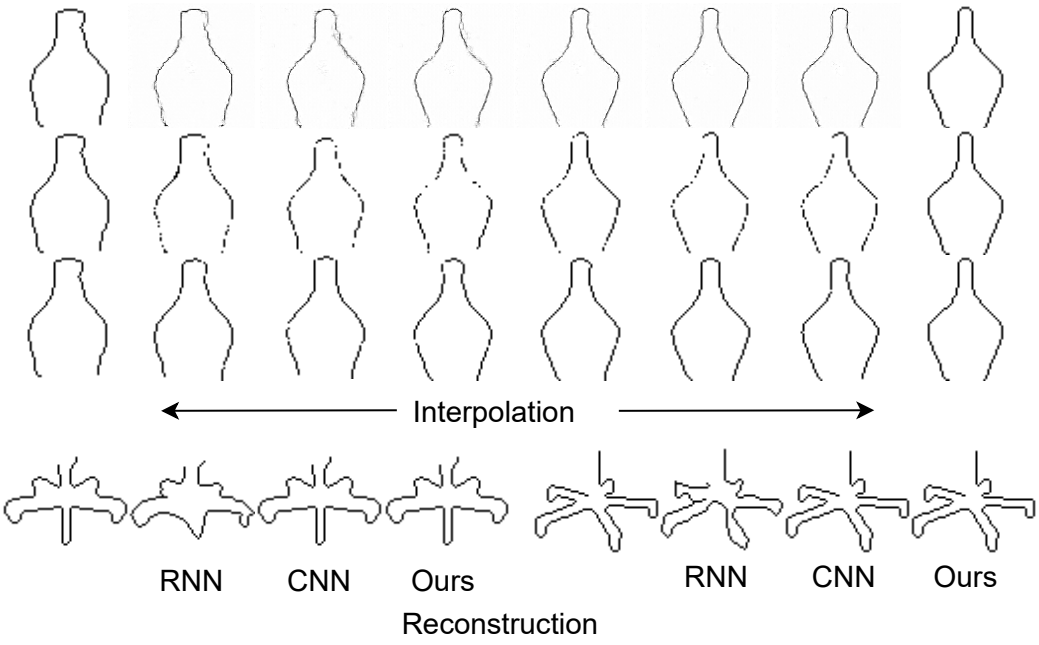


Fig. 1.5. Sketch interpolation based on CNN (first row), CNN after clean (second row), and our implicit representation (third row), and reconstruction with different representations (Bottom). Please zoom in to examine the details (blurs and diffusion artifacts surrounding the interpolated strokes of the CNN-based interpolation results (first row) and breaking strokes and missing parts after cleaning the diffused noises (second row)).

synthesize and instantiate users' conceptual drawings by interpolating the existing sketches.

In the structure beautification stage, to simulate the structural errors that exist in the user's input sketches, we apply random affine transformations to the ground-truth sketches in our training dataset and enforce our designed sketch assembly model to learn these transformations between the transformed part sketches and their ground truth counterparts. Then, simply feeding the input sketches to our sketch assembly model usually causes learning vibration and failures in model convergence due to the sparsity of the input sketches. To address this issue, we use part-level bounding boxes to enhance the spatial feature of the separate part sketches. We propose an IoU metric to further to evaluate the performance of the sketch assembly model.

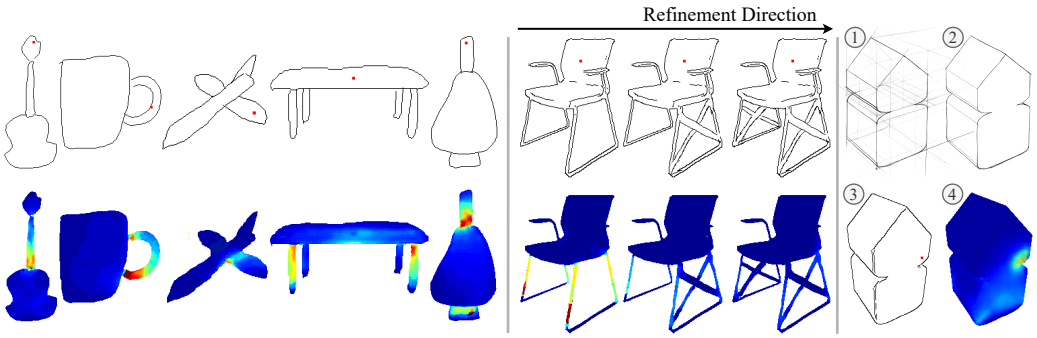
Training our beautification framework requires a considerably large amount of part-annotated sketch data. Considering the limited training sketches,

we collect a novel training dataset of part-labeled sketches by rendering the edge maps with the semantic labels under the best view [29] from the existing 3D shape repositories, i.e., PartNet [80], SDM-Net [34], and COSEG dataset [103]. We evaluate the beautification results in terms of faithfulness to the input sketch and the beautification quality. Intuitively, faithfulness can be evaluated quantitatively and via a user study, while beautification quality can only be evaluated via a user study due to its subjectiveness. In our sketch beautification approach, we perform a quantitative evaluation of faithfulness and a perceptive study on beautification quality to interpret the performance of sketch beautification methods more comprehensively.

1.2 Sketch-based Shape Structural Analysis

The design and fabrication process typically begins with sketching on paper, followed by digitization, and eventually, using fabrication machineries such as a waterjet, laser cutter, or 3D printer [52], as shown in Figure 1.1. Conventional structural analysis is used in both the digitization and manufacturing stages in a trial-and-error manner. This is a costly process, in terms of time, labor, and materials. To facilitate product design and digital fabrication, numerous structural analysis techniques [113, 155, 133, 91, 126, 87, 71] have been proposed to simulate the physical environment and directly analyze or optimize digital prototype structures virtually at the digitization stage. The goals of these techniques can be generally categorized into several aspects: weakness analysis [155, 87, 58], structural enhancement [113, 77], inner or surface material optimization [126, 91, 71, 151, 28, 31], and specified properties [144, 91, 100]. While these structural analysis tools enrich product design and fabrication, they are less accessible to designers at the early sketching stage since the effect of external physical factors on an object being designed is unknown to users during sketching.

In our *Sketch2Stress* work, we study the structural analysis of a sketched object and use the resulting analysis to generate the stress effect of the



(a) Structural Analysis on Freehand Sketches (b) Sketch-based Structure Refinement (c) Structural Analysis on a Product Sketch

Fig. 1.6. Our *Sketch2Stress* system supports users to easily perform structural analysis on their freely sketched objects by assigning forces at desired locations (shown in red dots) (a) and structural refinement (in each example, the upper row shows the progressively refined sketches while the bottom row shows our computed stress maps) on the weak regions of problematic sketched objects with real-time feedback of a stress map along with their editing operations (b). We also show that our system can handle professional product sketches, e.g., those in the OpenSketch dataset (c). In (c), we illustrate two examples of using professional product sketches for structural analysis, starting from the concept sketches, then the presentation sketches, the clean sketches, and finally, our generated structural stress maps.

object under external forces at specified locations, as displayed in Figure 1.6. Addressing this problem could enable designers to notice the potential structural weakness, specify their design space under different force configurations, and further refine the object at the sketching stage. Furthermore, this will open up possibilities for promoting sketch-based design and diagnosis to non-experts since sketching is an intuitive and universal tool for creativity and expression for novice users.

Since the existing digital structural analysis methods are mainly performed on 3D prototypes, a straightforward strategy to solve our sketch-based structural analysis problem might be to first use sketch-based shape reconstruction methods [72, 63, 24, 108, 152, 39, 131], followed by a 3D structural analysis method. However, existing sketch-based shape reconstruction approaches suffer several common limitations. First, they require specified multi-view sketches of the same object as input [72, 63], but their creation is highly demanding for users. Second, when taking a single-view

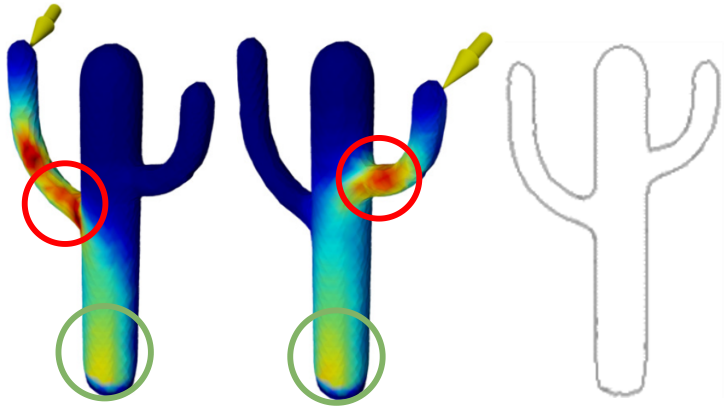


Fig. 1.7. Observations when using Ulu's 3D structural analysis algorithm [126]. Red circles show that similar structures produce similar stress distributions. And green circles display that neighboring regions have similar stress. Also, a sketch only is able to curve the geometry and structure of a 3D shape to some extent.

sketch as input, they often demand additional conditions such as camera parameters [152, 39] or 3D deformable templates [108, 131], making it difficult to reconstruct shapes with complex structures from one sketch only. Since 3D shape reconstruction is a difficult task, we are interested in directly performing structural analysis based on only an input sketch and the external force conditions.

While performing a structural analysis method [126] on 3D shapes, we observed that: (i) shapes with similar structures have similar stress distributions under the same external force with the same location, direction, and magnitude (see the regions with red circles in Figure 1.7); and (ii) on the same shape, neighboring points in local regions undertake similar stress under an external force (see the regions with green circles in Figure 1.7). This makes it possible to use a data-driven strategy to solve our problem. Therefore, we further transform the problem of sketch-based structural analysis into an image-to-image translation problem [49, 132], where we leverage a neural network to learn the mapping from input sketches to structural analysis results conditioned on external forces.

Since there is no existing dataset for sketch-based structural analysis, we construct a novel sketch-to-stress dataset by first defining rules to normalize and uniform force regions on 3D shapes in the same category based on Observation (i). Based on Observation (ii), we then uniformly sample

200 $\sim$ 300 key force locations on the surface of 3D shapes in each view to analyze their structural soundness rather than exhaustively sampling all the surface points, and apply 3D structural analysis to the shapes, where the external forces are set with equivalent magnitude [126], and finally render multi-view sketches and the corresponding view-dependent stress maps from the 3D structural analysis results. In this way, we collect a large-scale dataset consisting of quadruples of an input sketch, a point map indicating the force location, a normal map recording force directions of all possible forces, and a corresponding structural stress map. Note that inheriting the assumption of [126], we set the *magnitude* of external forces in our problem to be all the same and set the force *directions* to be the same as the surface normals (pointing inward) at the force locations. Also, since Ulu et al. [126] rely on the boundary shell to represent the shape structures of 3D models and further approximate the relationship between input forces and resulting stresses on this representation, the same boundary shell representation is inherited implicitly in our assumption for user-designed objects. Therefore, sketched objects corresponding to commonly seen real-world objects might exhibit severely fragile regions (see the "problematic structure" in Figure 4.11) since the inner material properties and inter-part connection manners are not considered in the boundary shell setting.

To synthesize a structural stress map from an input sketch conditioned on an external force with arbitrary *location* and uncertain *direction*, we present a novel framework combining a one-encoder-two-branch-decoder generator with two discriminators: one branch in the generator is used to synthesize a corresponding structural stress map from the input sketch and the force location; the other branch aims to infer the direction (opposite-normal) of the external force. These two branches jointly guarantee that the generator can perceive the distinctive locations and directions of external forces imposed on sketches. Two discriminators supervise the learning process of the two branches of the generator.

With our trained network (for *Sketch2Stress*), users can easily check the structural soundness of a sketched object under a single force assigned at any location. In addition, a well-known physical axiom states that: "If two forces act on an object in the same direction, the net force is equal to the sum of the two forces". Based on this axiom, we present an efficient

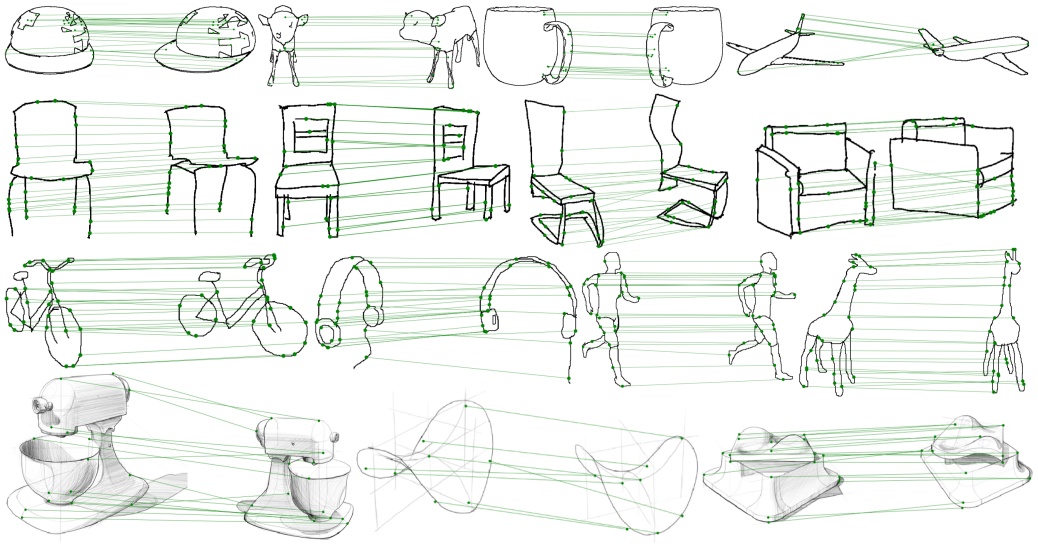


Fig. 1.8. Multi-view sketch correspondence results by *SketchDesc* on line drawings synthesized from 3D shapes (top) and freehand sketches (middle and bottom).

region-wise sketch-based structural analysis method to approximate the stress effect of the net force at a local region by aggregating the stress maps of multiple forces at different locations but in the same normal direction.

1.3 Correspondence Learning for Multi-View Sketches

Although it is not challenging for human viewers to interpret missing 3D information from single-view sketches, multi-view inputs are often needed for computer algorithms to recover the underlying 3D geometry due to the inherent ambiguity in single-view sketches. A key problem for interpreting multi-view sketches of the same object is establishing semantic correspondence among them. This problem has been mainly studied for 3D geometry reconstruction from careful engineering drawings in orthographic views [37]. The problem of establishing semantic correspondence from rough sketches in arbitrary views (e.g., those in Figure 1.8 (middle and bottom)) is challenging due to the abstraction and distortions in both shape and view,

and is largely unexplored. Addressing this problem can benefit various applications, for example, to design an interactive interface for users with little training in drawing to create 3D shapes using multi-view sketches.

In our *SketchDesc*, we take the first step to learning to establish the semantic correspondence between freehand sketches depicting the same object from different views. This requires a proper shape descriptor. However, traditional descriptors like Shape Context [7] and recent learning-based patch descriptors [122, 78, 123] are often designed to be invariant to 2D transformations (e.g. rotation, translation, and limited distortion) and cannot handle large view changes (e.g. with view disparity greater than 30 degrees). Such descriptors, especially used for the applications of stereo matching [43, 124, 18, 73, 142] and image-based 3D modeling [125, 69], heavily exploit the features containing textures and shadings of images for inferring similarities among different points or image patches, and thus are not directly applicable to our problem with the input of multi-view sketches. This is because sketches only contain binary lines and points, exhibiting inherent sparsity and ambiguity.

We observe that human viewers can easily identify corresponding points from sketches with very different views. This is largely because human viewers have knowledge of sketched objects from different views. Our idea is thus to adapt deep neural networks previously used for learning patch-based descriptors to learn descriptors for corresponding patches from multi-view sketches.

Training deep neural networks require a large-scale dataset of sketch images with ground truth semantic correspondence. Unfortunately, such training datasets are not available. On the other hand, manually collecting multi-view hand-drawn sketches and labeling the ground-truth correspondence would be a demanding task. Following previous deep learning solutions for 3D interpretation of sketches [83, 24, 63], we synthesize the multi-view line drawings from 3D shapes (e.g. from ShapeNet [146]) by using non-photorealistic rendering. Given the synthesized dataset of multi-view line drawings as sketches, we project the 3D vertices of 3D models to multi-view sketches to get ground-truth correspondences (Figure 5.2). This data generation pipeline is to emphasize the correspondence from 3D

shapes and force deep networks to learn the valuable 3D correspondences among 2D multi-view sketches.

We formulate the correspondence learning problem into a metric function learning procedure and build upon the latest techniques for metric learning and descriptor learning [122]. To find an effective feature descriptor for multi-view sketches, we combine a patch-based representation and a multi-scale strategy (Figure 5.1) to address the abstraction problem of sketches (akin to Sketch-A-Net [148]).

We further design a multi-branch network (Figure 5.4) with shared weights to process the patches at different scales. The patch-based representation helps the network specify the local features of a point on a sketch image by embedding the information of its neighboring pixels. Our multi-scale strategy feeds the network with local and global perspectives to learn distinctive information at different scales.

The multi-scale patch representation allows the use of ground-truth correspondences that are away from sketch lines as the additional training data. This not only improves the correspondence accuracy but also enables correspondences inside the regions of sketched objects (e.g., the cup in Figure 1.8), potentially benefiting applications like sketch-based 3D shape synthesis. We evaluate our *SketchDesc* by performing multi-view sketch correspondence and pixel-wise retrieval tasks on a large-scale dataset of synthesized multi-view sketches based on three shape repositories: ShapeNet, Princeton Segmentation Benchmark (PSB) [15] and Structure Recovery [101] to show the effectiveness of our proposed framework. We also test our trained *SketchDesc* on freehand sketches (Figure 1.8) drawn by volunteers and collected from the OpenSketch dataset [38], which shows its robustness against shape and view distortions.

2 Related Work

In this chapter, we review the literature that is closely related to our works: Sketch Beautification (Section 2.1), Sketch Representations (Section 2.2), Implicit Representations (Section 2.3), 3D Shape Structural Analysis (Section 2.4), Sketch-based Shape Reconstruction (Section 2.5), Image-to-Image Translation (Section 2.6), Correspondence Establishment for Images (Section 2.7), and Deep Learning in Sketch Analysis (Section 2.8). The Sections 2.1, 2.2, and 2.3 mainly discuss the related works to sketch beautification problem. While Sections 2.4, 2.5, and 2.6 focus on discussing the existing techniques for sketch-based structural analysis task. Finally, Sections 2.7 and 2.8 analyze the previous works related to the problem of inferring semantic correspondence among multi-view sketches.

2.1 Sketch Beautification

In the past decades, numerous algorithms were proposed to beautify free-hand drawings. The earliest algorithmic beautification method with aesthetic constraints dates back to SketchPad [120]. Later, Pavlidis and Van Wyk [90] proposed an algorithm for beautifying figures as a post-process. Igarashi et al. [48] presented a system for rapid geometric design that beautified strokes in an interactive way. Recently, great strides have been made in the sketch simplification [127, 68], rough sketch cleanup [105] and sketch vectorization [86]. Since these methods do not attempt to change or beautify the global structure of input sketches, we do not conduct a detailed review here. We refer interested readers to [141] for an insightful survey.

The more related works to our sketch beautification approach are iCanDraw [26], ShadowDraw [60], and ShipShape [32]. iCanDraw and Shad-

owDraw can output a sketch with a reasonable layout and more realistic details by providing reference scaffolds or shadow layouts for users. However, as discussed in Subsection 1.1, limited by their heuristic and iterative settings during drawing, they cannot beautify the existing sketches. For ShipShape, this method only performed a curve-level beautification in an interactive way by enforcing some geometric constraints to the afterward curves based on the previous strokes, such as symmetry, reflection, and arc fitting. These constraints are difficult to apply to the existing sketches in practice without knowing the drawing order of the component strokes. Besides, during its beautification process, extra efforts from users on the structure adjustment are still needed. Instead, we study the part-level beautification (Subsection 3.1.2) since the part labels are more accessible with existing segment approaches [46, 65, 143] or user-specified annotations. We further reduce user labor with our sketch assembly model by performing structure beautification automatically (Subsection 3.1.3).

A similar beautification problem was also explored in EZ-Sketching, where Su et al. [117] proposed a three-level optimization framework to beautify a sketch traced over an image. Their beautification process heavily relies on the image being traced, which is missing in our sketch beautification task. In DeepFaceDrawing, Chen et al. [14] presented a robust portrait image synthesis method, which can handle rough or incomplete sketches as input. By decomposing a face sketch into face components and projecting the face component sketches to the component-level sketch manifolds, the input freehand sketches can be implicitly refined for generating photo-realistic face images. As the layouts or structures of human portraits are almost fixed, their approach is not suitable for our beautification task. To handle the diverse structure of sketched man-made objects, we propose a novel sketch assembly module (Subsection 3.1.3), as inspired by [98].

2.2 Sketch Representations

A sketch is generally represented as either a rasterized binary-pixel map [97] or vector sequences [41], or both [8]. Since the sketch datasets TU-berlin [30] and Quickdraw [41] were introduced, the research commu-

nity has seen many works in sketch understanding. For example, Yu et al. [149] proposed a multi-scale and multi-channel CNN framework with a larger kernel size for sparse raster sketches. Ha and Erc [41] introduced a sequence-to-sequence variational auto-encoder with the bidirectional RNN backbone for vectorized sketches. Later, the CNN and RNN representations [140, 67] were combined to better represent sketches in their retrieval and recognition tasks, respectively. Most of the subsequent works [8, 9] basically inherited and employed the off-the-shelf backbones for the specified tasks. However, since the CNN-based representations are designed to learn the distribution of all the pixels in the 2D image space rather than the solo valid points on strokes, they tend to generate broken segments with shadow effects (Figure 1.5 (Top)) during the interpolation process. The RNN-based representations failed to reconstruct input sketches with vector sequences precisely (Figure 1.5 (Bottom)). Therefore, we turn to implicit functions to better represent part sketches in our sketch beautification approach.

2.3 Implicit Representations

Recently, implicit functions have attracted extensive attention in the research community [25, 16, 94, 88, 76, 17], since they can represent 3D shapes in a continuous and smooth implicit field. Existing implicit representations typically used a spatial function to represent a shape by mapping the inside and outside points distinguishably. In DeepSDF, Park et al. [88] utilized a signed distance function (SDF) to represent a watertight shape in 3D space where inside and outside points are respectively mapped to negative and positive values, and the underlying surface is implicitly represented by a zero-crossing surface. Similarly, Mescheder et al. [76] represented a target shape with a continuous occupancy function, indicating the probability of points being occupied by a shape. However, different from watertight 3D shapes, sketches are often created with no concept of *inside* or *outside* and are usually too sparse with limited valid points or pixels in the 2D image space. Hence, the aforementioned implicit representations cannot be directly applied to part sketches segmented from an input freehand sketch To address this issue, we specially design an intuitive 2D implicit function (Subsection 3.1.2) for part sketches, where we map all

sampled points of a canonical 2D space to two exclusive statuses: whether its projection to the image space of the part sketch *hits* the stroke or not (see Figure 3.3).

2.4 3D Shape Structural Analysis

Various recent works support computational analysis on the structural soundness of 3D shapes. Especially with the emergence of 3D printing techniques, numerous approaches were proposed for printed objects in a wide range of tasks, from structural weakness detection [126, 113, 155, 87] to material optimization [91, 71, 151, 28, 31]. Since it is challenging to convey the variations of materials from one sketch, we do not review the material-oriented approaches.

The first structural analysis work for 3D printed objects dates back to [121], where Telea and Jalba identify thin and thick parts and estimate whether a thin part could support its attached parts under several pre-defined geometric rules. Later, Stava et al. [113] use FEM (finite element method [47]) to discover and strengthen problematic components of a printed model under the applied gravity load and 2-finger gripping loads. Then Zhou et al. [155] propose an analysis technique to predict fragile regions under worst-case external force loads by identifying potential regions of a structure that might fail under arbitrary force configurations. Later, Langlois et al. [58] present a stochastic FEM for predicting failure probabilities under the force loads at contact regions. Different from previous works with specified force load settings, Ulu et al. [126] propose a more general structural optimization approach that examines 3D shapes with any force loads at arbitrary locations and computes a feasible material distribution to withstand such forces.

Different from 3D prototypes, sketches are usually created in the 2D space with sparse content. This makes it difficult to apply finite element analysis, which is the basic technique for most 3D structural analysis approaches. To model the relationship between the input forces and the corresponding structural stress of the sketched objects, in our *Sketch2Stress*, we translate

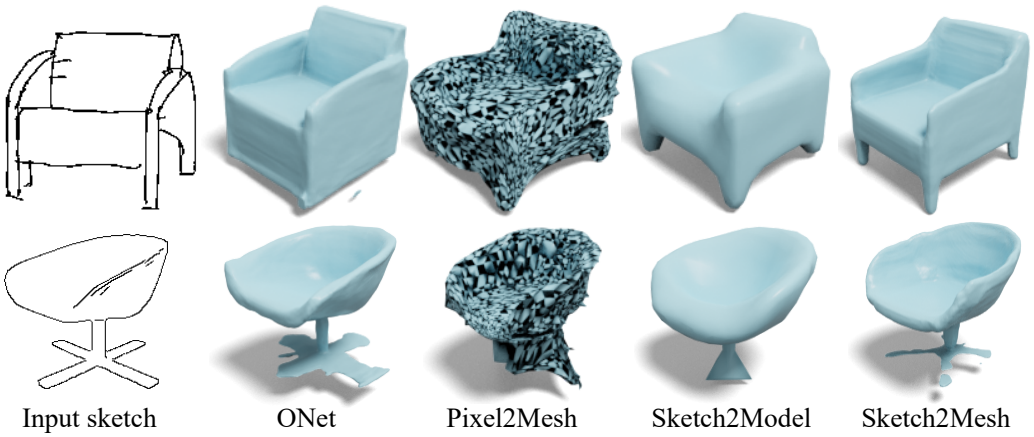


Fig. 2.1. Single-view sketch-based shape reconstruction methods. We can see ONet [76], Pixel2Mesh, Sketch2Model, and Sketch2Mesh fail to reconstruct the geometry details of the input sketches. ONet and Sketch2Mesh tend to generate detached noises, broken parts, and the inconsistent orientation of chair legs at the bottom row. While Pixel2Mesh and Sketch2Model generate too coarse shape results where the former's surface patches are widely corrupted, and the latter's local details are heavily over-smoothed.

the sketch-based structure analysis problem to the data-driven image-to-image translation task where we learn the mapping between the input sketches and the structural stress responses conditioned on the variable external forces from massive sketch-to-stress data pairs.

2.5 Sketch-based Shape Reconstruction

Using an additional step to convert input sketches to intermediate 3D shapes with sketch-based shape reconstruction approaches usually requires extra conditions such as multi-view inputs, camera parameters, and 3D category templates, which are lacking in our scenario. Figure 2.1 further displays the limitations of state-of-the-art single-view sketch-based shape reconstruction methods. The 3D meshes generated by ONet [76] and Pixel2Mesh [131] have obvious artifacts, like detached parts and inverse patches, which prevent performing FEM requiring continuous and closed input 3D surfaces. The meshes generated by Sketch2Model [152] lose too many geometry details. For Sketch2Mesh [39], its bottom reconstructed mesh

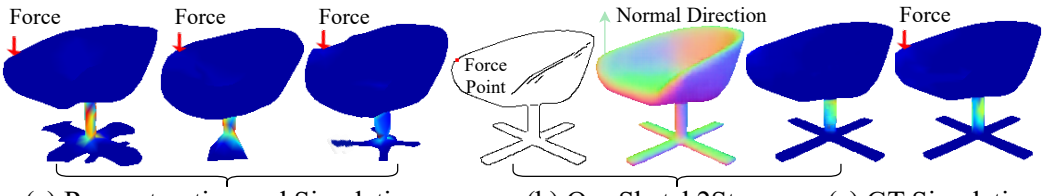


Fig. 2.2. Comparison of the reconstruction-and-simulation way (a) and our *Sketch2Stress* (b). The models in (a) are the reconstructed meshes in Fig. 2.1 (Bottom). The red arrows indicate the applied external forces. In (b), the force is plotted on the input sketch, and the generated normal map and stress map are side-placed. The ground-truth 3D stress simulation is given in (c). Please zoom in to examine the details of the above stress distributions.

has not only broken parts but also inconsistent legs compared with the input sketch. Hence, none of these approaches could perform shape reconstruction from sketches robustly. Therefore, we turn to image-to-image translation techniques to directly generate a feasible 2D structural analysis result for an input sketch (Section 4.1). To demonstrate the faithfulness and effectiveness of our sketch-based structure analysis approach, we provide a direct comparison between our *Sketch2Stress* approach and the reconstruction-and-simulation approach applying stress simulation [126] on generated meshes (Figure 2.1) by ONet/Sketch2Model/Sketch2Mesh after post-cleanup. Compared with the reconstruction-and-simulation results, as shown in Figure 2.2, our *Sketch2Stress* can reconstruct a view-dependent structure robustly and is more competent for the sketch-based structure analysis task than the reconstruction-and-simulation way.

2.6 Image-to-Image Translation

Since Isola et al. [49] and Wang et al. [132] introduced the general-purpose cGAN frameworks for diverse types of inputs, e.g., realistic images, sketches, and semantic masks, there are many sketch-based image synthesis tasks using image-to-image translation techniques. The most related to our *Sketch2Stress* are 3D-aware approaches [118, 51] with sketch inputs. For instance, Su et al. [118] present an interactive system for high-quality normal map generation. Later, Jiao et al. [51] propose a

joint framework that leverages category and depth information to improve shape understanding for tactile sketch saliency prediction. However, the aforementioned methods cannot be directly applied to our sketch-based structural analysis problem since they have no proper way to represent the external forces with their designed frameworks.

Recently, diffusion models [110, 44, 111] have been used to obtain state-of-the-art results in text-to-image synthesis and text-guided image synthesis [75, 82, 92]. In spite of the impressive and realistic generation performance of the aforementioned methods, it is still challenging for diffusion models to impose precise spatial control on the generation outputs due to the nature of the one-to-one mapping between the noise vector and the corresponding ground truth data samples. This limitation makes diffusion models unsuitable for our problem of sketch-based structural analysis under arbitrary force conditions, which requires precise and faithful control over the force locations. Additionally, diffusion models generate images from noise vectors through several iterative intermediate steps during the inference (denoising) stage, consuming more time than those deep networks. Therefore, existing diffusion models might not be suitable for our real-time editing scenario where the system is supposed to output the instant structural stress effects once it receives the user-specified external forces.

2.7 Correspondence Establishment for Images

2.7.1 Image-based Modeling and Stereo Matching

Image-based modeling often takes as input multiple images of an object [69, 1] or scene [125, 109] from different views, and aims to reconstruct the underlying 3D geometry. A typical approach to this problem is to first detect a sparse set of key points, then adopt a feature descriptor (e.g., SIFT [70]) to describe the patches centered at the key points, and finally conduct feature matching to build the correspondence among multi-view images.

Stereo matching [43, 124] takes two images from different but often close viewpoints and aims to establish dense pixel-level correspondence across images. Our problem of inferring semantic correspondence for multi-view sketches is different from these tasks in the following ways. First, the view disparity in our input sketches is often much larger. Second, unlike natural photos, which have rich textures, our sketches have more limited information due to their line-based representation.

2.7.2 Local Image Descriptors

Local image descriptors are typically derived from image patches centered at points of interest and designed to be invariant to certain factors, such as rotation, scale, or intensity, for robustness. Existing local image descriptors can be broadly categorized as hand-crafted descriptors and learning-based descriptors. A full review is beyond the scope of this dissertation. We refer interested readers to [20] for an insightful survey.

Classical descriptors include, to name a few, SIFT [70], SURF [6], Shape Context [7], and HOG [22]. The conventional local descriptors are mostly built upon low-level image properties and constructed using hand-crafted rules. Recently, learning-based local descriptors produced by deep convolutional neural networks (CNNs) [122, 78, 123] have shown their superior performance over the hand-crafted descriptors, owing importantly to the availability of large-scale image correspondence datasets [136, 3] obtained from 3D reconstructions. To learn robust 2D local descriptors, extensive research has been dedicated to the development of CNN designs [42, 150, 122, 154, 135], loss functions [57, 85, 5, 78, 54, 79] and training strategies [104, 19, 74]. The above methods, however, are not specially designed for learning multi-view sketch correspondence. The work of GeoDesec [74] shares the closest spirit to our *SketchDesc* and employs geometry constraints from 3D reconstruction by Structure-from-Motion (SfM) to refine the training data. However, SfM heavily depends on the textures and shadings in the image domain and is unsuitable for sketches.

2.7.3 Multi-scale Strategies for Descriptors

Shilane and Funkhouser [102] computed a 128-D descriptor at four scales in spherical regions to describe distinctive regions on 3D shape surfaces. Recently, Huang et al. [45] proposed to learn 3D point descriptors from multi-view projections with progressively zoomed-out viewpoints. Inspired by these methods, we design a multi-scale strategy to gather local and global context to locate corresponding points in sketches across views (Figure 5.1). Different from [45], which focuses on rendered patches of 3D shapes, our *SketchDesc* considers patches of sparse line drawings with limited textures as input. To reduce the network size and computation, we adopt a smaller input scale of 32×32 [122] rather than 224×224 in [45]. In addition, Huang et al. [45] used three viewpoints for 3D point descriptors, while our *SketchDesc* extracts descriptors of points in sketches drawn under a specific viewpoint for correspondence establishment across a larger range of views.

2.8 Deep Learning in Sketch Analysis

With the recent advances in deep learning techniques, a variety of deep learning-based methods have been proposed for sketch analysis tasks such as sketch synthesis [40], face sketch-photo synthesis [130, 156, 157], sketch recognition [148], sketch segmentation [66], and sketch retrieval [95, 140]. Among these works, [156] and [157] are the most relevant to our *SketchDesc*, and they aim to model the correspondence between face photos and face sketches. However, different from our *SketchDesc* to exploit the pixel-level correspondence between sketches, they explore the image-level mapping between face photos and sketches. In general, none of these existing solutions can be directly applied to our task of inferring the pixel-level correspondence for multi-view sketches. To make deep learning possible in sketch analysis, there exist multiple large-scale datasets of sketches including the TU-Berlin [30], QuickDraw [40], and Sketchy [95] datasets. However, they do not contain multi-view sketches and thus cannot be used to train our *SketchDesc* network. A sketch is generally represented as either a rasterized binary-pixel image [95] or vector sequences [40, 66] or both [140].

Since it is difficult to render line drawings from 3D shapes as a sequence of well-defined strokes, we use rasterized binary-pixel images to represent our training data (for *SketchDesc*) and input sketches.

2.8.1 Multi-view Sketch Analysis

Multi-view sketches are often used in sketch-based 3D modeling [93, 83, 72, 24, 63]. Early sketch-based modeling methods (e.g., [93]) require precise engineering drawings as input or are limited by demanding mental efforts, requiring users first to decompose a desired 3D shape into parts and then construct each part through careful engineering drawings [37]. To alleviate this issue, several recent methods [83, 72, 24, 63] leverage learning-based frameworks (e.g., GAN [36]) to obtain the priors from training data and then infer 3D shapes from novel input multi-view sketches (usually in orthographic views, namely, the front, back, and side views). However, these methods often process individual multi-view sketches in separate branches and do not explicitly consider the semantic correspondence between input sketches. Recently, a richly-annotated dataset of product design sketches, named OpenSketch, was presented by [38], which contains around 400 sketches of 12 man-made objects. However, the limited data in OpenSketch is insufficient for training deep neural networks. In our *SketchDesc*, we synthesized 6,852 sketches for 18 shape categories to learn local sketch descriptors.

SketchZooms by Navarro et al. [81] is a concurrent work to our *SketchDesc* and studies the sketch correspondence problem with a similar deep learning-based solution. Our *SketchDesc* is different from SketchZooms as follows. To generate training data for cross-object correspondence, the used 3D shapes need to be semantically registered together in advance in SketchZooms, while our *SketchDesc* is more fine-grained as it explores training data generation from individual 3D shapes for cross-view correspondence. SketchZooms follows [45, 115] and adapts AlexNet [56] (40M parameters) with the final layer replaced with a view pooling layer. In contrast, our designed framework for *SketchDesc* has a smaller size (1.4M parameters), and its high performance shown in our experiments paves the way for our *SketchDesc* to be more easily integrated into mobile or touch devices

where people can create and doodle their sketch drawings more easily and conveniently.

3

Learning Part Beautification and Structural Refinement for Imperfect Sketches of Man-made Objects

In this chapter, we first introduce the methodology of our proposed sketch beautification system in Section 3.1. We then elaborate the experiment details of our sketch beautification system in Section 3.2. Finally, we summarize this chapter and discuss the limitations in Section 3.3.

3.1 Methodology

In this chapter, we mainly focus on freehand sketches created for man-made objects that can be segmented partly. We treat this kind of sketches as a combination of individual part sketches, since 3D man-made objects often consist of semantically meaningful parts [64, 129, 137]. Our framework for sketch beautification is illustrated in Figure 3.1. We first parse an input freehand sketch depicting a man-made object like a chair to part-level sketches, and then beautify the individual part sketches in the part beautification module. In this module, we construct an implicit manifold for each part. For each part, we can search and absorb the geometry features from its key neighbors in this manifold for geometry beautification. Finally, the beautified part sketches are recomposed as the final beautified sketch

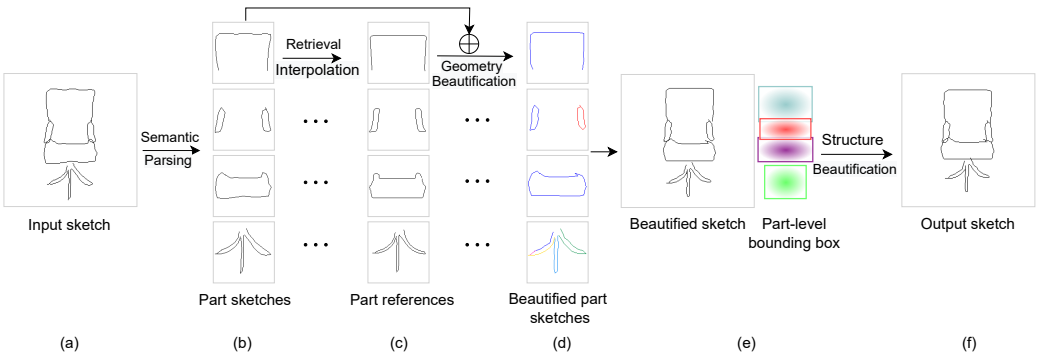


Fig. 3.1. System pipeline of sketch beautification. For an input sketch (a), we first parse it to individual part sketches (b), and then synthesize the corresponding part references (c) by retrieval and interpolation. After performing geometry beautification on the part sketches (b) towards part references (c), we obtain the new part sketches (d) with well-beautified geometry (see geometry differences between (b) and (d)). During the stage of structure beautification, we adjust the imperfect structure (notice the misalignment of chair arms) of the intermediate output (e) with the help of part-level bounding boxes (e) and generate the final beautified sketch (f). Different colors in beautified part sketches (d) indicate the different strokes. The colorful bounding boxes (e) denote the scales and spatial locations of different part sketches in the image space (256 $\times$ 256).

by our sketch assembly model, which essentially performs the structure beautification task.

3.1.1 Sketching Interface

Our algorithm requires semantic segmentation of an input sketch. Sketch segmentation can be done automatically, as demonstrated in previous works [46, 65, 143]. However, for simplicity, we design a simple interface (Figure 3.2) for users to interactively provide part-level semantic information while sketching. The key enablers of this system are the part beautification component and the structure beautification component. Once activated on the interface (through “Part Beautification" and “Structure Beautification” buttons), the two functions will run in the background and beautify the user’s input automatically after the user finishes one complete sketch. We provide more details of the mechanism of these two functions in Sections 3.1.2 and 3.1.3.

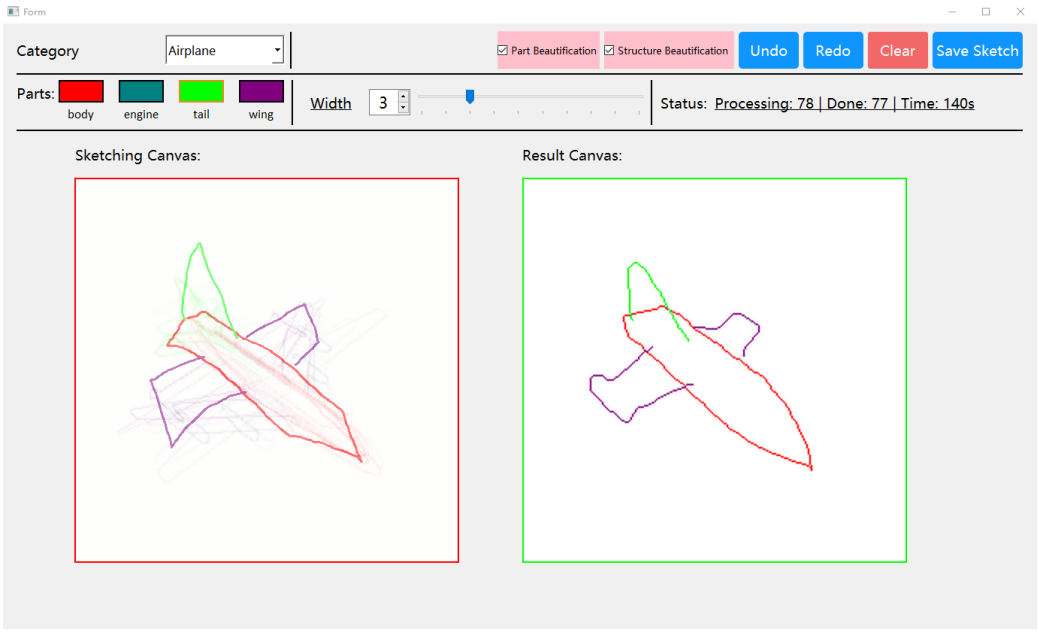


Fig. 3.2. Our sketching interface.

Before sketching, a user first needs to select a target object category from one of our prepared nine classes (e.g., airplane, chair, table). Our interface then shows the corresponding set of part labels. It also provides part-level shadows to provide initial geometry and structure guidance, similar to ShadowDraw [60]. Unlike ShadowDraw, we do not update the underlying shadows dynamically during the drawing process since we hope to give more freedom to users and not influence them too much. In this interface, we provide several basic drawing tools for users to amend their drawings, such as clear, undo, redo, etc. By clicking on a part label (e.g., airplane engine), the strokes drawn afterwards will be labeled with the selected label automatically. In this way, we obtain a freehand sketch with well-segmented parts. Note that our system saves each drawn sketch in both raster and vector graphics formats. For a better drawing experience, our interface supports the drawing of strokes with varying thickness. However, such strokes are pre-processed by extracting the skeletons [153] from the user's sketch input to have one-pixel width for further processing.

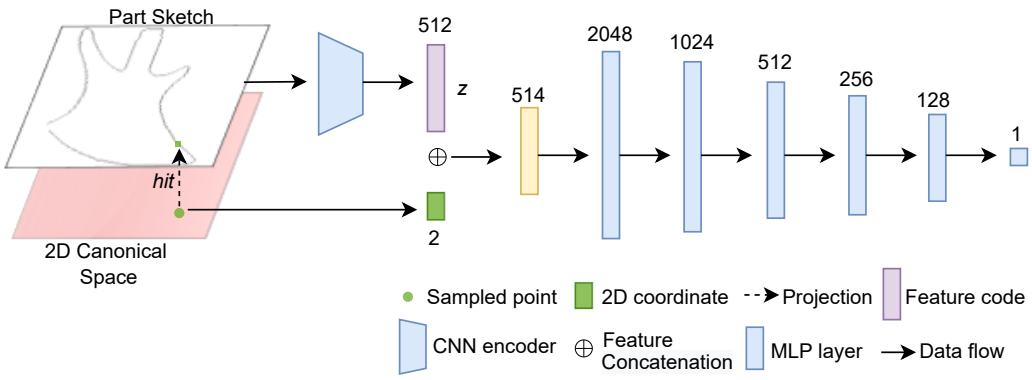


Fig. 3.3. Pipeline for learning a sketch implicit representation.

3.1.2 Part Beautification

Given a rasterized freehand sketch with part annotations, we treat its component sketches separately and beautify them individually in the corresponding part-level implicit manifolds. Different from CNN representations created for the dense and high-frequency RGB pixels, our novel implicit representation works for the low-frequency and sparse sketched points in the 2D image space. Existing implicit representations are not suitable for our sketch points in the 2D space, since a 2D sketch is a kind of more discrete representation having no inside and outside spaces and occupying no continuous regions compared with the closed 3D shapes. We thus design our own sketch implicit representation where we first sample points from a 2D canonical space. We then project these points to the same-size image space of sketches and record whether these points could hit any stroke of a sketch. Finally, a discrete sketch can be represented by the sampled points (hitting sketches) of this continuous 2D canonical field implicitly (see Figure 3.3).

Sketch Implicit Representation

An implicit field is typically defined by a continuous function over 2D/3D space. In this work, we take a sketch as an implicit function (hit function)

defined over the continuous coordinates P in a 2D canonical space, as given in Equation 3.1.

$$H(p) = \begin{cases} 1, & p\text{'s projection hits a stroke,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Here, we define the valid points those projections hit a stroke in a sketch as 1 and the invalid points those projections hit the empty background as 0. The underlying sketch is represented by the points with positive values, that is $H(\cdot) = 1$, similar to the zero-isosurface points in 3D implicit representations. We then employ a neural network f_θ to approximate the implicit function $H(p)$ as $f_\theta(p)$. Following [17], a CNN encoder is further added to extract the feature code z from the input sketch. Conditioned on the feature code, the implicit function can be further formulated as $f_\theta(z, p)$.

Here, we adopt Multi-layer Perceptrons (MLPs) with rectified linear unit (ReLU) as the implicit function f_θ. To represent part sketches in a uniform 2D space, we center-crop and resize all the part sketches to a 128×128 scale. We illustrate the model for learning our sketch implicit representation in Figure 3.3. The part sample is first encoded to a 512D feature code and then a sampled 2D coordinate is concatenated with the feature code as the input to the implicit model. After the model converges, an input (part) sketch can be represented as the implicit representation z and interpreted back to the 2D image space by sampling points to f_θ.

Loss function

We use a mean squared error between ground truth labels and predicted labels for each point as the loss function for training sketch implicit function f_θ. Specifically, we formulate it as follows:

$$L(\theta) = \sum\nolimits_{p \in P} \| f_\theta(p) - H(p) \|^2, \tag{3.2}$$

where P refers to the point set sampled from the 128×128 2D space and $H(p)$ is the ground truth value in our dataset given the query point p.

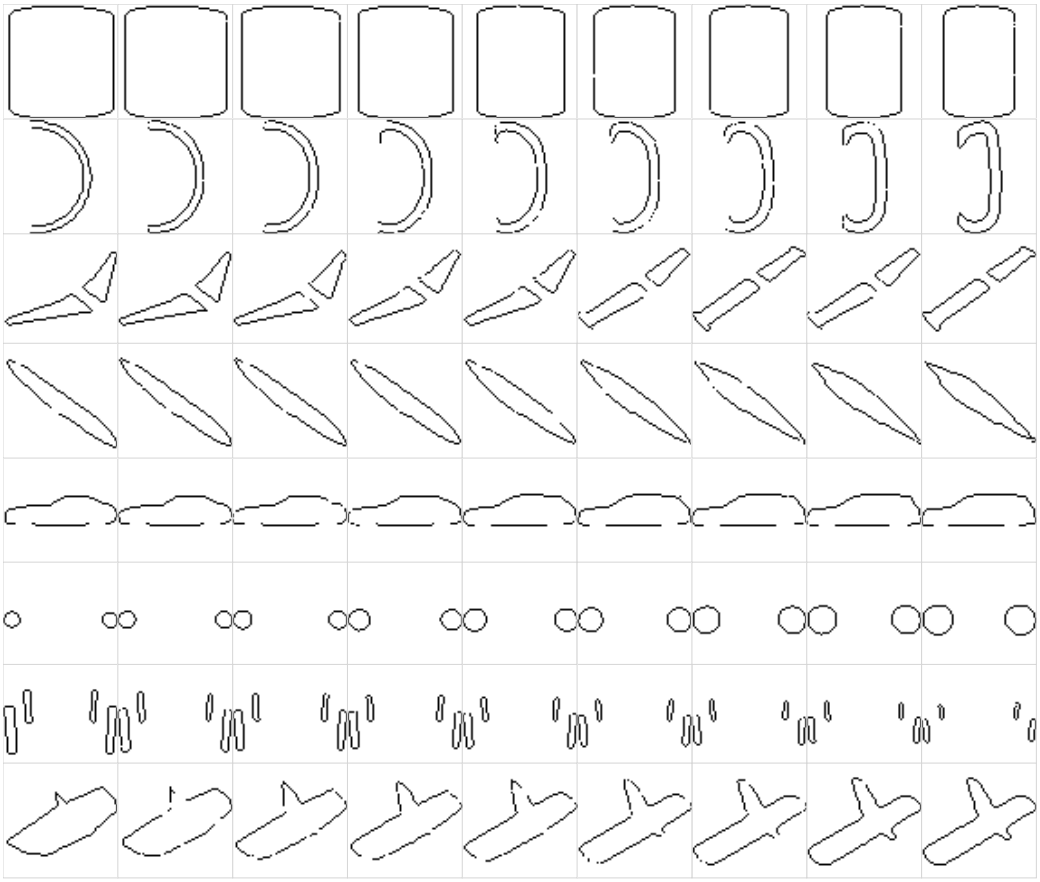


Fig. 3.4. Smooth interpolation of the leftmost and rightmost samples with our implicit representations.

Retrieval and Interpolation

With the part-level sketch implicit representations of our dataset $\{v_1, v_2, \ldots, v_N\}$, we can synthesize the novel samples continuously and smoothly by linear interpolation of implicit representations of existing sketches, as shown in Figure 3.4. To further bridge the gap between the user's conceptual freehand sketches and the existing part sketches in our dataset, we adopt the same retrieval-and-interpolation strategy as [14] to instantiate the user's conceptual sketches with on-hand samples in our dataset by local linear interpolation.

Specifically, we first take the implicit representation v_q of an input part sketch as the query to retrieve its top K neighbors in our dataset. In the following steps, we construct a part-level manifold with the retrieved K

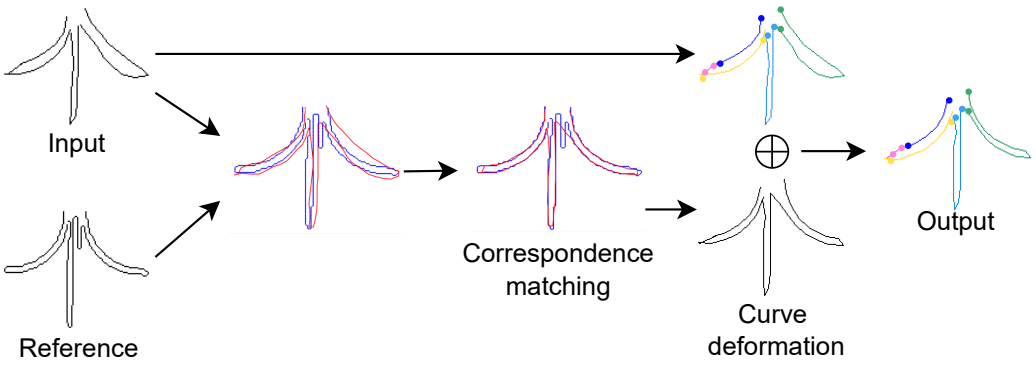


Fig. 3.5. Pipeline for part-level geometry beautification.

neighbors as the basis vectors $\{v_{t1}, v_{t2}, \ldots, v_{tK}\}$, and project the v_q to this manifold as Equation 3.3.

$$\min \|v_q - \sum_{i=1}^{K} w_i \cdot v_{ti}\|^2, \qquad s.t. \sum_{i=1}^{K} w_i = 1 \tag{3.3}$$

where we set $K = 3$ in our implementation and calculate the unknown weight parameter w_i for each basis vector v_{ti} by solving this constrained least-squares problem. We finally obtain the projected implicit representation v_p by lineally interpolating the basis vectors locally:

$$v_p = \sum_{i=1}^{K} w_i \cdot v_{ti}. \tag{3.4}$$

Obtaining the projected implicit representation v_p, we further interpret it back to the 2D image space as the beautified reference for the input conceptual sketch (see Figure 3.1 (c)).

Part-level Geometry Beautification

In this work, we consider the part-level beautification as a deformation process that deforms each part sketch towards the corresponding reference obtained from the retrieval-and-interp- olation step rather than directly replacing the input with the reference. In this way, we hope to preserve the users' original drawing intentions as much as possible. Thereby, given each input part sketch and its reference, we design the beautification pipeline as follows: correspondence matching and curve deformation. The former step performs a coarse shape-level registration that transforms all points of

the input sketch to approximate the general shape of the reference. Based on the output of correspondence matching, the latter step then conducts a fine-grained curve-level deformation that deforms the input sketch towards the above intermediate output meanwhile keeping the original endpoints of the input curves unchanged, as illustrated in Figure 3.5.

For correspondence matching between the input part sketch and the reference sketch, we adopt a classical non-rigid registration method [10]. Despite its robustness, this method fails to produce reasonable correspondence results efficiently when being applied to our scenario. We speed up this method from more than 3 seconds to around 0.3 seconds by adding two weight decay parameters α and β that decrease exponentially with the increase of the iteration T as shown in Equation 3.5. Here we refer to the sketched points of the input and the reference as X and Y, respectively.

$$\begin{aligned} E_c &= w_1 E_{match} + \alpha^T \cdot w_2 E_{rigid} + \beta^T \cdot w_3 E_{arap}, \\ E_{match} &= \sum_{i=1}^{n} \|z_i - P_y(z_i)\|_2^2, \\ E_{rigid} &= \sum_{i=1}^{n} \|z_i - R \cdot (x_i + t)\|_2^2, \\ E_{arap} &= \sum_{i=1}^{n} \sum_{j \in N_i} \|(z_j - z_i) - R \cdot (x_j - x_i)\|_2^2, \end{aligned} \tag{3.5}$$

where $P_y(\cdot)$, R, and t refer to the closest point in reference point set Y, the rotation matrix, and translate parameters, respectively. We initialize z_i with x_i in the input point set X, and set $w_1 = 1, w_2 = 100, w_3 = 0.9$, $\alpha = 0.4$, and $\beta = 0.9$. Note that we solve the overall objective function E_c in an iterative way. In our experiment, $T = 15$ iterations are sufficient to produce a satisfying result.

To keep the user's original intention as much as possible, after calculating the intermediate output of the first step, we further perform the curve deformation that deforms the recorded strokes S of the user's input towards the intermediate output following Equation 3.6. As we mentioned before, our interface can also record the strokes during the drawing process.

$$\begin{aligned} E_d &= E_{position} + E_{shape}, \\ E_{position} &= \sum_{s \in S} \|p_0 - X_0^s\|, \\ E_{shape} &= \sum_{s \in S} \sum_{i=1}^{n} \|\delta(p_i) - \delta(Y_i)\|. \end{aligned} \tag{3.6}$$

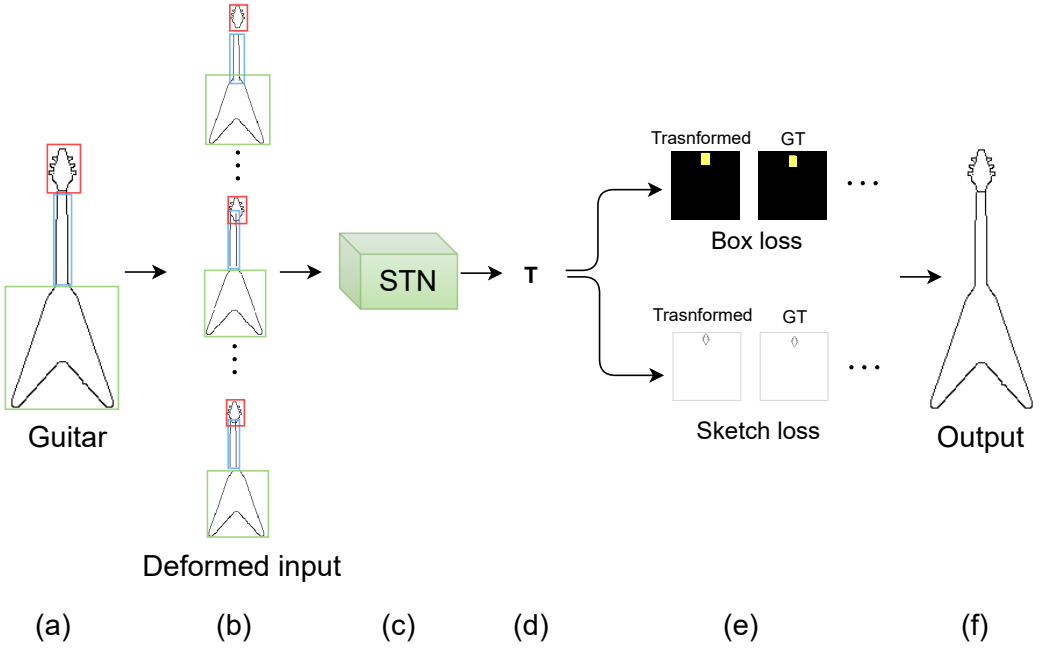


Fig. 3.6. Pipeline for learning structure beautification of the part-deformed sketch. (a) Ground truth. (b) Synthesized part-deformed sketches. (c) STN backbone of sketch assembly model. (d) Learned transformation matrix. (e) Losses for part-level bounding boxes and sketches. (f) Re-assembled output.

Here, $\delta(k) = k_{i-1} + k_{i+1} - 2k_i$, representing the local feature at each point. X_0^S represents the two endpoints of a stroke s of the input sketch X while p_0 refers to the corresponding endpoints of the optimized curve. Y_i represents the closest point of the intermediate output Y to the point p_i of the optimized curve under the Euclidean measurement. We optimize the energy function in the stroke level. The term $E_{position}$ enforces the optimized curve to have the same start point and the end point as the input stroke, and the term E_{shape} constrains the optimized curve to have the same shape (curvature) as the corresponding reference stroke, as illustrated in Figure 3.5. In this way, the final beautified sketch keeps the same start points and the end points as those in the input sketch. It can be regarded as that our system is redrawing the input strokes based on their original endpoints in a more professional way.

3.1.3 Structure Beautification

Through the part beautification module, the geometry of input individual part sketches can be well refined. However, there is still a structure inconsistency existing in the input sketch. To address this issue, we further introduce a structure beautification module to adjust the imperfect structure of the input sketch with the deformed parts. The basis of this module is the sketch assembly model designed to learn spatial transformations for beautifying the deformed sketches. Figure 3.6 illustrates the workflow of training our sketch assembly model, where we first simulate the structure inconsistency by randomly applying affine transformations to the individual part sketches in our dataset. In our implementation, we combine the affine transformation operations of random scaling (ranging from 0.8 to 1.2 folds) and random translations (varying in x and y directions with -3 to 3 pixels offsets) in the image space (see Figure 3.6 (b)). As there is no obvious rotation observed in part sketches and our part beautification module is able to reduce such rotation effects on part sketches, we do not apply the random rotations to our training data here. We then design a sketch assembly network (Figure 3.6 (c)) with the backbone of the spatial transform network (STN, in short) [50] to learn the inverse mapping that transforms the distorted parts back to their ground-truth locations (see Figure 3.6 (a)). To prevent training vibration caused by the sparse input (sketch), we further introduce the part-level bounding boxes M (Figure 3.6 (e)) to enhance the spatial features of the part sketches S.

Loss function

We use the following loss function for training the assembly network:

$$\begin{aligned} L = & \sum\nolimits_{p \in P} \lambda_1 \| S_p^a - S_p^{GT} \| + \lambda_2 \| M_p^a - M_p^{GT} \| \\ & + \lambda_3 \| T_p - T_p^{Identity} \|_2^2, \end{aligned} \tag{3.7}$$

where we regard the three loss terms as sketch loss, bounding box loss, and regularization loss respectively. S_p^a and M_p^a are the part sketch and the part bounding box with the random affine transformation, respectively, while S_p^{GT} and M_p^{GT} are the corresponding ground truth in our dataset. To

stabilize the learning process, we further constrain a regularization loss on the learned transformation matrix T_p to enforce it to have a slow and mild update rather than a large vibration during the learning process. $T_p^{Identity}$ refers to the Identity mapping matrix. We set $\lambda_1 = 100$, $\lambda_2 = 1$, and $\lambda_3 = 1$ in the experiment.

3.2 Experiments

We conducted extensive experiments on freehand sketches drawn by 8 volunteers with our sketch interface. Two of them were professional interior designers with years of drawing experience, and the rest were ordinary students aged 26 to 29 with no professional drawing skills. Given an object category, we asked the invited volunteers to sketch the man-made objects in their minds as casually as possible. They can draw any shape of part sketches around the common regions of the part shadows, like the airplane presented in the sketching canvas of Figure 3.2. Finally, we collected more than 200 freehand sketches and 15-45 sketched objects for each category. See Figures 1.4, 3.8, and 3.9 for more representative drawings and the corresponding beautification results. We also construct a large dataset of part-labeled synthesized sketches to train the part-level sketch implicit representations and the sketch assembly model in our sketch beautification framework. We showcase the rendered part-annotated sketches under the best view from the existing 3D shape repositories including, PartNet [80], SDM-Net [34], and COSEG dataset [103] in Figure 3.7. Our sketch dataset contains 17,172 sketches with clear part annotations distributed over 9 man-made object categories that are commonly observed in our daily life. Figure 3.7 gives a representative sketch under each category. We provide more details of the data distribution of our synthesized sketch dataset in Table 3.1.

Implementation Details We implemented our sketch implicit model and sketch assembly model with the PyTorch framework [89] and used the Xavier initialization [35]. To train sketch implicit representations for our 128×128 part sketch, we sampled both the valid sketched points and the

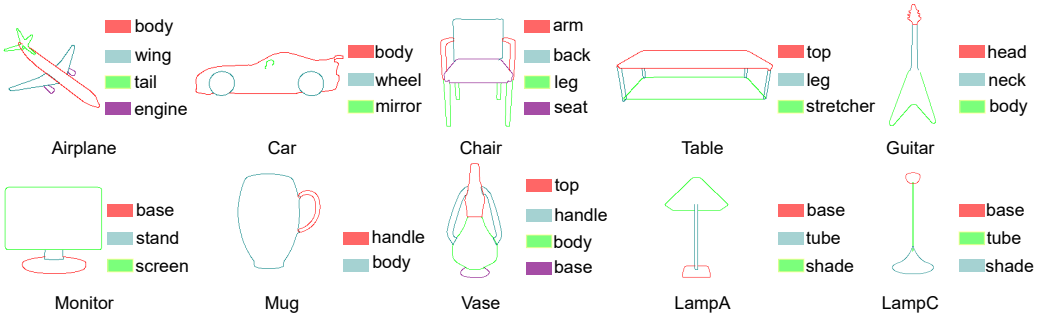


Fig. 3.7. Different categories and part annotations in our dataset. Note that the lamp class has two different labeling strategies (see the part annotations of LampA and LampC).

Categories	#Samples	#Parts	Source
Chair	5,962	4	PartNet
Table	4,440	3	PartNet
Airplane	2,467	4	SDM-Net
Car	1,813	3	SDM-Net
Guitar	741	3	SDM-Net
Monitor	559	3	SDM-Net
Vase	298	4	COSEG
Mug	211	2	COSEG
LampA	510	3	COSEG
LampC	171	3	COSEG

Tab. 3.1. Data distribution of our synthesized sketch dataset. The #Samples and #Parts refer to the number of synthesized sketch samples and components of a sketch in different categories. The Source tells which 3D repository our man-made object sketches are rendered from.

invalid points from the 2D image space, in total 16,384 points. We demonstrate the parameter structures of the sketch implicit model and sketch assembly model in the Appendix B. The two models were trained on an NVIDIA RTX 2080Ti GPU and optimized by the Adam optimizer ($\beta 1 = 0.9$ and $\beta 2 = 0.999$) with the learning rate of $5e^{-5}$ and $1e^{-4}$, respectively. Here we trained the two models to full convergence until the learning rate decayed to relatively small. Note that training the sketch implicit model takes around 48 hours on a single GPU with the batch size of 1 for one category on average. The batch size for training sketch assembly is 64. The iteration epochs for the two models were set as 800 and 600, respectively. Although it takes a long time to train these two models during the training stage,

our whole sketch beautification pipeline only spends around 1 second to beautify an entire sketch.

3.2.1 Baselines

To verify the effectiveness of our proposed sketch beautification pipeline, we compare our approach with existing methods both qualitatively and quantitatively. We use four baselines in our experiment and detail them as follows.

One of the baselines is the Laplacian smoothing algorithm [112]. We implement this method on the stroke level to process the input freehand sketches. The next one is a naive instance-level and retrieval-based approach. We take the user's freely sketched input as the query and retrieve its top-1 candidate from the dataset as the beautified result. In our implementation, we use the HOG [22] descriptor, which is efficient and able to capture the spatial features of the sparse sketches. Then, we further designed a part-level retrieval baseline. Different from instance retrieval using the entire input sketch as the query, we first parse the user's freehand sketch to individual parts and utilize the separated part sketches (not the entire sketch) to retrieve the corresponding top-1 part results from the part-annotated dataset with the HOG descriptor, and finally replace the original part sketches with the retrieved part candidates as the final beautified output. Another baseline is a learning-based sketch simplification approach Mastering Sketching [105]. Although this generative method is designed to remove the superfluous details (i.e., the repetitive scribbles) and synthesize (or strengthen) the important lines for the rough sketches, we utilize it in our sketch beautification task to generate the beautified results for the input freehand sketches conditioned on the strokes that are identified as important by its learned model. In our implementation, we directly utilize their original well-trained inference model to process the user's freehand drawings. Note that since the outputs of Mastering sketching are generated with bold effects, we post-process them with a thinning operation [153] to remove the redundant pixels and conserve the clear skeletons as final beautification results. For the closely related work ShipShape [32], since it was fully integrated into a quite old version of Adobe Illustrator and its plugin is

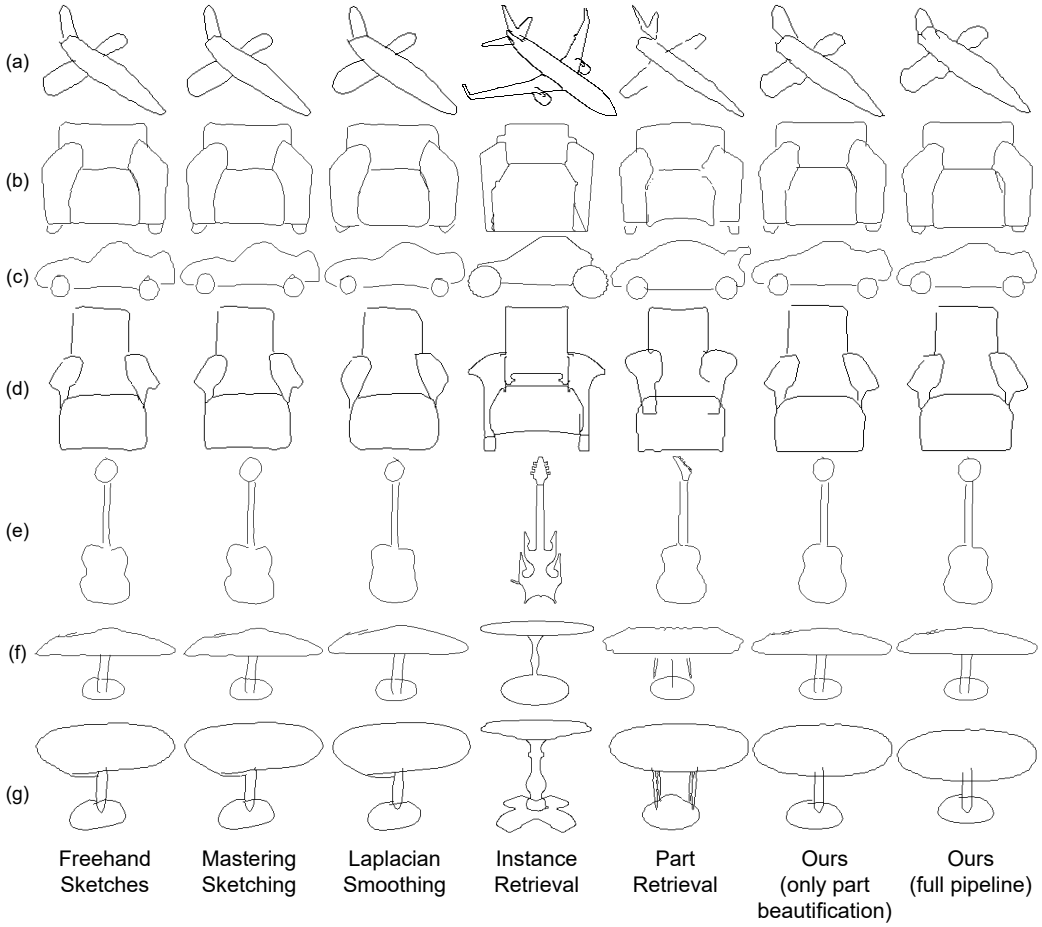


Fig. 3.8. Qualitative comparison of different approaches for sketch beautification. Please zoom in for better visualization.

no longer accessible today, we can only discuss the difference between our approach and ShipShape (Section 2.1) instead of conducting experimental comparisons.

3.2.2 Qualitative Evaluation

We evaluate our method and the competitors, namely, Laplacian smoothing, instance retrieval, part retrieval, and Mastering Sketching on freely drawn man-made object sketches. Figures 3.8 and 3.9 show the beautified results of different methods on the freehand sketches. The visual comparisons show that our method produces the most satisfying results across various

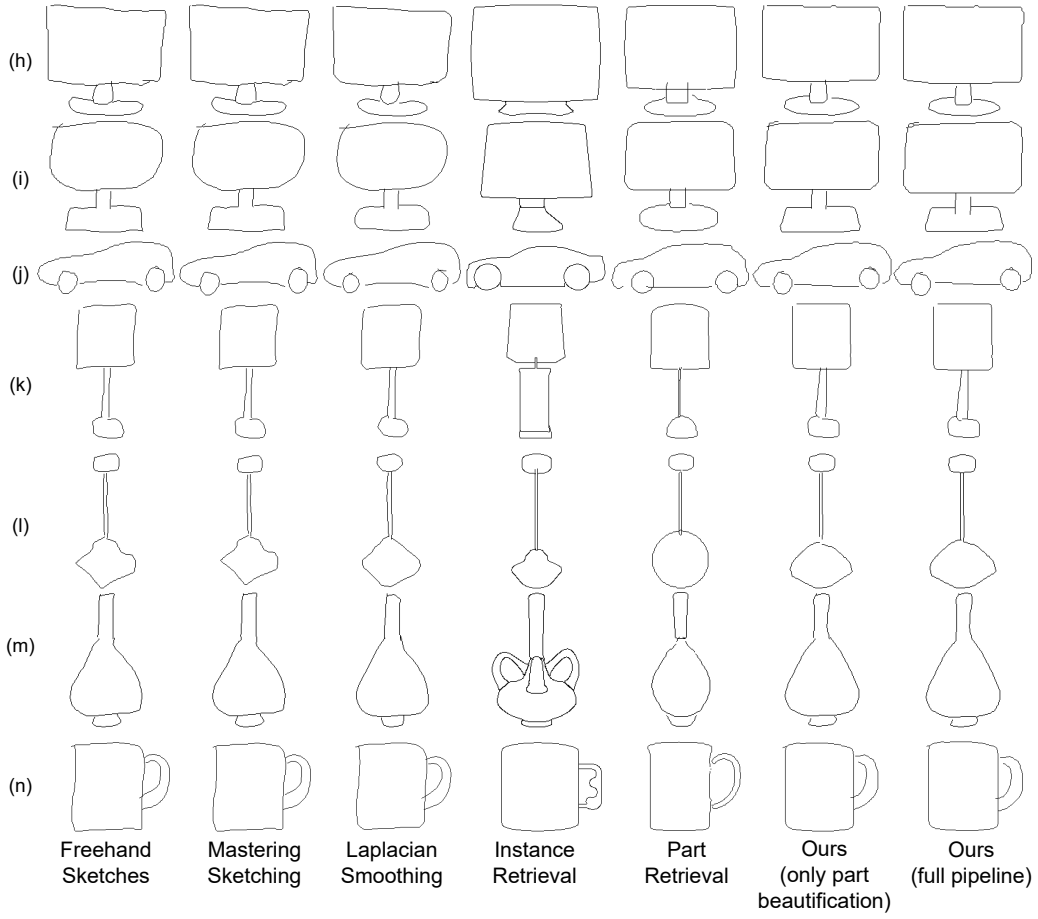


Fig. 3.9. Qualitative comparison of different approaches for sketch beautification. Please zoom in for better visualization.

categories of sketches, regardless of the local geometry or the global structure.

We observe that the outputs of Laplacian smoothing and Mastering Sketching basically keep the same shape as the input sketches. Compared with the input, the former results lose some sharp features (see car body, chair seat, and guitar body in Figure 3.8 (c), (d), and (e) respectively) and the latter are generated with bold or shadow effects. Although Mastering Sketching succeeds in healing some small seams between parts (e.g., the head and neck parts of the guitar sketch in Figure 3.8 (e)) by generating more pixels (along the two strokes of the guitar neck towards the guitar head) based on the original sketches (e)), it fails to beautify the larger seams (see the defects between the neck and body parts of the guitar sketch in Figure 3.8 (e)).

When visually inspected, the results produced by instance retrieval have the most dissimilar appearance compared with the input sketches. Due to the freedom and abstraction of hand drawings as well as the limited scale of our synthesized sketch dataset for retrieval, instance retrieval cannot find a pleasing beautified counterpart from the existing sketch dataset for the user's input sketch. Although part retrieval tries further to approximate the input sketches from a more fine-grained level, it only alleviates the dissimilar phenomenon. Still, these retrieval-based methods have no way to maintain the user's original drawing intentions.

To better demonstrate our sketch beautification method, we further present the intermediate outputs of the part beautification module in Figures 3.8 and 3.9. Unlike previous learning-based or retrieval-based competitors, our part beautification module not only produces the most pleasing and elegant part sketches by redrawing the input part strokes leveraging the knowledge of a large number of synthesized part-level sketches, but also retains the relative close shapes to the original inputs without introducing extra strokes, missing any strokes, and making no change to the input shape. However, as shown in Figures 3.8 and 3.9, some structure defects like seams (b), penetration (j), and misalignment (g) between part sketches still exist in the intermediate outputs of our part beautification module. This further emphasizes the importance of structural adjustment. When combined with the other component, our structure beautification module, we can obtain the most beautified outputs with pleasing part geometries and convincing global structures. However, since there is no existing standard criteria to evaluate our proposed sketch beautification approach and its competitors, we give our own insights and present the evaluation strategy as follows: In fact, beautifying a sketch means adding changes to it while respecting the original sketch. There is no perfect solution here, since two main aspects (i.e., improving beautification quality while respecting the original sketch) need to be balanced during the beautification. Regarding faithfulness to the input sketch and beautification quality, we further conduct a quantitative evaluation on the beautification faithfulness and a perceptive study on the beautification quality in Subsections 3.2.3 and 3.2.4.

Methods	mCD ↓	mEMD($\times 10^2$) ↓
Instance Retrieval	15.55	5.92
Part Retrieval	8.22	5.20
Laplacian Smoothing	3.44	4.52
Mastering Sketching	**1.40**	**2.04**
Ours (only part beautification)	4.59	4.62
Ours (full pipeline)	6.84	5.04

Tab. 3.2. Quantitative evaluation on the faithfulness of the beautified results (produced by different sketch beautification methods) to the input sketches.

3.2.3 Quantitative Evaluation on Faithfulness

To quantitatively evaluate the performance of different beautification approaches in preserving the user's original drawing intentions, namely, faithfulness, we report the statistic values of two metrics for the aforementioned methods in Table 3.2. To measure the difference between a beautified result and the user's original freehand sketch, we adopt the Chamfer Distance-L2 (CD) and Earth Mover's Distance (EMD) as the evaluation metrics (lower is better) for the faithfulness evaluation in the sketch beautification task. The former metric is employed to measure the point-wise distance between the sketched objects in two sketches and the latter one is utilized to compute the distribution-level distance of two point distributions over the entire image space (256×256). Given a pair of the user's freely sketched input and the beautified output produced by one of the compared methods, we calculate the Chamfer Distance of the only valid sketch pixels (i.e., excluding the background pixels) in the two sketches. When computing the Earth Mover's Distance, we preserve both the sketch pixels and the background pixels and treat the pair of the input and output sketches as two distributions. Finally, we average these two metrics over pairs of sketches before and after beautification in our collected freehand sketch dataset and present the mean values of CD and EMD in Table 3.2.

Consistent with what is observed from Figures 3.8 and 3.9, our method outperforms its retrieval-based competitors while falling behind Mastering Sketching and Laplacian smoothing quantitatively. It is reasonable that the results of Mastering Sketching have the closest distance to the input sketches since this method inherits (or accepts) all the input strokes

with limited pixel-level beautification. For Laplacian smoothing, since this method only slightly changes the position of the sketched points by smoothing the local-level strokes, it is easier for this method to achieve a remarkable performance (the second) in faithfulness evaluation. While the fine-grained part retrieval outperforms the instance retrieval significantly, it is still inferior to our method. The beautified outputs of our part beautification module and our full pipeline achieve the competitive performance to the Laplacian smoothing in preserving user's original drawing intentions (the third and the fourth close to the user's freehand sketches). It is expected that the results of our part beautification module have the closer distance to the users input sketches compared with our full pipeline since the part beautification module only beautifies the curve shape of the part sketches without adjusting the scales or locations of the part sketches. Just as we discussed before, there are still a lot of structure errors (e.g., seams, penetration, and misalignment) in the intermediate beautified outputs, as shown in Figures 3.8 and 3.9 (Ours (only part beautification)). To correct such structure errors during the structure beautification stage, a small range of structure refinement (including pixel-wise translations and scalings) is applied to the beautified part sketches. Therefore, it is inevitable that our full pipeline combining the part beautification and structure beautification modules slightly enlarges the distance to the input freehand sketches. For the faithfulness aspect of the sketch beautification task, since our framework is proposed to beautify both the local geometry and global structure of input sketches, our method cannot keep the input sketches almost unchanged as Mastering Sketching. But our approach does not heavily change the input sketches as instance and part retrieval, and can also be regarded as a larger "enhancement" operation (making the original input sketches aesthetically more beautiful in both local curve and global shape) to some extent. For the evaluation on the beautification quality of the different sketch beautification methods, we demonstrate it with a user study in the next section.

3.2.4 Perceptive Study on Beautification Quality

It is known that the concept of beautification is highly relevant to human preference and perception. Therefore, to evaluate the beautification quality of the beautified sketches produced by different methods in Section 3.2.1 (namely, Laplacian smoothing (LS), Mastering Sketching (MS), instance retrieval (IR), part retrieval (PR), our part beautification module only (Ours (PB)) and our full approach), we further conducted a user study to evaluate the performance of the compared approaches in the sketch beautification task.

Specifically, we first randomly picked a set of 15 input sketches from the drawn freehand sketches, spreading all the categories in our dataset. We then applied the aforementioned six sketch beautification methods to each input sketch to generate the beautified results. Figures 3.8 and 3.9 display some representative examples of inputs and outputs used in our user study. The evaluation was done through an online questionnaire. There were in total 23 participants (15 males and 8 females) in this study. We showed each participant sets of an input sketch and six beautified sketches generated by the compared approaches in random order set by set. Each participant was asked to select the most beautiful result in each set regarding the visual aesthetics from both aspects of the local geometry and the global structure. Finally, we received 23 (participants) $\times$ 15 (input sketches) = 345 subjective evaluations.

Figure 3.10 shows the statistics of the voting results. We conducted one-way ANOVA tests on the preference voting results, and found significant effects for aesthetic preference to our method ($F = 65.88$, $p < 0.001$). The further paired t-tests show that our method (mean: 13.93) got significantly more votes than all the other methods, Mastering Sketching (mean: 3.13, [$t = 2.14$, $p < 0.001$]), Laplacian smoothing (mean: 2.47, [$t = 2.14$, $p < 0.001$]), instance retrieval (mean: 1.00, [$t = 2.14$, $p < 0.001$]), part retrieval (mean: 0.47, [$t = 2.14$, $p < 0.001$]), and our part beautification module (mean: 2.00, [$t = 2.14$, $p < 0.001$]).

To summarize, our approach achieved the best performance that largely surpasses its competitors under human aesthetic perceptions. Although there are slightly higher deviations in the beautified results generated by our

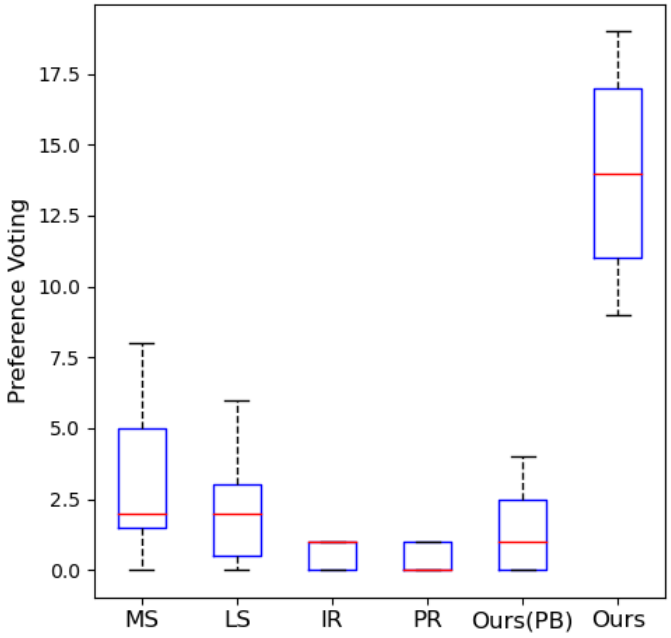


Fig. 3.10. Box plots of the average preference voting over the prepared questions (beautification quality) for each method. MS, LS, IR, and PR stand for Mastering Sketching, Laplacian smoothing, Instance retrieval, and Part retrieval, respectively. Ours (PB) and Ours refer to the part beautification module and the full pipeline of our sketch beautification approach, respectively.

method compared with the original input sketches, our beautified sketches are still able to be faithfully voted as the best beautification results for the given sketches by the invited users. In addition, our approach successfully beautifies the local-level part geometry and corrects the global-level structure errors of input sketches. In this way, our proposed method helps to improve the beautification quality of input freehand sketches significantly. We show more beautification results of our method (including the intermediate outputs after part beautification and final results after structure beautification) in the Appendix B.

3.2.5 Ablation Study

Since we have demonstrated the roles of the part beautification module and the structure beautification module of our sketch beautification pipeline qualitatively and quantitatively in the above two subsections, and detailed

Input	Loss	mIOU(%)↑	mCD↓	mEMD($\times 10^2$)↓
sk	skloss	0.00	$+\infty$	$+\infty$
sk	(sk+regu)loss	83.38	5.85	6.75
sk+bb	bbloss	91.87	2.79	5.42
sk+bb	(sk+bb)loss	91.83	2.79	5.42
sk+bb	(sk+bb+regu)loss	**91.89**	**2.78**	**5.40**

Tab. 3.3. Ablation study of our designed mechanism for the sketch assembly model. The sk and bb in the first column are the inputs of the sketch and the bounding box, respectively. The skloss, bbloss, and reguloss in the second column refer to the supervision losses of the sketch, the bounding box, and the regularization term, respectively.

the key components of the part beautification module in Section 3.1.2, we do not conduct the redundant ablation studies for the part beautification module here.

Hence, in this subsection, we mainly focus on validating the effectiveness of the key components in the structure beautification module. Since the sketch is a kind of sparse representation, it is a nontrivial task for our sketch assembly model to represent and learn its spatial transformations precisely and effectively. To validate our designed mechanism of the sketch assembly model, we perform ablation studies of its key components in turn, namely, the sketch loss, the bounding box loss, and their combinations with the regularization loss (see Table 3.3).

Specifically, we use 1,000 randomly sampled synthesized sketches of the Chair as the ground truth since this category has the most complex and challenging structures in our dataset. We then apply random affine transformations to the parts of the ground truth samples five times and take these 5,000 deformed sketches with the warped parts as the benchmark input. Finally, we evaluate the performance of the trained sketch assembly models under different training strategies by measuring the disparity between their assembly outputs and the ground truth. To better reflect the performance of the methods in the sketch assembly task, we compute a region-based and part-level IOU metric on bounding boxes of the transformed part outputs and the corresponding part ground truth, as formulated in the following equation:

$$IOU = \frac{1}{N_p} \cdot \sum\nolimits_{p \in P} \frac{M_p^a \cdot M_p^{GT}}{M_p^a + M_p^{GT}}, \tag{3.8}$$

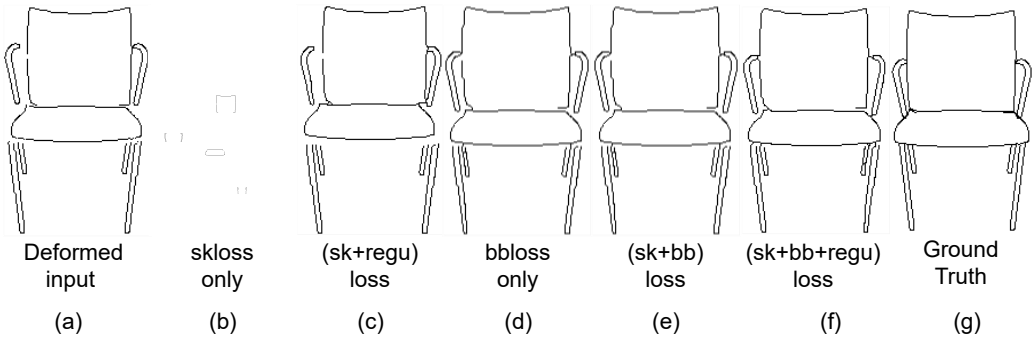


Fig. 3.11. Visual comparison of different supervision losses during the training process. Please zoom in for better visualization (in particular for "skloss only").

where N_p is the number of parts P in a sketch, M_p^a and M_p^{GT} are the part-level bounding boxes of the assembled sketches and ground-truth sketches, respectively. We also calculate the Chamfer Distance-L2 (CD) and Earth Movers Distance (EMD) between the assembled and ground-truth sketches to further verify the performance of the ablated sketch assembly models. Table 3.3 reports the mean values of the above metrics. We also show the performance of different components in the sketch assembly task qualitatively in Figure 3.11.

In our experiments, we found that only utilizing the sketch loss cannot supervise the learning process of spatial transformations of sparse sketches. Due to limited valid pixels in sketches, the network tends to shift its focus from sketched pixels to the background pixels during the training process, even under the L1 loss of the input and output sketches. With the number of training epochs increasing, the model degrades rapidly and ignores the sketched pixels until the part sketches are rescaled to none. Hence, the mean IOU value of 'skloss only' model in Table 3.3 is 0. The other two metrics also show the failure of training on sketch input with the sketch loss only. This can be further witnessed in Figure 3.11 (b). Note that we screen-captured this subfigure of "skloss only" at the very beginning of training since the part sketches would disappear soon after several further epochs. Along with the degradation of "skloss only" model, we observed a significant increment in the parameters of the learned transformation matrices. We further designed and added the regularization loss to punish this huge vibration of the learned transformation matrices. However, although the

input part sketches are no longer shrunk to nothing by impelling the additional regularization loss over the learned transformation matrices, the network still failed to learn meaningful spatial transformations (just random translation or scaling under the constraint of the regularization loss), as shown in Figure 3.11 (c). Only by introducing the bounding box input and the corresponding bounding box loss, the networks were able to learn the spatial transformation of sparse sketches stably and effectively (see the inputs containing 'bb' and the losses with 'bbloss' in Table 3.3 and Figure 3.11 (d-f)). With the combination of the sketch loss, the bounding box loss, and the regularization loss, the network achieved the best performance, as shown in Table 3.3 quantitatively and Figure 3.11 (f) qualitatively. These results further confirm the necessity and importance of our design choices for the structure beautification module.

3.3 Discussion

We have introduced an intuitive and generic beautification approach for freehand sketches depicting man-made objects by conducting part-level geometry beautification and global structure refinement sequentially. As one of the key components in our approach, the sketch implicit model can be easily plugged into contemporary deep neural networks for a variety of tasks relevant to sketches including sketch recognition, classification, retrieval, reconstruction, and generation thanks to its promising representation capacity over existing representations. The other component, i.e., the sketch assembly model, provides a robust and effective solution for compositing sparse 2D components by having the part-level bounding boxes. The whole beautification pipeline could further inspire and boost the downstream applications that operate over freehand sketches as input. However, beautifying a freehand sketch under an arbitrary view is still challenging for our method. We leave this as future work to explore.

Our method has some limitations. First, in our method, all the training samples of the sketches of man-made objects are rendered under the canonical view (best view) for each category. This limits the adaptation ability of our method to freehand sketches with large view variations or multi-view

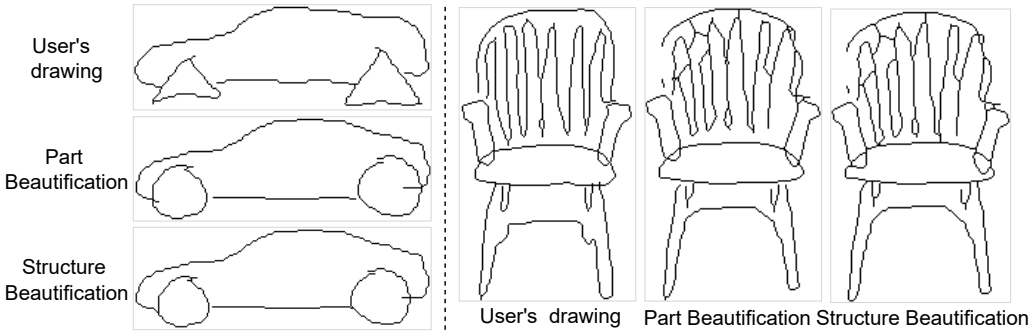


Fig. 3.12. Failure cases of our sketch beautification method.

sketches. Although this is partially solved by adding a shadow guidance to constrain the user's input, we are interested in addressing this issue in a more elegant way [147]. Second, a key merit of our method is the representative power of the sketch implicit representations that can interpolate the sketched points smoothly and continuously. But this implicit representation requires a longer training time than the CNN representations on the same input data, since the implicit model needs to sample all the coordinates and remember the ground truth value of each point in the 2D image space. One possible solution for speeding up the training process is to change the sampling strategy, e.g., by keeping sampling the points close to the strokes instead of sampling all the points from the whole 2D space. Third, while our method is able to instantiate conceptual freehand sketches, our approach could fail if input sketches are drawn too poorly or too complexly, as shown in Figure 3.12. If a user draws a very unnatural sketch (see the triangle-like wheels of a car on the left-top, our method might not follow the user's drawing intention and even totally changes the geometry of part sketches drawn by the user. This violates the beautification constraints we set in this chapter (as discussed in Section 1.1). While a user sketches a very complex part (see the chair back with too many sticks on the right), our approach also cannot beautify such a part. This is due to the failure of the correspondence matching step in Section 3.1.2. It is known that correspondence matching is still an open problem in the research community. Therefore, the improvement on registration and correspondence matching could also further boost the performance of our method. Lastly, as humans' perception and preference for beautification concepts are similar but not exactly the same (shown in Figure 3.10 in our perceptive study), ideally, users should be allowed to adjust the degree of beautification (more signif-

icant beautification would lead to larger deviation from the input sketch). In our approach, the beautification function is designed in a closed and automatic way for efficiency reasons. In the future, we would like to extend our sketch beautification approach to allow for more user control.

4

Sketching with Structural Stress Awareness

With our proposed sketch beautification approach (Chapter 3), both the part geometry and global structure of a freehand sketch can be well refined. However, the structural soundness of the sketched objects under external environmental forces is still unknown to designers and novice users during the process of product design and digital fabrication. In this chapter, we propose our solution of *Sketch2Stress* to bridge this gap. We present the methodology of our *Sketch2Stress* in Section 4.1. We introduce quantitative, qualitative, and ablation evaluations in Section 4.2. We present one application of *Sketch2Stress* in Section 4.2.5. Finally, we summarize our *Sketch2Stress* in Section 4.3.

4.1 Methodology

In this chapter, we focus on the study of structural analysis of sketched objects under external forces at user-specified locations. Adapting the structural analysis task from informative 3D objects to 2D sketches is nontrivial due to the ill-posed nature of sparse sketches to represent continuous and closed 3D surfaces as well as the challenge of representing external forces applied to the sketched objects. To address these issues, we simplify the problem and make our assumptions as follows: (i) We decouple the external forces to the constant force magnitude of $100N$ and directions based on the estimation of a normal map (the force direction and the normal direction at the force location are opposite in our approach, as shown in Figure 2.2 (b)). (ii) Then we utilize an effective data-driven way to approximate the mathematically/physically precise stress by constructing a novel large-scale sketch-force-stress dataset and proposing a new two-branch (for force

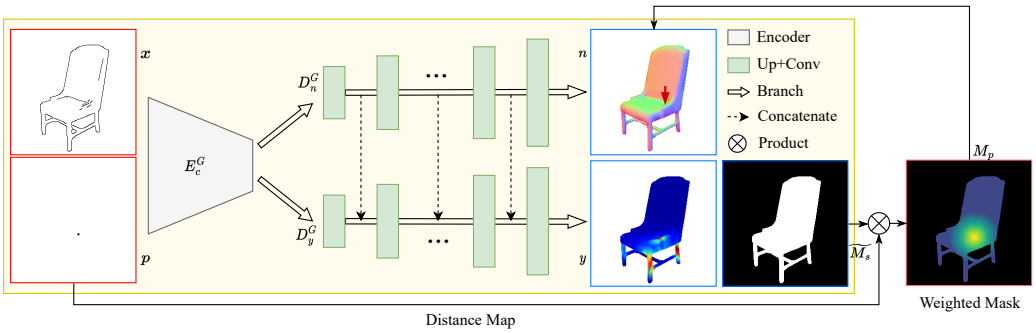


Fig. 4.1. Overview of the two-branch generator of *Sketch2Stress*. Given an input sketch (upper left) and an input point map (lower left) indicating a force location, the two-branch generator uses its encoder to learn a sketch-force joint feature space, and then leverages two decoders to synthesize the corresponding stress map (lower branch) and a normal map (upper branch). We use warmer colors (reds and yellows) to show high stress and cooler colors (greens and blues) to show low stress. The normal map infers the force direction at the input force location. A shape mask and a point-attention mask are proposed to further emphasize the shape boundaries and the user-specified force location during the generation process.

location and direction) generation pipeline (see Figure 4.1). (iii) Note that the materials of the sketched objects are assumed to be the same with linear isotropic materials and small deformations, following [126].

To faithfully represent the external forces applied to sketched objects, we utilize a set of 2D point maps P to specify the force locations (one point map for each force location) and a 2D normal map $n \in R^{256\times256\times3}$ of each view of an object to record the force direction $-n_p$ at the corresponding location p. In this way, we decouple the original 3D external forces into the above 2D representations that can be further treated as conditions for mapping the input sketches X to the corresponding stress maps Y. Let N denote the set of normal maps.

As illustrated in Figure 4.1, Our framework for sketch-based structural analysis consists of two components: (1) a two-branch generator G: $(x, p) \rightarrow (n, y)$, including a common sketch-force encoder E_c^G and two separated decoders D_n^G and D_y^G for the normal map n and the stress map y, illustrated in Figure 4.1, and (2) two multi-scale discriminators D_n and D_y for normal and stress maps. Specifically, given an input sketch and the condition of a point map, the common encoder of our two-branch generator

constructs a joint feature space $E_c^G(x, p)$ for the input sketch and the input point map. The subsequent two decoders (7-layer up-sampling and convolution) D_n^G and D_y^G infer the correct normal directions $\tilde{n}$ and a feasible structural stress map $\tilde{y}$ from this joint feature space, respectively. These two branches enforce that the common encoder E_c^G should learn a joint feature representation that captures not only the geometry and normal directions of the input sketch but also the distinctive force location on the input point map. Note that the feature maps in the normal decoder D_n^G are layer-wise concatenated to the stress decoder D_y^G to enrich its structure perception. Finally, the two multi-scale discriminators distinguish real images from the translated ones at 256×256, 128×128, and 64×64 scales. This is a standard way to represent distinctive, fine-grained details in images [132]. Finally, we jointly optimize G, D_n, and D_y with the objective:

$$\begin{aligned} L_{G,D} = & E_y[\log D_y(y)] + E_n[\log D_n(n)] \\ & + E_{x,p}[\log(1 - D_n(G(x,p))) + \log(1 - D_y(G(x,p))], \end{aligned} \tag{4.1}$$

where x, p, n, and y refer to the quadruple of an input sketch, a point map, a normal map, and the corresponding stress map.

Shape Constraints To overcome the issue that the generated pixels are often outside of the shape boundary of the sketched objects in the stress maps, we further predict a one-channel shape mask $\widetilde{M_s}$ (Figure 4.1) from the joint feature space $E_c^G(x, p)$. This shape mask is also useful for reducing shape ambiguity in normal map generation. Therefore, we use a shape loss L_{shape} to measure the $L1$ distance between a generated shape mask and the ground truth, as formulated below:

$$L_{shape} = L1(\widetilde{M_s}, M_s). \tag{4.2}$$

Force-point Constraints To emphasize the importance of a force location in the point map, we compute a point attention map M_p by multiplying a point-centered distance map [1] (emphasizing the spatial importance of

[1] This is computed by $D = \frac{1}{|Q|} \sum_{q \in Q} \|q_i - q_f\|_2$, q_i and q_f are the locations of every spatial point and the force point of a point map, respectively.

regions that surround the assigned force point) with the shape mask. We multiply this attention map respectively with the normal map and the stress map to ensure that the synthesized stress and normal directions surrounding this force point should be consistent with the ground-truth values as much as possible. Here, we design a loss term L_{point} to compute the $L1$ distance of the generated stress and normal maps compared with their respective ground truths inside the regions M_p, as defined below.

$$L_{point} = L1(M_p \cdot \widetilde{y},\ M_p \cdot y) + L1(M_p \cdot \widetilde{n},\ M_p \cdot n). \tag{4.3}$$

Therefore, our final objective function is formulated as follows:

$$L = L_{G,D} + \lambda_1 L_{shape} + \lambda_2 L_{point}, \tag{4.4}$$

where we set $\lambda_1 = 500$ and $\lambda_2 = 100$ in our experiments.

4.1.1 Sketch-to-Stress Data Rendering

To learn our network for sketch-based structural analysis, we need a considerably large dataset of training data. However, such a dataset is not available and expensive to acquire since it requires point-wise labeling for external forces and corresponding stress responses on the sketches. Hence, we propose to synthesize the sketch-to-stress data from existing 3D repositories, as shown in Figure 4.2.

We first collect 3D shapes from several public shape repositories, including ShapeNet [12], AniHead [27], and COSEG [134]. We convert 3D objects to watertight surfaces with [114] to make it ready for the subsequent 3D structure analysis. We then orient all 3D shapes uprightly [33], move them onto the ground plane (for fixing their bottom on the ground plane), and normalize them to a standard sphere. To normalize and uniform the force regions of different 3D shapes with diverse structures, for shapes in the same category, we use the same ratio (0.02% $\sim$ 0.04%) to define regions on 3D surfaces near the ground plane as fixed boundary conditions and the rest as contact regions allowing for any external forces (of $100N$ magnitude), as illustrated in Figure 4.2. Given each 3D shape, we uniformly

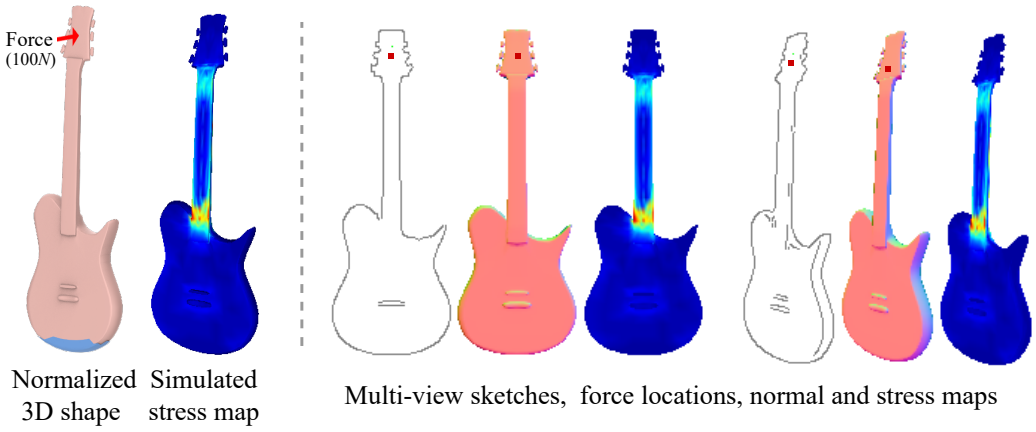


Fig. 4.2. Illustration for data preparation. The left is a normalized guitar model (the bottom blue part is the fixed boundary condition and the upper is the contact regions) and the 3D structure stress result under an external force at a specific position. The right is the synthetic sketch-to-stress data. We plot the force location on the 2D sketches and normal maps.

sample force locations on its contact regions and adopt the structural analysis approach in [126] to simulate the stress responses on the shape's surface under such forces in opposite-normal directions. Finally, we render the multi-view sketches, normal maps, force locations, and corresponding structural stress values S from the simulated 3D stress results. All of the above renderings are projected in the 256×256 spatial resolution. The synthetic multi-view sketches are extracted from 2.5D normal maps using the Canny edge detector [11]. We project the 3D stress results with the azimuth angles of $[0, 45, 90]$ degrees and the elevation angles in $(0 \sim 15)$ degrees.

Since the magnitude of the simulated stress values S spans an extremely large range from ten to ten million, we further normalize the structural stress values S to a common $[0, 1]$ space in two manners: One is a shape-grained normalization to compare the fine-grained regional stress among single shapes; the other is a category-grained normalization to compare more general-grained shape stress among all the shapes in the same category.

Shape-grained Normalization As mentioned in Section 1.2, similar shape structures tend to have similar weak regions under the same external forces.

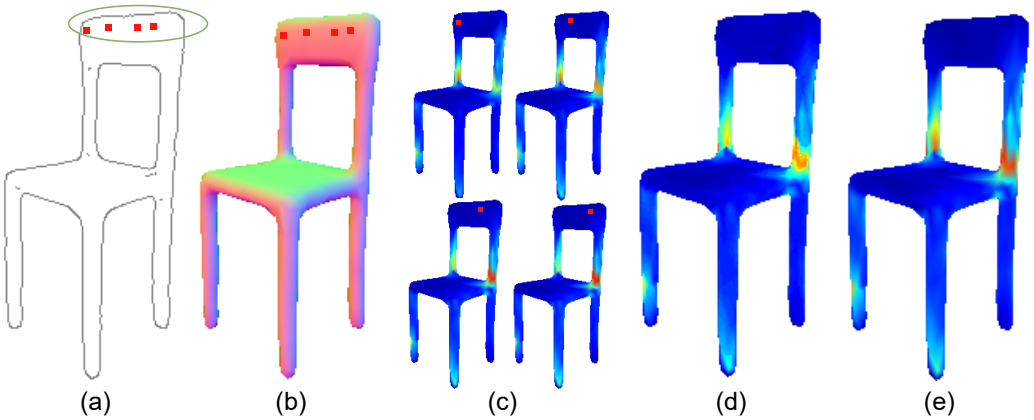


Fig. 4.3. Pipeline of multi-force aggregation. (a) Input sketch and multi-force locations in a local region. (b) Normal directions of multiple forces. (c) Four stress maps corresponding to each of the force locations. (d) Aggregated stress effect from (c). (e) Ground-truth 3D simulation of multiple forces.

To highlight such region-wise stress similarity, we normalize the stress values inside each shape, as formulated below:

$$S_i = \frac{S_i}{\max(\{S_i\})}, \qquad i \in a\ single\ shape \tag{4.5}$$

where i is an index for points in the contact regions of a 3D surface, and S_i is a stress value at the i-th point.

Category-grained Normalization To study the general pattern (knowledge) of how different shape structures respond to the same external forces, we normalize the stress values of all the shapes in the entire dataset as Equation 4.6.

$$S_j = \frac{S_j}{\tau}, \quad s.t.\ S_j = \frac{S_j - u(S_j)}{\sigma(S_j)}, \quad j \in all\ shapes \tag{4.6}$$

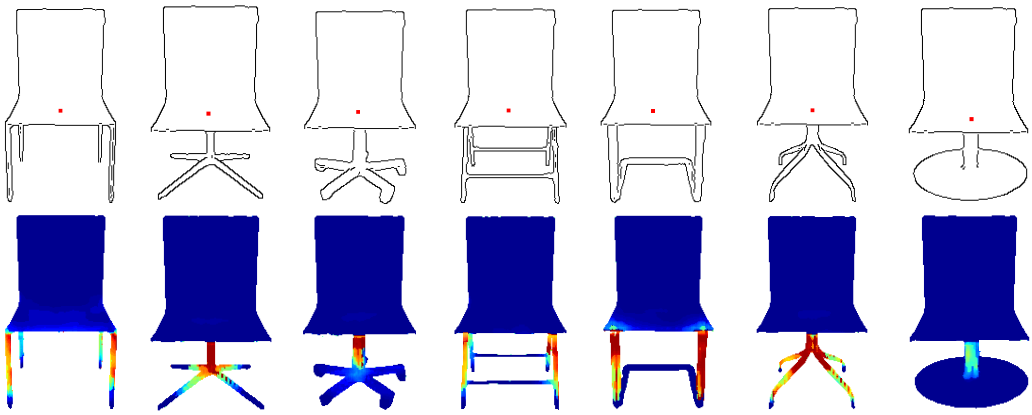
where j is the index of points in the contact regions on 3D surfaces, and $u(\cdot)$ and $\sigma(\cdot)$ are the mean and standard variance of the entire stress value set, respectively. $\tau = 100$ is the upper boundary of the 99% stress value.

4.1.2 Region-wise Multi-force Aggregation

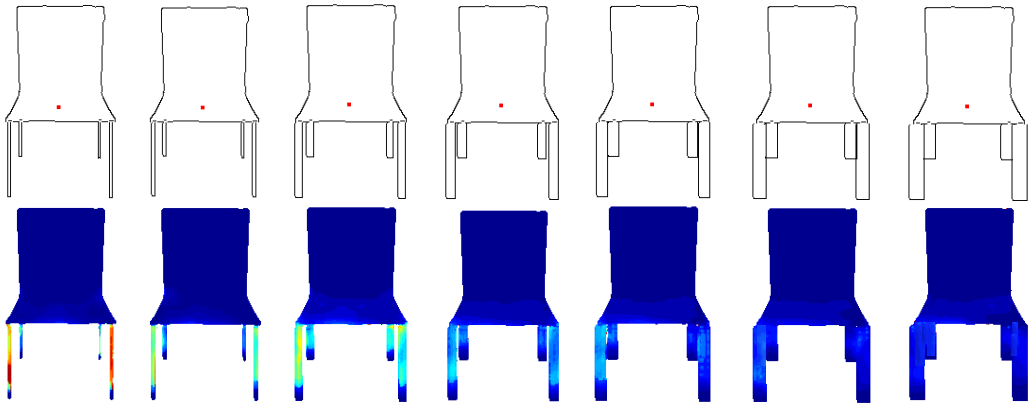
With our trained network, users can easily explore the structural stress anywhere under a manually-assigned force location by clicking on the sketched object. To further improve the efficiency of structural analysis on input sketches, we provide a region-wise analysis method that aggregates the stress effects of multiple forces in a small region along the same normal directions. After the user specifies a small region on a sketch (Figure 4.3 (a)), with the predicted normal map, we automatically compute the force locations that have the same normal direction as the center point of this region (Figure 4.3 (b)). Then we directly add and average these stress effects (Figure 4.3 (c)) together following the physical axiom in Section 1.2 and produce an aggregated stress map (Figure 4.3 (d)). Compared with a 3D simulated result (Figure 4.3 (e)), although the overall stress effect in our aggregated stress map is diluted to some extent, it can still approximate the stress distribution of the 3D simulated result well and can thus be utilized as guidance for fragile detection.

4.1.3 Structural-Stress Awareness Replacement and Interpolation

To demonstrate the sensitiveness of our *Sketch2Stress* to the variations in sketch structures, such as the significant structure changes (Figure 4.4 (a)) and the more subtle geometry interpolations (Figure 4.4 (b)), we first decompose an example chair into parts and then replace the original chair legs with legs featuring significantly varied geometries and by linearly interpolating the thickness values of the original chair legs, respectively. Note that we keep the other parts of this example chair unchanged in these two tasks. As shown in Figure 4.4 (a), the results are in line with our expectation that our well-trained *Sketch2Stress* is natural to perceive the structural soundness among highly changeable structures and identify corresponding fragile regions. Figure 4.4 (b) further demonstrates our *Sketch2Stress* algorithm's capability in perceiving the tendency of thickness increment, distinguishing the subtle differences among highly similar structures, and generating the smooth stress distributions for those interpolated structures.



(a) Chair-leg Replacement



(b) Chair-leg Interpolation

Fig. 4.4. Examples of our *Sketch2Stress* on Structure Replacement (a) and Geometry Interpolation (b).

This could facilitate a sketch-based structural soundness suggestion task, where users could easily improve the structural soundness of their created sketches with our *Sketch2Stress* tool combined with the replacement and interpolation operations.

4.1.4 Sketching Interface

To illustrate how our proposed method aids users in analyzing and strengthening the structural weakness of their sketched objects under external forces, we design a simple interface (Figure 4.5) for users to interactively edit sketches, assign external forces at specific positions to examine the stress effects, and refine their design.

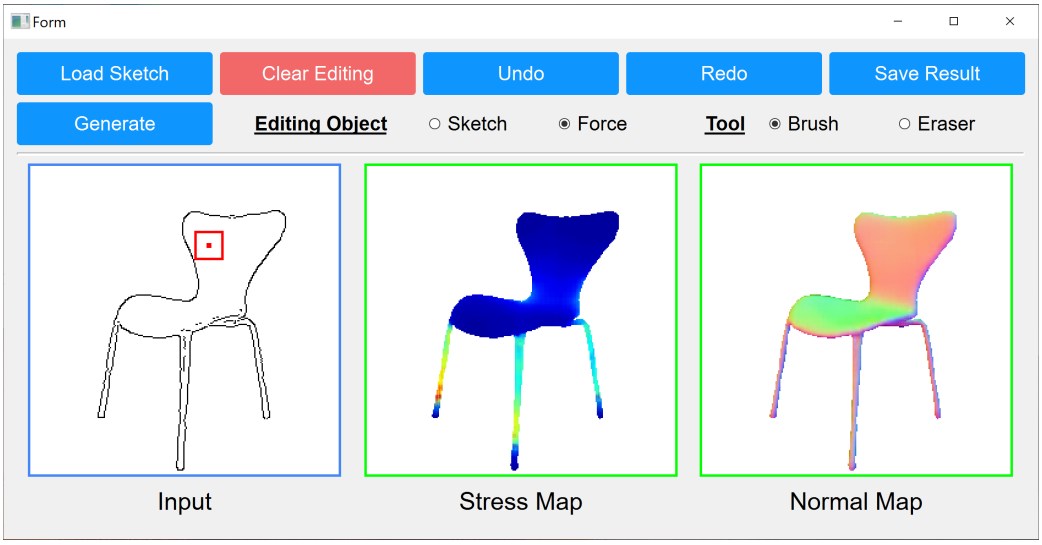


Fig. 4.5. Our sketching interface.

Our system has two modes (named the sketching mode and the simulation mode), and they can be selected through the “Sketch" and “Force” radio buttons. In the sketching mode, users can load their drawn sketches or directly create one from scratch in the “Input" region. We also provide several basic drawing tools for users to edit their drawings, such as clear, undo, and redo. After finishing one complete sketch, users may change to the simulation mode. In this mode, users can freely impose external forces at desired positions by clicking on their drawn sketch and examine the potential weaknesses through the simulated structural stress map and the normal map. The auxiliary normal map provides clearer (2.5D) shape details for designers than the input sketch and the predicted stress map (most regions are in the same color, deep blue, providing limited shape details), as observed in the middle and bottom rows in Figure 4.6. During the structure refinement process, the generated normal map can greatly help users to iteratively improve their original drawings at a fine-grained level with its provided shape details. By iteratively using these two modes, designers can create their desired shapes that are also structurally sound under certain external forces.

Fig. 4.6. Result gallery of eleven categories in our synthetic sketch-to-stress dataset. The top row shows the input sketches and external force locations (plotted as red dots), while the middle and bottom rows are our generated normal maps (with predicted force directions at the center of red boxes) and synthesized stress maps, respectively. Please zoom in to examine the details of the applied force locations/directions and the generated structural stress results.

4.2 Experiments

We evaluate our approach on 11 shape categories with a large variety of geometry and structure, as shown in Figure 4.6. The 3D shapes used for sketch rendering and force-conditioned structural stress simulation are collected from the existing 3D shape repositories including ShapeNet [12], COSEG dataset [103], and AniHead dataset [27]. In total, our synthetic dataset contains over 2.7 million sketch-force-stress data pairs with clear point-wise force annotations. The dataset spans 11 categories, namely, chairs (1.7 million), tables (0.7 million), airplanes (0.4 million), vases (22K), mugs (15K), skateboards (24K), rockets (4K), guitars (9K), fishes (2.6K), animal heads (78K), and four-leg animals (13K). After data augmentation, our collected data is able to train our neural network with satisfying generation quality. We provide more details of the data distribution of our sketch-to-stress dataset in Table 4.1.

Implementation Details We implemented our *Sketch2Stress* with the PyTorch framework [89] and used the Xavier initialization [35]. We show

Category	#Shape	#Views	#Sketches	#Force-points	#Stress-map
Chair	4,277	3	12,831	1,523,390	1,523,390
Table	3,656	3	7,312	715,566	715,566
Airplane	2,231	3	6,693	403,926	403,926
Vase	184	1	184	22,824	22,824
Mug	164	1	164	15,840	15,840
Skateboard	134	3	402	24,471	24,471
Rocket	49	1	49	4,291	4,291
Guitar	39	2	78	9,186	9,186
Fish	20	1	20	2,640	2,640
Fourleg	42	3	126	13,181	13,181
AniHead	208	3	624	78,528	78,528

Tab. 4.1. Data distribution of our synthesized sketch-to-stress dataset. The #Shaps and #Views refer to the numbers of 3D shapes and projection views in different categories, respectively. While #Sketches, #Force-points, and #Stress-map represent the numbers of rendered 2D sketches, sampled force locations to apply external forces, and the ground-truth simulated 2D stress maps, respectively.

the parameter structures of the two-branch generator of *Sketch2Stress* in the supplemental materials. The entire pipeline of our *Sketch2Stress* was trained on an NVIDIA TITAN Xp GPU and optimized by the Adam optimizer ($\beta 1 = 0.9$ and $\beta 2 = 0.999$) with the learning rate of $2e^{-4}$. Here we trained our models to full convergence until the learning rate decayed to relatively small. Note that training takes $24 \sim 48$ hours on a single GPU with a batch size of 16 for one category on average. The iteration epochs are set to 10 for those categories with a large number of training samples, namely, Chair, Table, and Airplane. For the rest categories, we set the training epochs to 100, which is sufficient to achieve the satisfying generation performance in our experiments. Although it takes a long time to train our *Sketch2Stress* during the training stage due to the large size of training samples, the well-trained two-branch generator of *Sketch2Stress* only spends around 0.0005 seconds on average to infer a structural stress map for an input sketch under a specified force.

Here we compare our method with two image-to-image baselines, i.e., pix2pix [49] and pix2pixHD [132], quantitatively and qualitatively. We then perform an ablation study to illustrate the improvement provided by each key component in our method. Finally, we use three user studies to demonstrate the practicality of our proposed method.

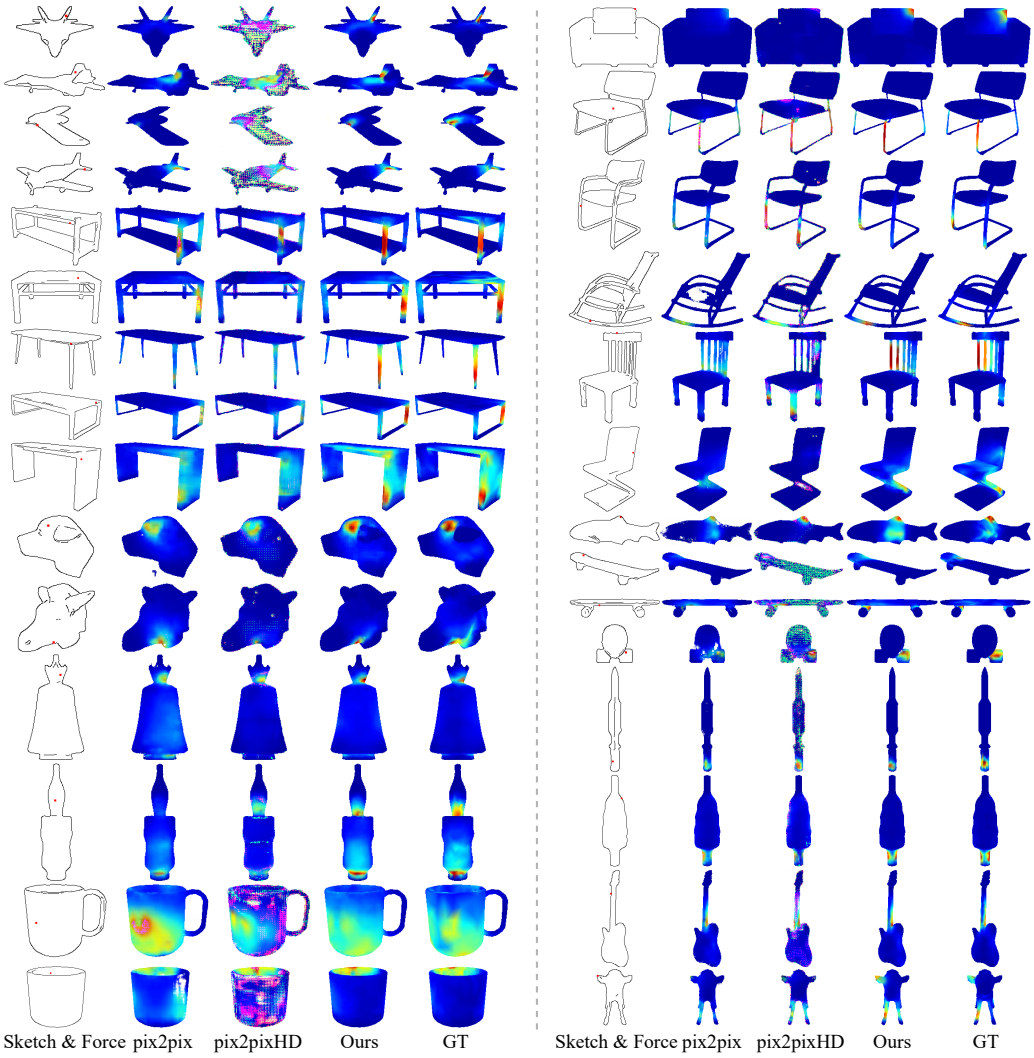


Fig. 4.7. Qualitative comparison of results generated by different methods of pix2pix, pix2pixHD, our method, and ground truth.

4.2.1 Qualitative Evaluation

In Figure 4.6, we illustrate a number of stress maps generated from input sketches under user-specified external forces using our method. It demonstrates the robustness of our method for input sketches with diverse geometry.

We also provided visual comparisons to pix2pix and pix2pixHD trained on our sketch-stress data in Figure 4.7. In comparison with the ground truth, it can be easily seen that our method achieves the best generation quality. Among the results generated by pix2pixHD, we see obvious high-frequency

noises. It is because the VGG loss pre-trained on the high-frequency natural images in pix2pixHD cannot well measure the feature difference of the low-frequency stress maps. Compared with the chair category, the airplane has less training data, making the performance of pix2pixHD significantly worse. pix2pix tends to lose the detail control of local regions during generation, especially surrounding the force locations. More specifically, pix2pix usually fails to synthesize correct colors for the higher-stress regions but only flattens or diffuses these regions with background low-frequency colors, as shown in airplanes, chairs, animal heads, tables, and vases in Figure 4.7. Please find more qualitative results in Appendix C.

4.2.2 Quantitative Evaluation

In the study of sketch-based structural analysis, we focus not only on the generation quality but also on the pixel-level stress accuracy of generated results, compared with the ground truth. We adopt four metrics to comprehensively evaluate the performance of different methods and compare their generated stress maps with the corresponding ground truth, namely, mean absolute error (MAE), F-Measure (FM), earth mover's distance (EMD), and Fréchet inception distance (FID). The former two metrics are used for the pixel-wise stress accuracy measurement, and the latter two are for the image quality evaluation. We report the quantitative evaluation results of the aforementioned methods on all eleven categories in Table 4.2. Note that we test the compared methods on the unseen data. For instance, the test data for Chair and Airplane contains 100 shapes and 60 shapes with $35K$ and $18K$ force samples, respectively.

From Table 4.2, we observe that our method yields overall better image generation quality while achieving significant improvements in pixel-level stress accuracy compared to the competitors. Compared to our approach, pix2pix struggles to accurately predict stress regions surrounding the user-specified force locations, resulting in compromised image quality (FID and EMD values) and pixel-wise stress accuracy (MAE and FM values). For pix2pixHD, its generated stress maps contain too many high-frequency noises (see the qualitative results in Figure 4.7), leading to its poor performance in metrics associated with image generation quality (EMD and

Category	Method	MAE ↓	EMD ↓	FID ↓	FM ↑
Chair	pix2pix	16.598	0.606	28.346	0.275
	pix2pixHD	**13.165**	1.250	75.852	0.186
	Ours	13.601	**0.374**	**15.083**	**0.412**
Airplane	pix2pix	2.128	0.106	7.380	0.438
	pix2pixHD	10.070	1.553	184.201	0.061
	Ours	**2.008**	**0.079**	**3.903**	**0.517**
Table	pix2pix	14.359	0.405	24.405	0.329
	pix2pixHD	16.626	0.701	55.440	0.204
	Ours	**14.282**	**0.321**	**10.421**	**0.434**
Vase	pix2pix	14.043	0.396	42.908	0.395
	pix2pixHD	11.0061	0.493	63.969	0.322
	Ours	**10.834**	**0.225**	**31.647**	**0.546**
Skateboard	pix2pix	5.574	0.253	51.883	0.168
	pix2pixHD	15.019	1.731	315.004	0.044
	Ours	**4.798**	**0.091**	**25.563**	**0.341**
Rocket	pix2pix	82.230	6.011	72.085	0.029
	pix2pixHD	54.663	4.848	215.600	0.025
	Ours	**2.136**	**0.344**	**54.561**	**0.414**
Guitar	pix2pix	12.452	0.830	34.000	0.242
	pix2pixHD	53.627	4.478	166.893	0.038
	Ours	**6.176**	**0.188**	**31.481**	**0.455**
Mug	pix2pix	41.016	1.321	43.275	0.321
	pix2pixHD	32.056	2.362	112.037	0.205
	Ours	**30.156**	**0.399**	**26.421**	**0.541**
Fourleg	pix2pix	12.719	0.704	76.126	0.263
	pix2pixHD	11.945	0.664	128.312	0.215
	Ours	**9.121**	**0.250**	**46.609**	**0.498**
Fish	pix2pix	117.787	6.891	124.301	0.098
	pix2pixHD	14.544	1.804	178.352	0.123
	Ours	**10.391**	**0.174**	**44.758**	**0.478**
AniHead	pix2pix	17.435	0.396	52.854	0.418
	pix2pixHD	13.819	0.650	117.058	0.290
	Ours	**12.150**	**0.266**	**27.837**	**0.506**

Tab. 4.2. Quantitative comparison of different methods in the sketch-based structural stress generation task on the eleven categories.

Category	Method	MAE ↓	EMD ↓	FID ↓	FM ↑
Chair	w/o Normal Branch	16.598	0.606	28.346	0.275
	w/o Shape Mask	14.432	0.390	14.718	0.402
	w/o Point Mask	13.862	0.378	15.092	**0.415**
	Full	**13.601**	**0.374**	**15.078**	0.412
Airplane	w/o Normal Branch	2.128	0.106	7.380	0.438
	w/o Shape Mask	2.329	0.126	4.456	0.457
	w/o Point Mask	2.161	**0.078**	4.227	0.514
	Full	**2.008**	0.079	**3.903**	**0.517**
Guitar	w/o Normal Branch	12.452	0.830	34.000	0.242
	w/o Shape Mask	6.597	0.253	34.762	0.409
	w/o Point Mask	6.797	0.192	32.982	0.438
	Full	**6.176**	**0.188**	**31.481**	**0.455**

Tab. 4.3. Quantitative comparison of the ablated methods of our approach on four categories with complex and diverse shape structures.

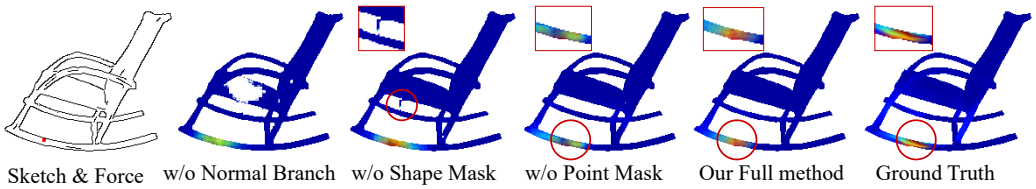


Fig. 4.8. Qualitative comparison of our ablated methods.

FID values) and pixel-wise stress accuracy (MAE and FM values). Although the value of the four metrics fluctuates across different categories due to the different data amounts of training data, our proposed method remains consistently superior to the competitors in the sketch-based structure analysis task.

4.2.3 Ablation Study

To evaluate the effects of the key components (namely, normal branch, shape mask, and point mask) of our approach, we present both the quantitative comparison of the ablation results in Table 4.3 and the qualitative comparison in Figure 4.8. Note that we report the quantitative comparison of our ablated methods on the chair, airplane, and guitar categories, which were chosen based on their highly varying levels of diversity, complexity, and number of training structures. From Table 4.3, we observe that removing the normal branch (the supervision on force directions and 2.5D

shape information), as expected, has a noticeable effect on the four metrics, leading to a significant drop in image generation quality and pixel-level accuracy (also see Figure 4.8). Without the shape mask, our approach's performance shows a heavy decrease in the four metrics. This mask plays a critical role in regularizing the shape boundary of the generated images while also reducing the outlier noises, such as the outlier defects below the chair seat in Figure 4.8. In terms of the point mask, we observed a slightly poorer performance on the three categories if this component was removed. As shown in Figure 4.8, our method tends to lose fine-grained control over regions surrounding the force point without the point mask.

In summary, our approach relies primarily on normal maps to perceive the underlying 3D shape structures and infer the surface stress. The shape mask is the second most important component in supervising the shape boundaries of generated stress maps. Finally, the point mask plays a vital role in guaranteeing the region consistency surrounding the force points. In our approach, the shape mask affects the stress map indirectly by regularizing the shape boundaries in normal maps directly while the point mask influences the final stress map directly during the generation process.

4.2.4 User Studies

To validate the practicality of our proposed sketch-based analysis tool, we design three user studies. The first study is the sketch-based weakness analysis to help users to analyze and summarize weak regions on their freely drawn sketches. The second study is the sketch-based structural refinement to help users to refine the structures of given sketches based on our computed stress maps. The third study examines the usefulness of our *Sketch2Stress* tool in assisting designers by providing them with structural stress awareness during the structure refinement process. Thus, we deploy a controlled task where designers are asked to refine the same problematic structures under the specified force conditions. During the trials, they are requested to refine the structure twice; first, without our *Sketch2Stress* tool, relying on their intuitions and experience, and second, with our tool. We invite 9 volunteers to participate in our user studies. Two of them were professional interior designers with years of drawing experience, and one

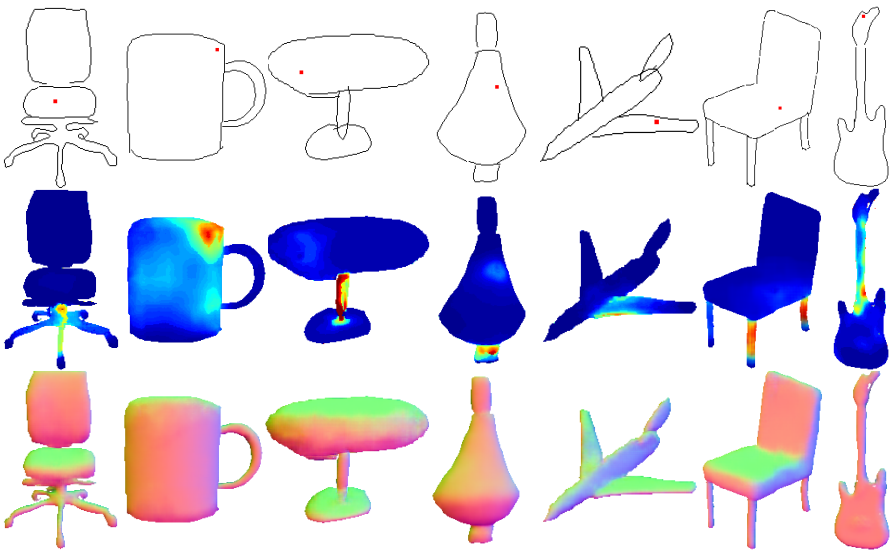


Fig. 4.9. User study of sketch-based weakness analysis. The top row is the user-drawn freehand sketches and the interested force points (red dots), the middle is the computed structure stress maps with our method, and the bottom is the inferred normal maps. The results demonstrate that our *Sketch2Stress* are robust to the common defects (poorly drawn curves, imperfect straight lines, detached chair back and guitar head, and the unclosed circular table) existing in input sketches and are able to generate consistent normal maps and stress effects.

was a new media artist, and the rest were postgraduate students aged 26 to 29 with no professional drawing skills.

Sketch-based Weakness Analysis In this study, we invite the participants to sketch their interested shapes among our prepared categories freely, with our interface, and let them figure out the fragile structural weakness by clicking on the sketched objects. After examining their own sketched objects, we require users to summarize their analysis process and answer one question “Which kinds of regions of a shape are the potential or possible weak regions?”. Their answers to this question are "joint regions, thin structures, non-straight legs, and single legs with variable thickness". We showcase representative freehand sketches with the user-assigned forces, the corresponding stress maps, and the inferred view-dependent 3D structures in Figure 4.9. Although the viewpoint of the freely sketched airplane in Figure 4.9 is quite different from training samples (Figure 4.6) in our dataset,

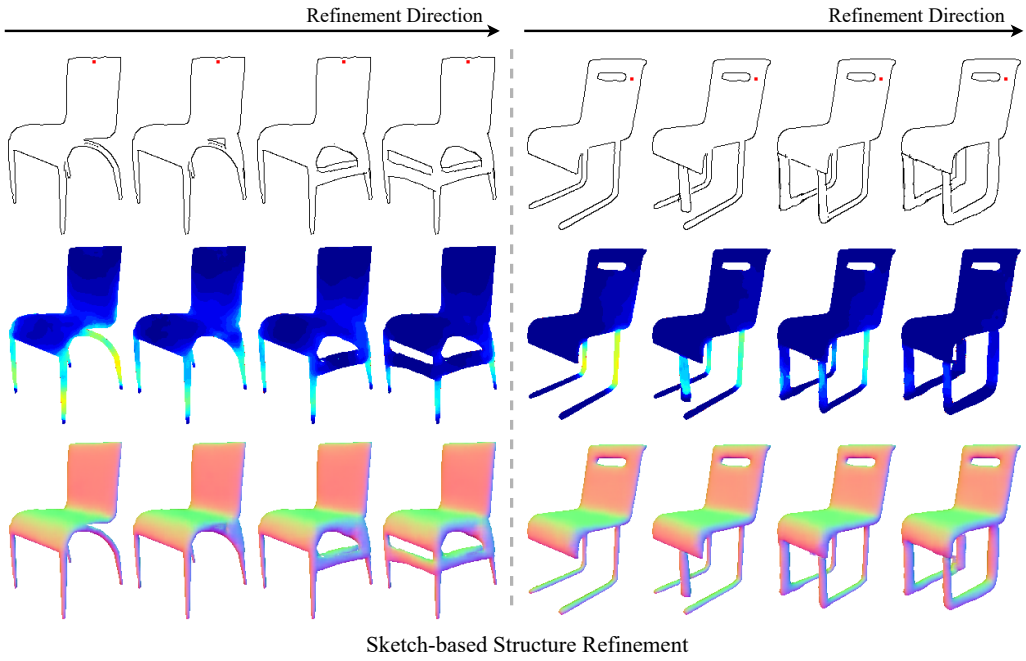


Fig. 4.10. Two examples from the user study of sketch-based structure refinement. We display the refinement directions of how users enhance the problematic structures and detail the intermediate refined sketches along the arrows of the refinement directions. The stress feedback and additional normal maps are side-placed with the sketches. Please zoom in to examine the details.

our *Sketch2Stress* is still able to infer a faithful view-dependent structure and a feasible stress map for the input.

Sketch-based Structure Refinement As the chair category exhibits the most complex shape structures, in this study, we invite all the participants to refine and enhance two initial chair structures with weak or problematic regions undertaking higher stress (see the regions with warmer and lighter colors in Figure 4.10) among the whole dataset. During the refinement process, we do not provide any suggestions and ask users to refine the structure based on the guidance of the computed stress map in our interface. We illustrate the refinement process of two representative sketch-based structure refinements from users, as shown in Figure 4.10. We also display the generated stress maps and normal maps besides the refined sketches at each time step in Figure 4.10.

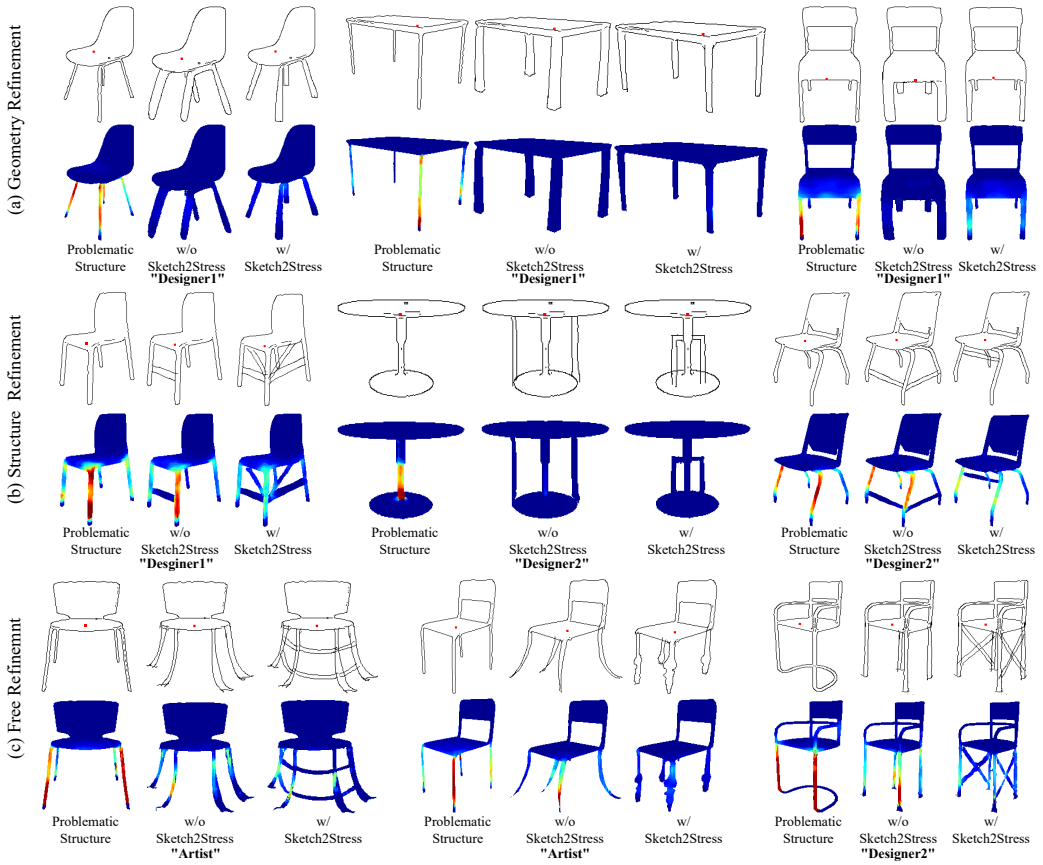


Fig. 4.11. User study of structure refinement with or without our *Sketch2Stress* tool. Each triplet contains a structurally problematic sketch under different force configurations (red dots on sketches), and the user-refined results without and with our tool, respectively. The corresponding stress maps are provided under the refined sketches.

Through the previous two studies, we show that both novice users and designers can easily identify the weak regions that sustain higher stress under the specified external forces with our proposed method. Our method also provides an effective way for users to interactively enhance their created shape structures by step-by-step refinement with our generated stress maps. However, people might be quite interested in how useful our *Sketch2Stress* tool is for these professional designers with years of design experience. So we further deploy the following controlled trials to answer this question.

Controlled trials: Structure Refinement with or without *Sketch2Stress* In this study, we invited two professional designers and one artist among the previous study participants and asked them to refine problematic sketch

structures as much as possible. In the beginning, we showed the participants both a problematic sketch and its stress map which indicates the weak regions of these fragile structures (see "problematic structures" in each triplet in Figure 4.11).

In conventional refinement tasks, designers are commonly requested to respect the original geometry (like the thickness for simplicity) and structure as faithfully as possible. Our sketch-based structure refinement task also follows the same rule. However, in our scenario, drastically changing the original structure by adding extra structures also works for improving the structural soundness, so we further customize and define our own requirements for different types of refinements as shown in Figure 4.11: (a) Geometry refinement: participants are only allowed to adjust the thickness of the fragile parts to improve the problematic structure without changing the original structures. (b) Structure refinement: participants are only allowed to change structures but not modify the thickness of the original structures. (c) Free refinement: participants are allowed to edit both the geometry and the entire structures.

In the first trial, guided by the requirements stated previously, the participants were asked to heal these weak regions with their own learned knowledge, design experience, and intuitions but without our *Sketch2Stress*. Although three participants used our *Sketch2Stress* tool in the previous two user studies and learned where might be the potential regions and how the problematic structures were iteratively improved with our tool, we are still interested to know how well the participants could use the learned knowledge and fix the novel problematic cases by themselves. During this process, the participants were allowed to edit the problematic sketch structures multiple times following the different refinement requirements until they were satisfied. Note that we did not update the stress maps during this refinement process. The final refined sketches and their corresponding stress maps of the first trial can be seen in Figure 4.11 ("w/o Sketch2Stress" in each triplet). In the second trial, we allowed the participants to refine the problematic structure obeying three refinement requirements with our *Sketch2Stress* tool. During this process, we provided an instant response in the form of a stress map after each editing operation. The final sketch refinements and their corresponding stress maps from the

second trial can be seen in Figure 4.11 ("w/ Sketch2Stress"). Through the controlled trials, we found that relying solely on designers' experience without our *Sketch2Stress* tool could only mildly relieve the weak regions' stress or sometimes worsen the situation. For example, in geometry refinement (Figure 4.11 (a)), the designers usually attempted to thicken these thin legs as much as possible. However, this is not the optimal way to strengthen problematic regions meanwhile not modifying the original geometry too much. While our *Sketch2Stress* tool can help the designers and the artist to iteratively adjust and obtain a more suitable, even optimal thickness by giving them instant stress feedback after each modification. Also, as illustrated in Figure 4.11 (b) and (c), our *Sketch2Stress* informs the designers and the artist where (potential fragile regions), which auxiliary strategies (thickening, adding extra structures), and how effective their modifications by showing them the instant stress responses after their edition operations. Using *Sketch2Stress*, all the participants successfully refined the problematic sketch structures to better versions. The practicality and usefulness of our *Sketch2Stress* tool received high appreciation from the designers and artist. All three participants spoke highly of our designed tool for helping them quickly locate the weak regions and inform them of the vivid and instant stress responses after every editing operation. Before our user studies, they did not have much experience performing structural analysis and refinement in the sketching phase.

4.2.5 Structural Analysis on Real Product Sketches

As product designers extensively use sketches in their creation and communication, to demonstrate the powerful feature of our method in aiding sketch-based structural analysis, we further apply our *Sketch2Stress* method to real product design sketches in OpenSketch [38]. We first leverage the 3D objects in OpenSketch to render the 2D sketch-to-stress data, (as described in Subsection 4.1.1) and then train our network on the projected synthetic data. As shown in Figure 4.12, with clean sketches and the user-assigned forces, our method is able to generate feasible and high-quality structural stress maps for product sketches. With the aid of our method,

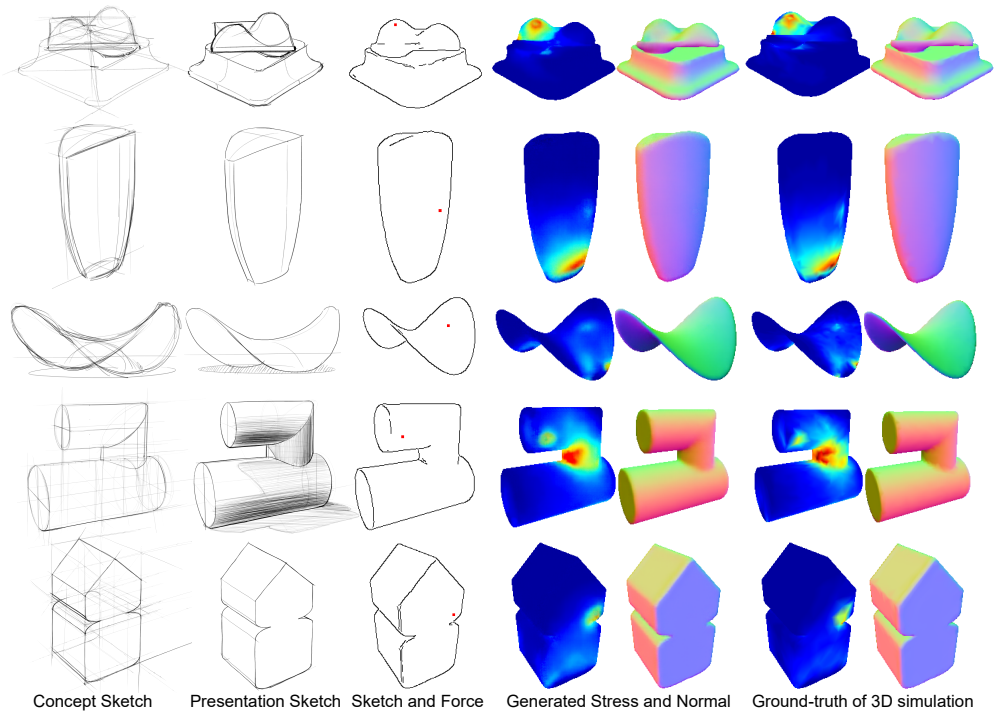


Fig. 4.12. Our *Sketch2Stress* method applied to the OpenSketch dataset. The concept and presentation sketches of the bump, shampoo bottle, and potato chip (in the first, second, and third rows) are from "Professional1" while the bottom two rows of the tube and the house are from "Professional5" and "Professional6" in the OpenSketch dataset. Please zoom in to examine the details.

designers will have more opportunities to check and refine the structural weaknesses of their ideal products in advance at the sketching stage, following the same operations in Subsection 4.2.4. Furthermore, with our method, designers will have a larger design space by incorporating external physical factors in the form of different force configurations.

4.3 Discussion

We have introduced the novel problem of sketch-based structural analysis, where we constrain the external forces to variables with the same magnitude but different locations and opposite-normal directions. We further present a two-branch generator to synthesize feasible structural stress maps by considering the sketches' geometry and force variables simultaneously.

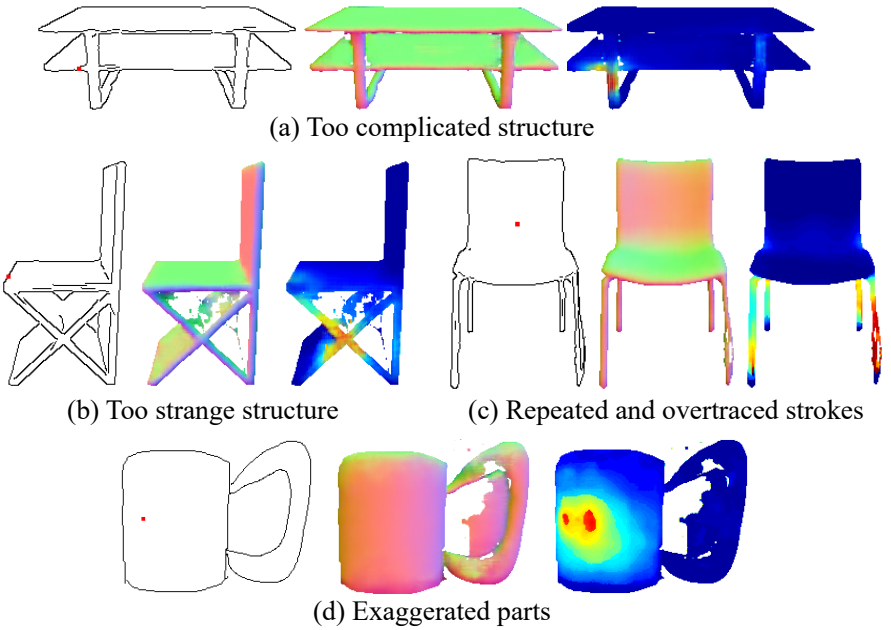


Fig. 4.13. Fail cases. Our inference model might fail when the sketched structures are too complex or too strange from the observed structures.

We find that usually, the long, thin, tilt, and joint regions tend to suffer higher stress, and shapes with such regions are weaker than those shapes without them.

While the work we proposed provides an efficient approach for sketch-based structural analysis, our method has some limitations. First, our method cannot synthesize the stress effects of forces that are not in opposite-normal directions. Second, the force magnitude in our problem is set to a fixed value, which makes it challenging to analyze the stress effects of external forces with dynamic values. These two limitations are inherited from the method [126] we adopted for synthesizing the training data. Hence, further advances in new structural stress analysis solutions on 3D models could also boost our approach. Third, only having one sketch provides limited information to indicate material properties. Fourth, our approach cannot take as input multi-forces at different directions since the combination of multiple forces requires an extra module to process carefully, not simply recording the mapping between input multiple forces and the output corresponding structural stress effect. In the future, we aim to further explore the proper representations and definitions for materials and external multiple forces in the generative process. Fifth, due to the constraints

of learning-based approaches, our well-trained *Sketch2stress* also cannot jump out of the bound of the empirical distribution of the observed data and might fail to infer reasonable stress maps and faithful underlying structures for input sketches with too complicated and strange structures, repeated and over-traced strokes, or exaggerated part geometries. As shown in Figure 4.13 (a) and (b), some defects can be observed in the generated normal maps and synthesized structural stress maps of the double-layer table and "X"-leg chair. Also, the over-traced strokes and the exaggerated parts will lead to failures with our approach, i.e., the holes and the mismatched mug-handle in (Figure 4.13 (c) and (d)). For the small flaws in generated stress maps and normal maps in Figure 4.13 (c) and (d), they could be fixed by refining the normal map (similar to [118]). Lastly, in our user studies, users needed to draw their refinements on our *Sketch2Stress* interface repeatedly in a trial-and-error to obtain the final sound structures. Ideally, a more intuitive interface will further guide users to fix the problematic parts better, such as by providing a slider for users to adjust the thickness.

5

Learning Local Sketch Descriptors for Multi-view Correspondence

In previous chapters, we present two approaches to beautify both the part geometry and entire structure of the sketch of a man-made object (Chapter 3) and analyze the structural stress effects of a designed sketch prototype under the user-specified external forces (Chapter 4). The above two approaches mainly focus on sketched objects' shape and structure analysis from a single view. As discussed in Section 1.3, there is an inherent ambiguity in single-view sketches due to the occlusion that makes them unable to convey the full overviews of the underlying 3D shapes. To faithfully represent the user's desired 3D shapes in mind, in this chapter, we explore the multi-view sketches and study the problem of inferring the semantic correspondence of multi-view sketches. In this chapter, we introduce the methodology of our local sketch descriptors aimed for inferring multi-view correspondence, *SketchDesc*, in Section 3.1. Quantitative and qualitative evaluations are showed in Section 3.2. We present two applications based on *SketchDesc* in Section 5.3, summarize this method and discuss the limitations in Section 5.4.

5.1 Methodology

Terminology In this chapter, we differentiate between the word "points" for shapes, sketch images, and sketch lines as follows. We refer to the

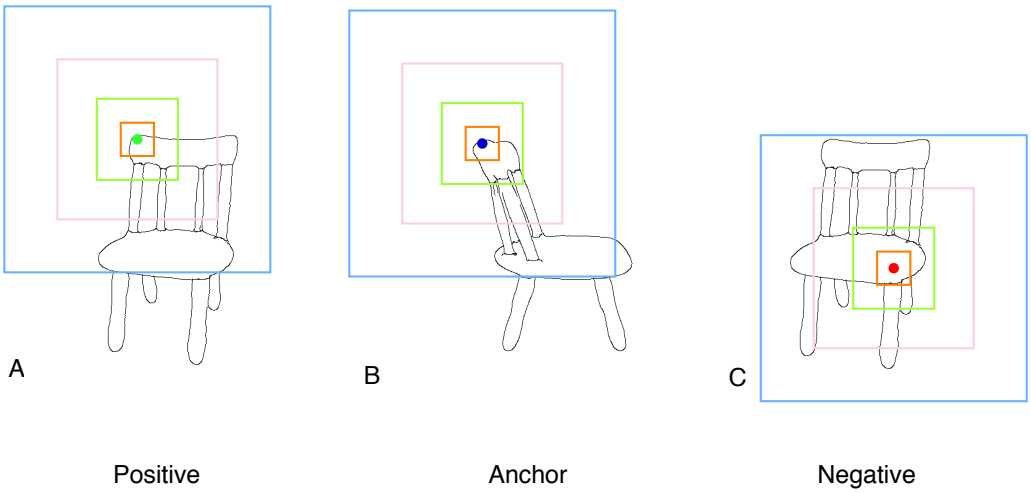


Fig. 5.1. Illustration of our multi-scale patch-based representation. Given a 480×480 sketch input and a pixel of interest, we use multi-scale (32×32, 64×64, 128×128, and 256×256) patches to capture the neighborhood of the pixel. The positive, anchor and negative patches are formed as a training triplet.

points on the surface of a 3D shape as *vertices*, the points on a 2D sketch image as *pixels*, and the points exactly on the sketch lines as *points*.

For input sketches represented as rasterized images, a key problem to semantic correspondence learning is to measure the difference between a pair of pixels in a semantic way. This is essentially a metric function learning problem where a pair of corresponding pixels has a smaller distance than a pair of non-corresponding pixels in the metric space. Formally, it can be expressed as follows:

$$D_m\langle p_{c1}, p_{c2}\rangle < D_m\langle p_{c1}, \widetilde{p}_{nc}\rangle, \tag{5.1}$$

where $D_m\langle\cdot,\cdot\rangle$ is a metric function, and p_{c1} and p_{c2} represent corresponding pixels from different sketches (e.g., the pair of anchor and positive pixels in Figure 5.1) while p_{c1} and $\widetilde{p}_{nc}$ are a pair of non-corresponding pixels (e.g., the pair of anchor and negative pixels in Figure 5.1).

We follow [122] to build the metric function D_m by learning a sketch descriptor (*SketchDesc*) with a triplet loss function. Basically we optimize the loss function in Equation 5.2, which minimizes the distance between pairs of corresponding pixels and maximizes the distance between pairs of

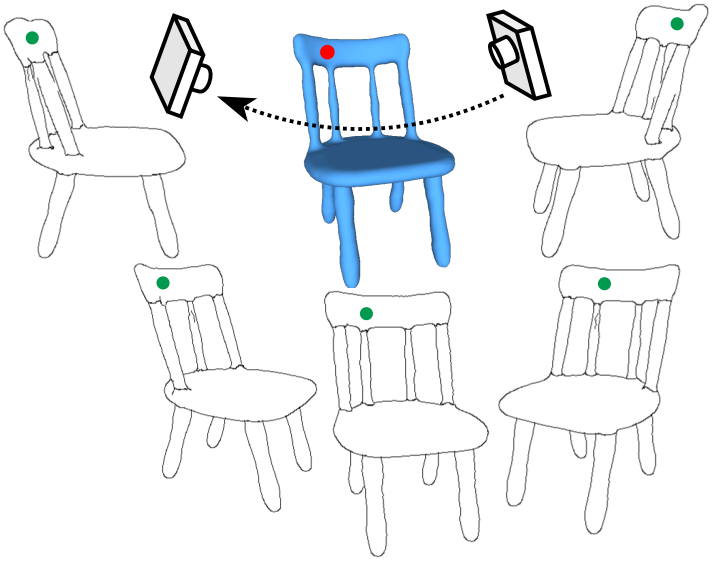


Fig. 5.2. Illustration of synthesized multi-view sketches (of size 480 × 480), with corresponding pixels (in green) projected from the same vertex (in red) on a 3D shape.

non-corresponding pixels. Since sketch images are rather sparse, we adopt a multi-scale patch-based representation, that is, multi-scale patches centered at a pixel of interest in a sketch. We resort to deep CNNs, which own the superior capability of learning discriminative visual features from sufficient training data. As illustrated in Figure 5.4, our designed network (Section 5.1.2) takes a multi-scale patch as input and outputs a 128-D distinctive descriptor.

To train the network, we first synthesize line drawings of a 3D shape from different viewpoints as multi-view sketches. We then generate the ground-truth correspondences by first uniformly sampling points on the 3D shape and then projecting them to the corresponding multiple views. We will discuss the process of data preparation in more detail in the next section.

5.1.1 Data Preparation

Multi-view Sketches with Ground-truth Correspondences We follow a similar strategy in [65, 139, 115] to synthesize sketches from 3D shapes.

Specifically, we first render a 3D shape with the aligned upright orientation to a normal map under a specific viewpoint, and then extract an edge map from the normal map using Canny edge detection. We adopt this approach instead of the commonly used suggestive contours [23], because the latter is suited for high-quality 3D meshes and cannot generate satisfactory contours from poorly-triangulated meshes (e.g., airplanes and rifles in the Structure Recovery dataset [101]). Hidden lines of the edge detection results are removed. In our implementation, each sketch is resized to a 480×480 image. As mentioned in [30], most humans are not faithful artists and create sketches in a casual and random way. Unlike [45, 81] using limited views, to better accommodate the shape and viewpoint variations in freehand sketches, we sample viewpoints on the upper unit viewing hemisphere (in an elevation angle of 15$\sim$45 degrees) at every 15 or 30 degrees in the azimuth angle for rendering each 3D model.

By projecting each vertex to the corresponding views, we naturally construct ground-truth correspondences (with the projections from the same vertices) among synthesized multi-view sketches, as illustrated in Figure 5.2. We do not consider hidden vertex projections (invisible under depth testing). If the projections of a vertex are visible only in less than two different views, this vertex is not considered in ground-truth correspondences. Following this strategy, we can generate from 28K to 60K ground-truth pairwise correspondences from each 3D shape. Our synthesized correspondence dataset for training and testing are derived from 6,852 multi-view sketches distributed over 18 shape categories of existing shape datasets [101, 15, 146] (Section 5.2).

Multi-scale Patch Representation We represent each visible projection of a 3D vertex on sketch images with a patch-based representation centered at the corresponding pixel to capture the distinctive neighboring structures (Figure 5.1). To better handle the sparsity and the lack of texture information in sketches, we adopt a multi-scale strategy. Given a 480×480 sketch image, we employ a four-scale representation (i.e., 32×32, 64×64, 128×128, and 256×256) for a pixel, as illustrated in Figures 5.1 and 5.3.

The multi-scale patch-based representation allows us to sample ground-truth correspondences inside sketched objects, and not necessarily on sketch

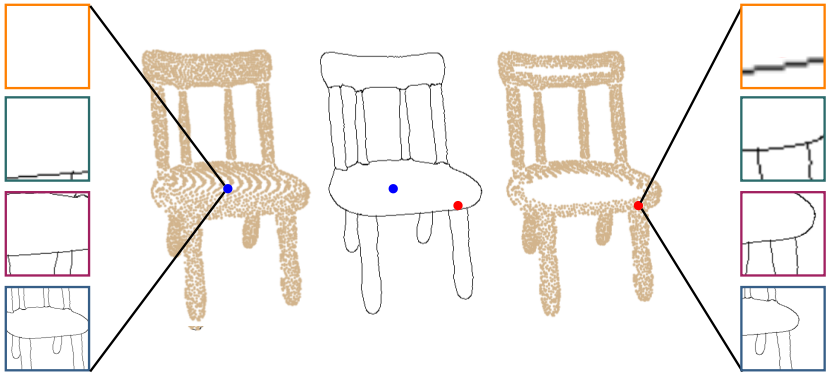


Fig. 5.3. An illustration of two sampling mechanisms (Left: OR-sampling; Right: AND-sampling) for training data. OR-sampling generates more challenging data (blue) than AND sampling (red).

lines. This significantly increases the number of ground-truth correspondences for training. However, we do require valid information existing in a multi-scale patch representation. We have tried two sampling mechanisms:
1) *OR-sampling*: a multi-scale patch is valid if the patch is non-empty at any of the scales (Figure 5.3 left);
2) *AND-sampling*: a multi-scale is valid if the patch is non-empty at every scale (Figure 5.3 right).
The former generates valid multi-scale patches at almost every visible vertex projection, since the patch at scale 256×256 is often non-empty given its relatively large scale. In contrast, the AND-sampling generates valid multi-scale patches only near to sketch lines. We will compare the performance of these two sampling mechanisms in Section 5.2.

Before feeding these patches into our multi-branch network, we rescale all the patches to 32×32 (i.e., the smallest scale) by bilinear interpolation (Figure 5.4). Below we describe our network architecture in more detail.

5.1.2 Network Architecture

We design a network architecture to learn local descriptors for measuring the semantic distance between a pair of pixels in multi-view sketches. As illustrated in Figure 5.4, our network has four branches to process the set

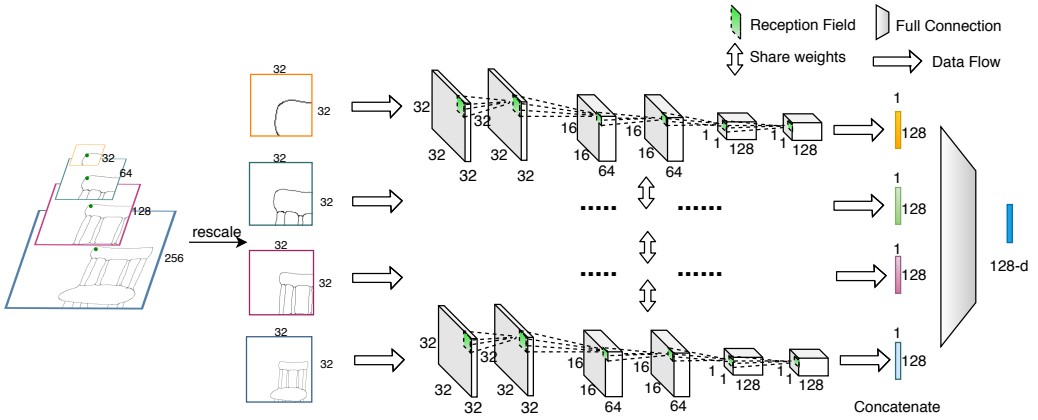


Fig. 5.4. The architecture of *SketchDesc*-Net. Our input is a four-scale patch pyramid (32×32, 64×64, 128×128, 256×256) centered at a pixel of interest on a sketch, with each scale rescaled to 32×32. Given the multi-scale patches, we design a multi-branch framework with shared weights to take as input these rescaled patches. The dashed lines represent the data flow from an input patch to an output descriptor. For the kernel size and stride in our network, we adopt the same settings as [122]. Finally, the output as a 128-D descriptor embeds features from the four scales by concatenation and full connection operations.

of four scaled input patches. The four branches share the same architecture and weights in the whole learning process. Each branch receives a 32 $\times$ 32 patch and outputs a 128 $\times$ 1 (i.e., 128-D) feature vector which is then further fused at the concatenation layer and the final fully-connected layer. Note that due to the shared weights among the branches, the multi-branch structure does not increase the number of parameters. Our network produces a 128-D descriptor as output, which will be later used for sketch correspondence and pixel-wise retrieval in Section 5.2.

5.1.3 Objective Function

To train our network, we employ the random sampling strategy of [122] to assemble the triplets (Figure 5.1) in a training batch. We define the output descriptors of a triplet as (f_a, f_p, f_n), where f_a is the descriptor vector of an anchor pixel in one sketch, and f_p and f_n represent the descriptors of the corresponding pixel (corresponding to the same vertex in a 3D shape as the anchor pixel) and a non-corresponding pixel. The non-corresponding pixel

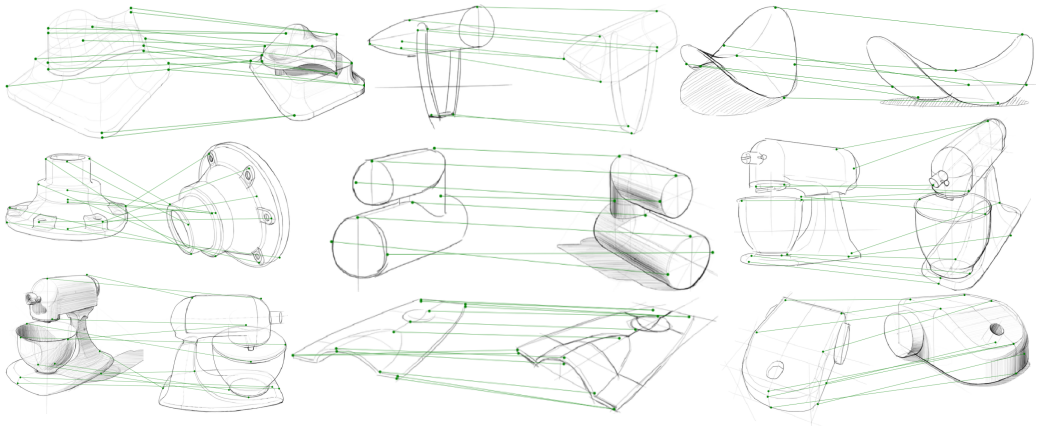


Fig. 5.5. Sketch correspondence in the OpenSketch dataset computed with *SketchDesc* descriptors. Note that even with limited ground-truth correspondence in the training set of OpenSketch, *SketchDesc* can still establish a robust correspondence for the multi-view sketches.

can be selected from either the same sketch or in the other views. With the descriptor triplets, we adopt the triplet loss [99] to train the network. The triplet loss, given in Equation 5.2, aims to pull closer the distance between a pair of corresponding pixels (f_a, f_p) and push away the distance between a pair of non-corresponding pixels (f_a, f_n) in the metric space.

$$L_{triplet} = \frac{1}{n}\Sigma_{i=1}^{n} \max\left(0, d\left(f_{a_i}, f_{p_i}\right) - d\left(f_{a_i}, f_{n_i}\right) + m\right), \tag{5.2}$$

where n is the number of triplets in a training batch, $(f_{a_i}, f_{p_i}, f_{n_i})$ denotes the i-th triplet, $d(\cdot,\cdot)$ measures the Euclidean distance given two descriptors, and the margin m is set to 1.0 in our experiments.

5.2 Experiments

We conducted extensive experiments on two sketch datasets: our synthesized multi-view sketch dataset and the OpenSketch dataset[38]. For our synthesized dataset (Figure 5.9), we utilize three existing 3D shape repositories: the Structure-Recovery database [101], Princeton Segmentation Benchmark (PSB) [15], and ShapeNet [146]. Table 5.1 shows some information about the three shape repositories, including the selected cate-

	Structure-Recovery						PSB									ShapeNet								
Category	Airplane	Bycicle	Chair	Fourleg	Human	Rifle	Airplane	Bust	Chair	Cup	Fish	Hand	Human	Octopus	Plier	Airplane	Bag	Cap	Car	Chair	Earphone	Mug	Pistol
Shapes	54	12	27	16	16	25	20	20	20	20	20	20	20	20	20	38	35	35	39	23	28	34	25
Views	11	12	12	12	12	12	11	12	12	11	12	11	12	11	11	11	12	12	12	12	12	12	12

Tab. 5.1. Object categories and the number of shapes and views used in our dataset.

gories, the number of 3D shapes per category, and the number of views used per category. The shapes in each category were manually selected based on the criterion of increasing shape diversity and decreasing the redundant and repeated shapes. For the OpenSketch dataset, it has around 400 sketches of 12 man-made objects, which were drawn by professional product designers. Different from the synthesized sketches, the OpenSketch data contains abundant annotations of additional shadings, skeletons and auxiliary lines (shown in Figure 1.8 (bottom) and Figure 5.5).

In addition to the hand-drawn sketches in OpenSketch, we also performed evaluation on a set of more freely-drawn sketches. Several participants were invited to create freehand sketches on a touchscreen, after observing a 3D shape for a fixed amount of time. Each participant was given three salient views of each shape that ordinary users would be familiar with. Figure 1.8 (two middle rows) shows some representative results of sketch correspondence (i.e., chair, bicycle, and fourleg). Note that to avoid visual clutter, for each freehand sketch pair, we randomly selected 20$\sim$50 pairs of matched correspondences computed by nearest neighbor search with *SketchDesc* (see Section 5.2.1). In the Appendix D, we provide more results of the computed correspondences among hand-drawn multi-view sketches.

Implementation Details We implemented our network with the PyTorch [89] framework and used the Xavier initialization [35]. We train our network for each object category with a data splitting ratio of 8 : 1 : 1 (training : validation : testing). All multi-view sketches are rendered to the size of 480×480. The batch size is set to 64. Our network is trained on an NVIDIA RTX 2080Ti GPU and optimized by the Adam [55] optimizer ($\beta_1 = 0.9$ and

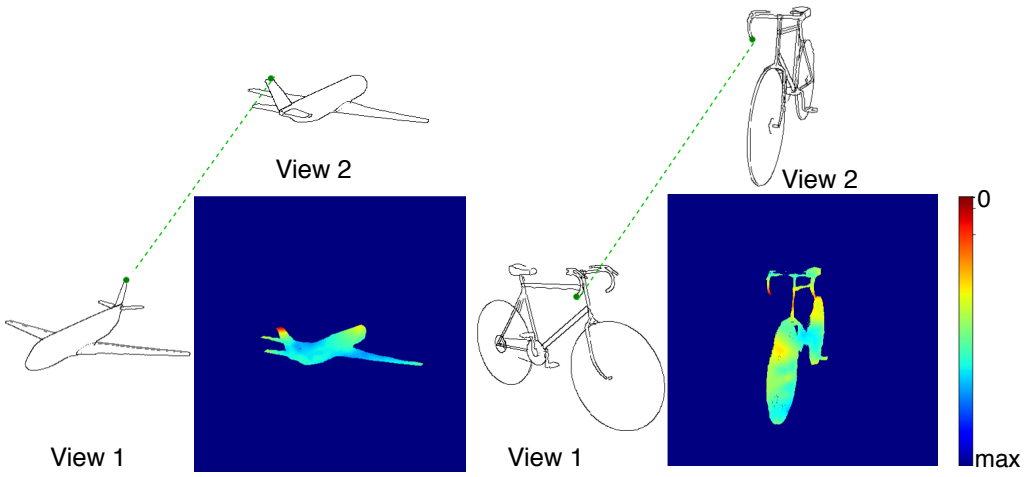


Fig. 5.6. For a given pixel inside a sketched object under View 1, we find a corresponding point in the other sketch under View 2 by computing a distance map through our learned descriptor.

$\beta_2 = 0.999$) with a learning rate of $1e^{-3}$. The number of iteration epochs in our experiments is set to 100.

5.2.1 Sketch Correspondence

In this task, we validate the performance of the learned descriptors in finding corresponding pixels in pairs of multi-view sketches in the testing set. Given a pair of testing sketches, for each pixel in one sketch, we compute its distances to all pixels in the other sketch (see distance visualization in Figure 5.6). We consider it as a successful matching if the pixel with the shortest distance is no further than 16 pixels (half of the smallest patch scale) away from the ground-truth pixel. Note that not all pixels in a sketch image have ground-truth correspondences (only among those projected).

To clarify the dissimilarity between the training set and the testing set, we designed a retrieval-based baseline (illustrated in Figure 5.7) in the task of sketch correspondence.

For a given pixel on one sketch in View 1 of a testing pair, we perform the pixel matching operation (by nearest neighbor search) to get the most similar pixel among all the training sketches under View 1. Note that the

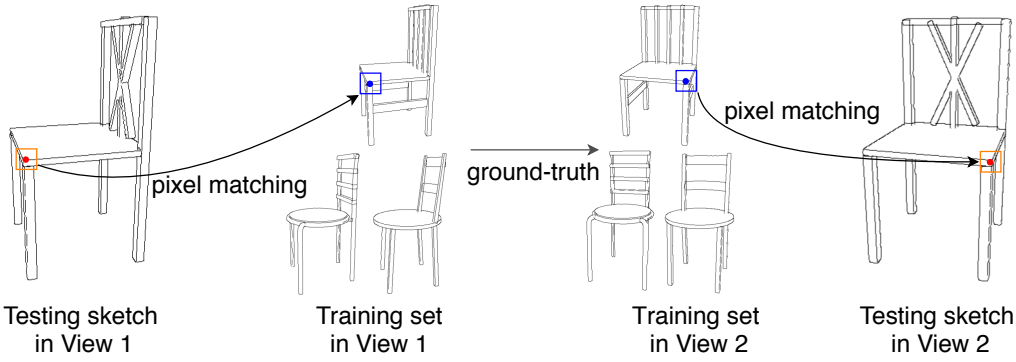


Fig. 5.7. The pipeline of a retrieval-based baseline, which directly utilizes the ground-truth correspondence in the training set. For illustration, we show the top-3 sketches retrieved from the training set by pixel matching.

pixel matching operation takes as input the patches (32×32) centered on the pixels. With the matched pixel, we utilize the ground-truth cross-view correspondence in the training data to find its corresponding pixel in View 2 (i.e., the same view as the other sketch in the testing sketch pair). Finally, we perform the pixel matching again to find the most similar pixel on the testing sketch in View 2. Note that we employ the conventional image matching method [22] for the pixel matching operation.

The performance of our approach and other competing methods on several representative categories is reported in Table 5.2. We report the averaged success rate as the correspondence accuracy. We observe that our network outperforms its learning-based and retrieval-based competitors. Due to the lack of texture in sketch patches, the frameworks designed for image patches (i.e., L2-Net, HardNet, and SOSNet) cannot learn effective descriptors to match the corresponding pixels in multi-view sketches. Our descriptor also surpasses those based on LeNet and AlexNet-VP. Figure 5.8 gives some qualitative comparisons between different approaches. Since 32×32 patches are required to be non-empty, the testing pixels are near sketch lines. We observe that the descriptors learned by LeNet, HardNet, L2-Net, and SOSNet are less discriminative, and are thus not effective in finding corresponding pixels among multi-view sketches. AlexNet-VP uses the max-pooling layer, making it difficult to focus on local details for robust descriptors, as discussed in [78]. As a consequence, AlexNet-VP produces more ambiguous distance maps (e.g., the chair example in Figure

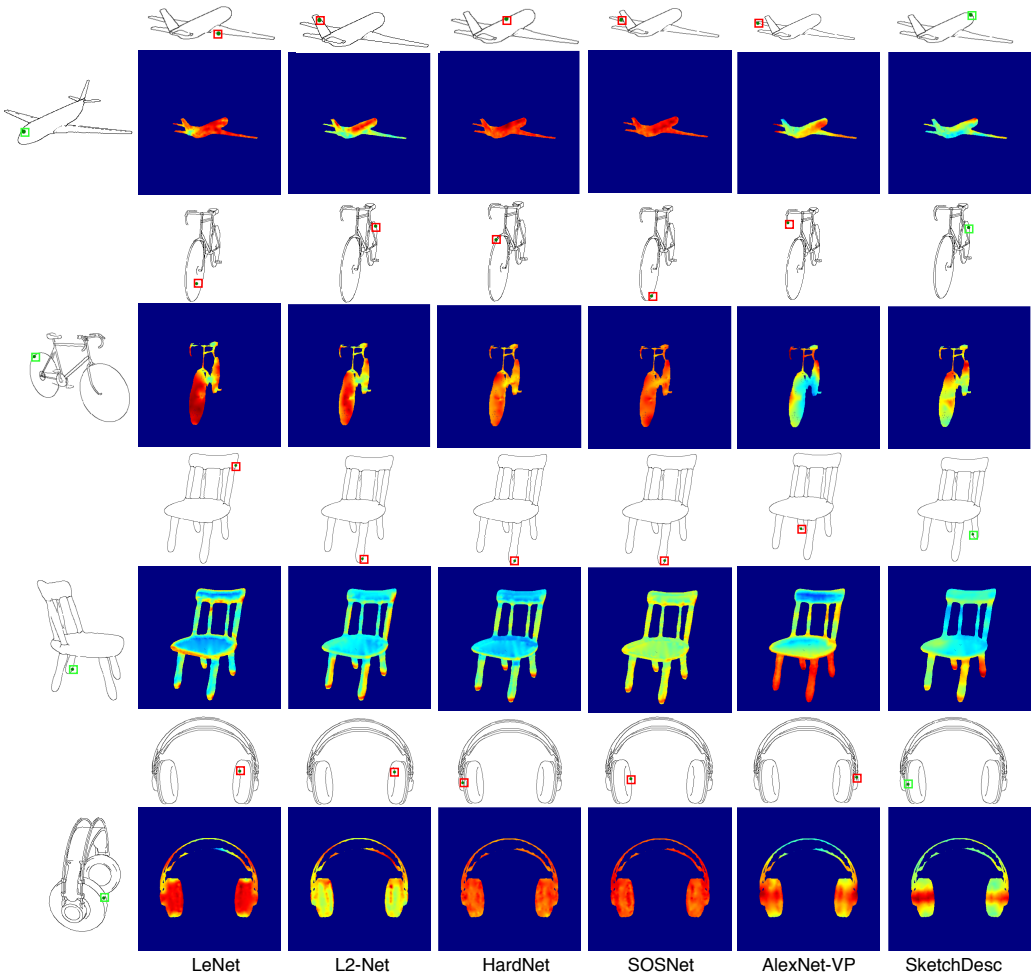


Fig. 5.8. Sketch correspondence results by computing the distance maps with different approaches. Correct and wrong matching results are marked as green and red boxes, respectively.

		base.	LeNet	L2Net	HardNet	SOSNet	AlexNet-VP	SketchDesc
Structure-Recovery	Airplane	0.14	0.18	0.24	0.21	0.19	0.36	**0.54**
	Bicycle	0.28	0.38	0.39	0.42	0.40	0.66	**0.71**
	Chair	0.27	0.40	0.40	0.39	0.39	0.51	**0.56**
	Fourleg	0.27	0.29	0.34	0.32	0.28	0.52	**0.66**
	Human	0.21	0.28	0.38	0.40	0.35	0.64	**0.73**
	Rifle	0.27	0.43	0.52	0.53	0.50	0.67	**0.76**
	avg_acc	0.24	0.33	0.38	0.38	0.35	0.56	**0.66**
PSB	Airplane	0.07	0.18	0.23	0.13	0.10	0.52	**0.72**
	Bust	0.20	0.33	0.22	0.23	0.20	0.31	**0.45**
	Chair	0.22	0.29	0.27	0.24	0.26	0.44	**0.60**
	Cup	0.26	0.22	0.25	0.19	0.22	0.35	**0.57**
	Fish	0.29	0.30	0.36	0.35	0.31	0.63	**0.64**
	Human	0.20	0.35	0.35	0.38	0.34	0.54	**0.79**
	Octopus	0.08	0.10	0.11	0.09	0.07	0.24	**0.28**
	Plier	0.09	0.10	0.09	0.06	0.14	0.57	**0.73**
	avg_acc	0.18	0.23	0.24	0.21	0.21	0.45	**0.62**
ShapeNet	Airplane	0.21	0.39	0.44	0.40	0.35	0.67	**0.69**
	Bag	0.11	0.17	0.14	0.15	0.14	0.31	**0.38**
	Cap	0.17	0.21	0.25	0.26	0.22	0.55	**0.75**
	Car	0.13	0.17	0.21	0.21	0.21	0.55	**0.73**
	Chair	0.16	0.19	0.20	0.18	0.19	0.42	**0.47**
	Earphone	0.12	0.15	0.18	0.16	0.14	0.25	**0.42**
	Mug	0.21	0.19	0.20	0.17	0.17	0.32	**0.39**
	Pistol	0.23	0.30	0.34	0.33	0.34	0.66	**0.71**
	avg_acc	0.17	0.22	0.25	0.23	0.22	0.47	**0.57**

Tab. 5.2. Sketch correspondence accuracy (i.e., the average success rate) for the different methods. The best results in each object category are in boldface.

5.8) compared to our method. The retrieval-based baseline employs the ground-truth from the training set, but it still fails to build reliable correspondences for multi-view sketches. In fact, the retrieval-based baseline has the worst performance. It is mainly because of the incapability of the image-based descriptor [22] on dealing with the highly similar local patches with limited textures in multi-view sketches, and also because of the large disparity between multi-view sketches in the training set and testing set (Figure 5.9). In addition, it is challenging to use this baseline in practice with hand-drawn sketches, due to the required view prior and its large running time to infer the correspondences for a pair of multi-view sketches (3.15 minutes vs 8.76 seconds by our method on average).

Figure 1.8 shows successful matchings randomly selected from all successful matchings on the synthesized multi-view sketches (top) and the hand-drawn sketches (middle). In Figures 1.8 (bottom) and 5.5, we also show

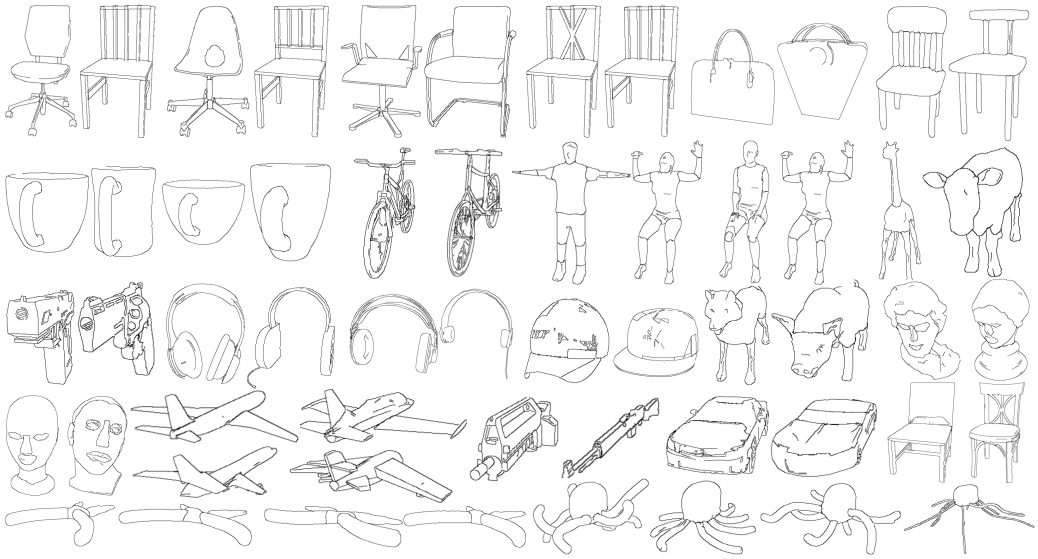


Fig. 5.9. The pairs of a testing sketch (left) and its most similar training sketch (right) retrieved from the training set under the same view, utilizing the image matching method [22].

that *SketchDesc* can be further utilized to infer some correspondences of multi-view sketches in the OpenSektch dataset. Due to the limited ground-truth correspondence in OpenSketch, we show all the successful matchings. More correspondence results (successful and unsuccessful matchings) of multi-view sketches can be found in the Appendix D.

5.2.2 Multi-view Pixel-wise Retrieval

We further design a multi-view corresponding pixel retrieval task. Given multi-view sketches synthesized from multiple shapes, we uniformly sample a set of pixels (1000∼1600 pixels) on one sketch and search in the other sketches for the corresponding pixels which are from the same shape (in different views). We use the descriptors computed from the compared networks as queries and adopt the Mean Average Precision (MAP) [4] to measure the retrieval performance. Let $y = (y_1, \ldots, y_n) \in \{-1, +1\}^n$ be the labels of a ranked list of pixels returned for a pixel query, with -1 and $+1$ indicating negative and positive match, respectively. Then the *precision*

		LeNet	L2-Net	HardNet	SOSNet	AlexNet-VP	SketchDesc
Structure-Recovery	Airplane	0.33	0.42	0.45	0.40	0.45	**0.68**
	Bicycle	0.37	0.40	0.44	0.39	0.60	**0.84**
	Chair	0.39	0.45	0.42	0.42	0.70	**0.82**
	Fourleg	0.33	0.45	0.41	0.30	0.75	**0.82**
	Human	0.20	0.44	0.40	0.27	0.74	**0.92**
	Rifle	0.56	0.56	0.57	0.54	0.76	**0.83**
	avg_map	0.36	0.45	0.45	0.39	0.67	**0.82**
PSB	Airplane	0.26	0.26	0.11	0.11	0.42	**0.68**
	Bust	0.01	0.29	0.29	0.26	0.51	**0.64**
	Chair	0.33	0.48	0.42	0.46	0.75	**0.86**
	Cup	0.05	0.08	0.04	0.06	0.22	**0.50**
	Fish	0.19	0.28	0.23	0.21	0.58	**0.73**
	Human	0.13	0.53	0.52	0.46	0.79	**0.93**
	Hand	0.08	0.37	0.37	0.28	0.52	**0.64**
	Octopus	0.38	0.29	0.29	0.46	0.67	**0.71**
	Plier	0.18	0.36	0.21	0.18	0.79	**0.86**
	avg_map	0.18	0.33	0.28	0.27	0.58	**0.73**
ShapeNet	Airplane	0.14	0.22	0.15	0.09	0.37	**0.55**
	Bag	0.17	0.20	0.15	0.14	0.53	**0.69**
	Cap	0.16	0.24	0.19	0.17	0.59	**0.71**
	Car	0.20	0.19	0.22	0.22	0.56	**0.67**
	Chair	0.11	0.15	0.14	0.140	0.51	**0.67**
	Earphone	0.25	0.30	0.25	0.20	0.49	**0.83**
	Mug	0.05	0.06	0.06	0.08	0.28	**0.34**
	Pistol	0.29	0.35	0.35	0.36	0.69	**0.82**
	avg_map	0.17	0.21	0.19	0.17	0.50	**0.66**

Tab. 5.3. Pixel-wise retrieval performance for the different methods.

Methods	Structure-Recovery&ShapeNet / PSB	PSB&ShapeNet / Structure-Recovery	Structure-Recovery&PSB / ShapeNet
LeNet	0.27	0.35	0.21
L2-Net	0.23	0.40	0.26
HardNet	0.24	0.39	0.27
SOSNet	0.24	0.40	0.26
AlexNet-VP	0.26	0.39	0.26
SketchDesc	**0.55**	**0.49**	**0.54**

Tab. 5.4. Cross-dataset (chair) performance of different methods in the sketch correspondence task. The training sets are labeled with underlines.

at rank i is given by[1] $P_i(y) = \sum_{k=1}^{i}[y_k]_+ / \sum_{k=1}^{i}|y_k|$ and the *average precision* (AP) is given by $AP(y) = \sum_{k:y_k=+1} P_k(y) / \sum_{k=1}^{N}[y_k]_+$. Finally, given N_q as the number of total query pixels, the mean average precision (MAP) is computed by $MAP = \sum_{y=1}^{N_q} AP_y / N_q$. Experiments are performed category-wise in Structure-Recovery, PSB, and ShapeNet. To further report the performance on a dataset, we adopt avg_map which averages the MAP over all the tested categories.

Table 5.3 shows the results given by the compared methods. *SketchDesc* achieves the best performance among all the learned descriptors. Our learned descriptor surpasses the image-based descriptors of L2-Net, HardNet, and SOSNet by a large margin. For AlexNet-VP, it achieves a closer but still lower performance compared with our method.

5.2.3 Cross-dataset Validation

Considering the limited data in the testing set, we further demonstrate the generalization ability of *SketchDesc* with a cross-dataset validation.

For the three shape datasets (Structure-Recovery, PSB, and ShapeNet), we take two of them as the training set and the remaining one as the testing set. To reduce the effects of imbalanced data distribution, we report the performance on a more balanced and overlapping category (Chair), which has 27, 20, and 23 shapes in each of the three datasets respectively (Table 5.1). We report the performance of the sketch correspondence task and the multi-view pixel-wise retrieval task in Table 5.4 and Table 5.5 respectively.

[1] Here $[z]_+ = \max\{0, z\}$.

Methods	Structure-Recovery&ShapeNet / PSB	PSB&ShapeNet / Structure-Recovery	Structure-Recovery&PSB / ShapeNet
LeNet	0.07	0.01	0.04
L2-Net	0.16	0.35	0.13
HardNet	0.19	0.32	0.14
SOSNet	0.17	0.35	0.15
AlexNet-VP	0.54	0.40	0.24
SketchDesc	**0.73**	**0.64**	**0.54**

Tab. 5.5. Cross-dataset (chair) performance of different methods in the pixel-wise retrieval task. The training sets are labeled with underlines.

We observe that *SketchDesc* still shows an overwhelming superiority over its competitors. There is only a slight drop (0.02 on average) in the cross-dataset performance of the sketch correspondence task, compared to Table 5.2. However, the cross-dataset performance of the pixel-wise retrieval task shows a noticeable degradation (0.15 on average), compared to Table 5.3. This is mainly because that the pixel-wise retrieval task has a much larger search space (multi-view sketches of multiple shapes) than the sketch correspondence task (paired multi-view sketches of a single shape), and tends to lead to more mismatches due to ambiguity. In general, the robustness and effectiveness of *SketchDesc* are further verified by the large and unseen testing data.

5.2.4 View Disparity

To further show the robustness of different learned descriptors against the degree of view disparity, given the same input testing pixels in one sketch, we visualize how the quality of correspondence inference changes with the increasing view disparity. As shown in Figure 5.10, *SketchDesc* shows a more stable performance of correspondence inference than the competitors. Please note that the ground-truth corresponding pixels might become invisible in certain views, and all the learned descriptors could not distinguish the visibility of corresponding pixels. Nevertheless, *SketchDesc* still produces the most reasonable results.

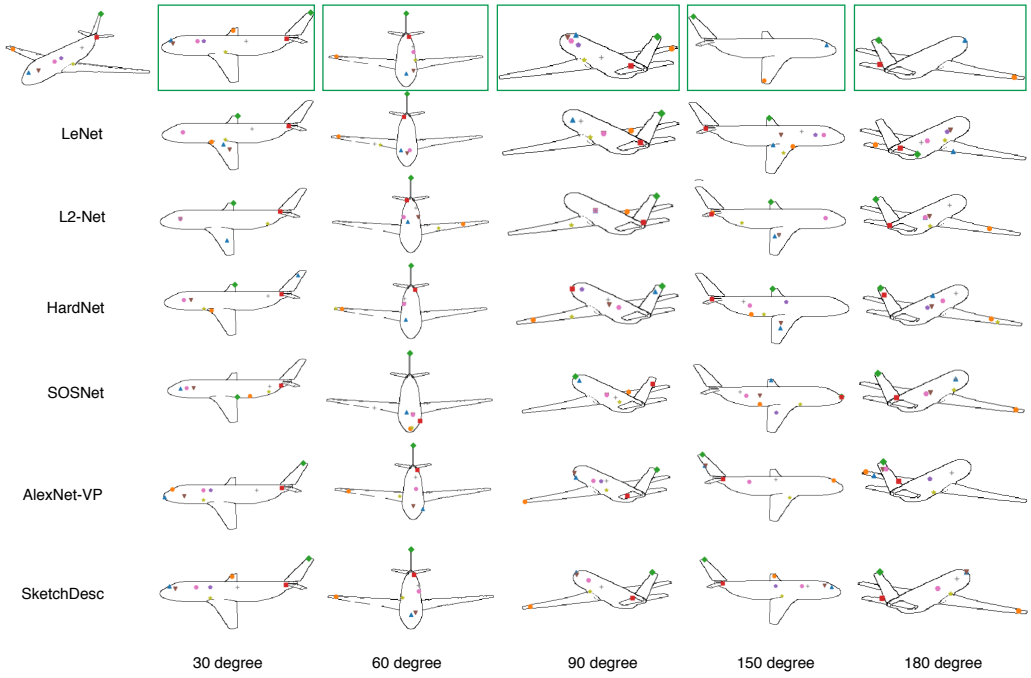


Fig. 5.10. Performance of different methods with increasing view disparity (30, 60, 90, 150, and 180 degrees). Given some anchor pixels on the sketched object at top-left, we show the corresponding pixels computed by the different methods. The ground-truth correspondences are labeled with green boxes.

5.2.5 Ablation Study

In this subsection, we validate the effectiveness of the key components (multi-scale strategy, shared weights structure, training data generation strategy) of our method with ablation studies.

Multi-scale Strategy. The designed multi-scale patch-based representation ($32 \times 32, 64 \times 64, 128 \times 128, 256 \times 256$) plays an essential role in our method. We first show how different scales can influence the performance of the learned descriptors. We test our network with increasingly more scales and evaluate its performance on the pixel-wise retrieval task. We employ the average MAP (Mean Average Precision) metric over the whole dataset. Quantitative results are shown in Table 5.6. We can see that the features from the larger scales are more discriminative than the features from smaller scales by ablating one of the multiple scales in Table 5.6 (rows 5 - 8). As we remove the larger scale, the poorer performance of *SketchDesc* is wit-

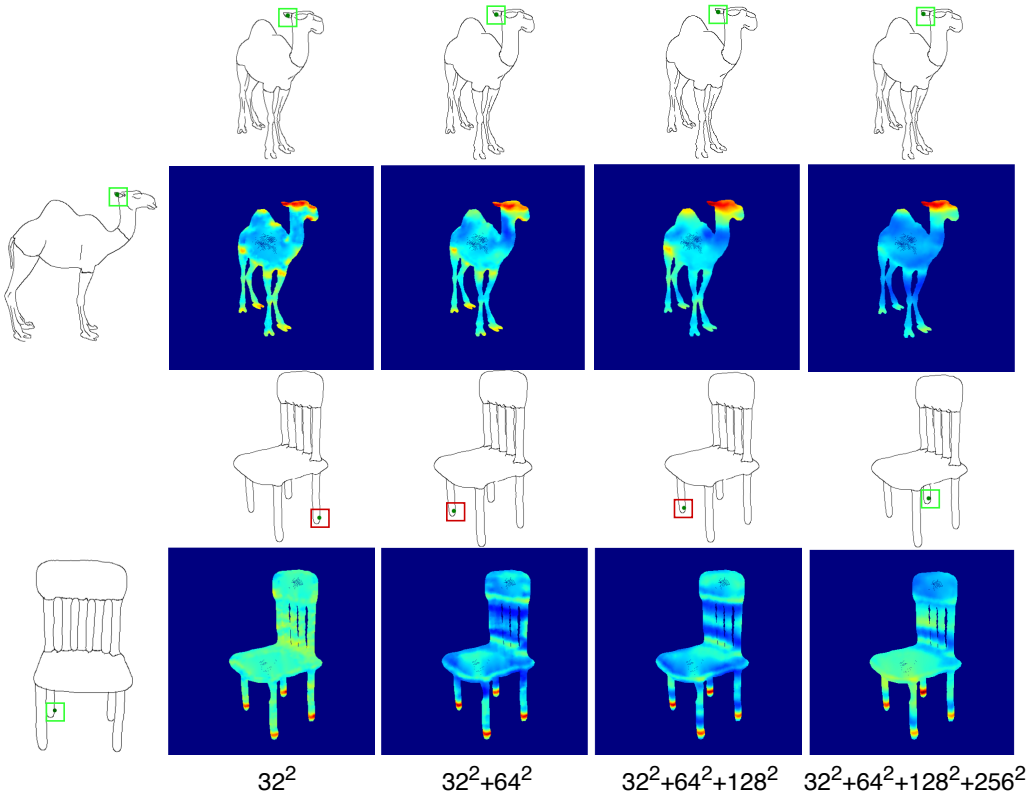


Fig. 5.11. Visualization of different multi-scale choices. The distance maps show the distances from the highlighted point in the left sketch to all the pixels in the other sketch.

nessed on all three datasets. Multiple scales are better than a single scale. A representative visual comparison is shown in Figure 5.11. It is found that as larger scales are involved, the ambiguous regions (bright yellow regions) on the feet, legs and backs of the camel are gradually rejected. In other words, with the multi-scale patches as inputs, our network can enjoy not only a more precise local perception but also a global perspective.

Shared Weights. In our method, the multi-scale patches are processed by a shared-weight scheme. To verify its effectiveness, we perform a comparison on the pixel-wise retrieval task with an unshared-weight network structure. The comparison results are reported in Table 5.7. It is found that the shared-weight structure in our network achieves higher accuracy. The improvement is even more significant on PSB. The results confirm a similar design choice of shared-weight structure used in existing studies like [45] and [81].

Different scales	Structure-Recovery	PSB	ShapeNet
32^2	0.47	0.31	0.20
64^2	0.64	0.44	0.45
128^2	0.75	0.65	0.61
256^2	0.76	0.66	0.62
$64^2 + 128^2 + 256^2$	0.81	0.69	0.66
$32^2 + 128^2 + 256^2$	0.81	0.67	0.65
$32^2 + 64^2 + 256^2$	0.79	0.68	0.64
$32^2 + 64^2 + 128^2$	0.77	0.67	0.61
$32^2 + 64^2$	0.70	0.57	0.48
$32^2 + 64^2 + 128^2$	0.77	0.67	0.61
$32^2 + 64^2 + 128^2 + 256^2$	**0.82**	**0.73**	**0.66**

Tab. 5.6. The performance of using different scale combinations as inputs to *SketchDesc*-Net in the pixel-wise retrieval task.

	Structure-Recovery	PSB	ShapeNet
W/o shared weights	0.80	0.65	0.63
W/ shared weights	**0.82**	**0.73**	**0.66**

Tab. 5.7. The performance of our method with or without shared-weights.

Training Data Generation. We evaluate the performance of two patch sampling mechanisms on the task of pixel-wise retrieval: OR-sampling and AND-sampling (Section 5.1.1). It can be found from Table 5.8 that OR-sampling achieves a significantly better performance. This is because the OR-sampling leads to a significantly larger dataset for training our network.

Sampling Mechanism	Structure-Recovery	PSB	ShapeNet
AND-sampling	0.79	0.68	0.59
OR-sampling	**0.82**	**0.73**	**0.66**

Tab. 5.8. The performance of two sampling mechanisms for data preparation on the task of multi-view pixel-wise retrieval. Note that different sampling mechanisms are only used to generate the training data.

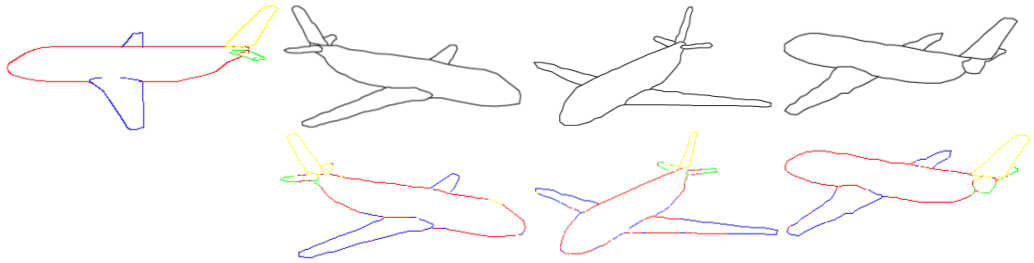


Fig. 5.12. Sketch segmentation transfer. The top row are the inputs: one segmented sketch and several unlabeled sketches. The bottom row are the outputs: sketches with point-wise labels after graph-cut postprocessing. With *SketchDesc*, we can transfer the labels among multi-view sketches with the computed correspondence.

5.3 Applications

5.3.1 Sketch Segmentation Transfer

Sketch segmentation is a challenging task that demands plenty of human-labeled training data [96, 65]. Noris et al. [84] proposed a scribble-based UI for user-guided segmentation of sketchy drawings, which still requires substantial human efforts. With *SketchDesc*, we show that segmentation labels can be easily transferred across multi-view sketches (Figure 5.12). Specifically, we use *SketchDesc* to produce 128-d descriptors for every point on the sketches. With the correspondences established by the descriptors, we transfer the segmentation from one labeled sketch to other views. As shown in Figure 5.12, although there are some distortions in the hand-drawn sketches, we can still obtain reasonable segmentation results through transfer.

5.3.2 Multi-view Image-based Correspondence

Image-based descriptors [122, 78, 123] mostly rely on the information of textures in patches to build the correspondence among multi-view images. If the input images lack discriminative texture details (Figure 5.13), the

Fig. 5.13. Correspondence matching among multi-view images of different objects.

image-based methods (e.g., the state-of-art SOSNet [123]) may fail to extract robust descriptors with their single-scale input. Here we show that our sketch-based descriptor *SketchDesc* can also be extended to this correspondence establishment among multi-view photos (Figure 5.13) under such challenging situations even with no textures. Here we use edge maps as a proxy, that is, we first convert a photo to its edge map format and then apply *SketchDesc* to obtain local descriptors for matching corresponding points in different views. Figure 5.13 shows that *SketchDesc* can infer a reasonable correspondence among multi-view images of different objects on mere edge maps, indicating the potential of our proposed *SketchDesc* in the image domain. The reason why SketchDesc outperforms SOSNet is because the multi-scale strategy in our method gives more confidence with the global and local perspectives. This further emphasizes the importance of th multi-scale idea in both the image and sketch domains. On the other hand, while we believe *SketchDesc* can assist existing photo correspondence

techniques especially when images do not have rich textures, we do not claim that *SketchDesc* is a general solution for photo correspondence.

5.4 Discussion

In this chapter, we have introduced a deep learning based method for correspondence learning among multiple sketches of an object in different views. We have proposed a multi-branch network that encodes contexts from multi-scale patches with global and local perspectives to produce a novel descriptor for semantically measuring the distance of pixels in multi-view sketch images. The multi-branch and shared-weights designs help the network capture more feature information from all scales of sketch patches. Our data preparation method provides the ground truth effectively for training our multi-branch network. We believe the generated data can benefit other applications. Both qualitative and quantitative experiments show that our learned descriptor is more effective than the existing learning-based descriptors. In the future, it would be interesting to exploit more neighboring information and learn the per-point features in a joint manner.

Our method has the following limitations. First, although our method could generalize well to unseen sketches or even hand-drawn sketches of the same objects, when the viewpoint differs from the examples in the training set drastically, our method could fail. This is a common generalization problem for any learning-based methods. Increasing the training data could help yet in the cost of additional training burden. We use a rather simple method to sample viewpoints for preparing the training data. A more careful view selection might be made by adopting best-view selection methods [59]. Additionally, our method is currently designed for multi-view correspondences of rigid objects. If the object undergoes articulation or non-rigid deformations (e.g. people dancing), our method may not perform well.

6

Conclusion and Future Work

In this dissertation, we have studied the topic of sketch-based shape and structure analysis and presented algorithms to automatically beautify imperfect freehand sketches in the form of part geometry and global structure, analyze the structural soundness of user-created sketch prototypes in product design and digital fabrication, and exploit the semantic shape correspondence from multi-view sketches.

To beautify imperfect sketches of man-made objects, we have introduced an intuitive and generic pipeline by conducting part-level geometry beautification and global structure refinement sequentially in Chapter 3.

For structural analysis on sketch prototypes, to simplify the complex and high-dimension physical forces, we have constrained the high-dimension and complex forces to the constant force magnitude and direction based on the estimation of a 2.5D normal map; to simulate the stress undertaken by 2D sketched structures under specified force locations, we have presented a two-branch generator *Sketch2Stress* to synthesize feasible structural stress considering the sketches geometry and force variables simultaneously in Chapter 4.

Specifically, for the semantic shape correspondence among multi-view sparse sketches, we have successfully proposed a novel multi-branch network that encodes contexts from multi-scale patches with global and local perspectives to produce a novel descriptor *SketchDesc* for semantically measuring the distance of pixels in multi-view sketch images in Chapter 5.

6.1 Future Work

There are many exciting and promising directions for future work. The geometry beautification step in Chapter 3 relies on the classical non-rigid registration approach [10]. Since registration is still known as an open problem in the current research community, we believe the combination with a more advanced registration approach could further boost the performance of freehand sketch beautification.

As for *Sketch2Stress* in Chapter 4, its ideas of simplifying physical forces and synthesizing the equivalent view-dependent 2D structural stress in advance in the sketching stage will be inspiring for future work, especially the subsequent series of *Sketch2Stress*, such as incorporating material properties, lightweight properties, and aesthetical factors (like the local geometrical or global structural aesthetics). Since 3D shapes are the most common and informative representations to connect the virtual world and reality, more interactions and applications on 3D shapes are flourishing in augmented reality (AR), virtual reality (VR) settings, etc. We speculate that adapting these interactions and applications initially operated on 3D shapes to 2D/3D sketches would produce more accessible, straightforward, and effective sketch-based applications for the public. This would further facilitate the creation, idea communication, and self-expression in arts. We are optimistic about the future of sketch-based interactions in design, fabrication, and art, especially in AR and VR scenarios that use sketches to augment or simulate the real world.

While we put significant efforts into exploring multi-scale context features of point-centered patches for semantic corresponding points in multi-view sketches, it remains an open question if the features (*SketchDesc*) discussed in Chapter 5 are sufficient to compare the relations/distances of the corresponding points and not corresponding ones. In particular, for sparse sketches in the rasterized format, most regions of a rasterized sketch image are likely to be empty. Instead of designing sketch representations and learning features from its rasterization format, it could be interesting to represent a multi-view sketch to its vectorization format and use graph neural networks (GNNs) to explore the features of not only the valid point data

(sketch implicit representation in Chapter 3) but also the neighboring relationships among points [143]. Our *SketchDesc* work demonstrated the usefulness of the correspondence of multi-view sketches in the segmentation transfer task (Figure 5.12). A more promising direction would be extending such multi-view correspondence relationships to the co-segmentation scenarios.

In this dissertation, as discussed above, we mainly focus on the sketch-based shape and structure analysis of a *single object* from the aspects of beautification, interaction with external factors, and multi-view correspondence. We find that the sketch is such an intuitive, simple, representative, and much clearer medium to curve the local geometry and global structure of objects. Actually, there is a wider space in exploring the scene-level sketch that contains *multiple objects*, such as richer presentation power and semantic meanings of multi-object sketches, the flexible spatial arrangements of combining multiple objects, and the interactions, motions, and deformations of multiple objects under external environmental factors (like forces). In the future, we are quite interested in using multi-object sketches to represent and simulate the mechanism of mutual effects of shapes in reality. Typically, when designers create a sketch, they may also include some writing or sketching annotation on the side, such as circles, arrows, crosses, pig-tales, words, and so on. However, in this thesis, we only consider the geometry and structure information of the strokes in a sketch, which ignores the annotations and design intentions provided by users. Nonetheless, this kind of information is quite important, such as the shadows indicating the lighting conditions, the hatching line representing the plat plane or curved surfaces, and scaffolds depicting the spatial information of the designed objects. In the future, we are also interested in utilizing these annotations to create more effective and intuitive design tools for the public.

Bibliography

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, et al. “Building rome in a day”. In: *Communications of the ACM* 54.10 (2011), pp. 105–112 (cit. on p. 23).

[2] Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. “ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models”. In: *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 2008, pp. 151–160 (cit. on p. 7).

[3] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. “HPatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors”. In: *Proc. IEEE CVPR*. 2017 (cit. on p. 24).

[4] Vassileios Balntas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. “HPatches: A benchmark and evaluation of handcrafted and learned local descriptors”. In: *CVPR*. 2017 (cit. on p. 91).

[5] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. “Learning local feature descriptors with triplets and shallow convolutional neural networks”. In: *Proc. BMVC*. 2016, pp. 119.1–119.11 (cit. on p. 24).

[6] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “Surf: Speeded up robust features”. In: *European Conference on Computer Vision*. Springer. 2006, pp. 404–417 (cit. on p. 24).

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. “Shape context: A new descriptor for shape matching and object recognition”. In: *NIPS*. 2001, pp. 831–837 (cit. on pp. 15, 24).

[8] Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Yongxin Yang, et al. “Vectorization and rasterization: Self-supervised learning for sketch and handwriting”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 5672–5681 (cit. on pp. 18, 19).

[9] Ayan Kumar Bhunia, Subhadeep Koley, Abdullah Faiz Ur Rahman Khilji, et al. “Sketching without worrying: Noise-tolerant sketch-based image retrieval”. In: *arXiv preprint arXiv:2203.14817* (2022) (cit. on p. 19).

[10] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. “Dynamic 2D/3D Registration.” In: *Eurographics (Tutorials)*. Citeseer. 2014, p. 7 (cit. on pp. 36, 102).

[11] John Canny. “A computational approach to edge detection”. In: *IEEE Transactions on pattern analysis and machine intelligence* 6 (1986), pp. 679–698 (cit. on pp. 4, 59).

[12] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, et al. “Shapenet: An information-rich 3d model repository”. In: *arXiv preprint arXiv:1512.03012* (2015) (cit. on pp. 58, 64).

[13] Jiaxin Chen and Yi Fang. “Deep cross-modality adaptation via semantics preserving adversarial learning for sketch-based 3d shape retrieval”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 605–620 (cit. on p. 6).

[14] Shu-Yu Chen, Wanchao Su, Lin Gao, Shihong Xia, and Hongbo Fu. “DeepFaceDrawing: Deep generation of face images from sketches”. In: *ACM Transactions on Graphics (TOG)* 39.4 (2020), pp. 72–1 (cit. on pp. 18, 34).

[15] Xiaobai Chen, Aleksey Golovinskiy, and Thomas Funkhouser. “A benchmark for 3D mesh segmentation”. In: *ACM Transactions on Graphics*. Vol. 28. 3. ACM. 2009, p. 73 (cit. on pp. 16, 82, 85).

[16] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. “Bsp-net: Generating compact meshes via binary space partitioning”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 45–54 (cit. on p. 19).

[17] Zhiqin Chen and Hao Zhang. “Learning implicit fields for generative shape modeling”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 5939–5948 (cit. on pp. 19, 33).

[18] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu, and Chang Huang. “A deep visual correspondence embedding model for stereo matching costs”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 972–980 (cit. on p. 15).

[19] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. “Universal Correspondence Network”. In: *NIPS*. 2016 (cit. on p. 24).

[20] Gabriela Csurka and Martin Humenberger. “From handcrafted to deep local invariant features”. In: *CoRR* abs/1807.10254 (2018) (cit. on p. 24).

[21] Guoxian Dai, Jin Xie, Fan Zhu, and Yi Fang. “Deep correlated metric learning for sketch-based 3d shape retrieval”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017 (cit. on p. 6).

[22] Navneet Dalal and Bill Triggs. “Histograms of oriented gradients for human detection”. In: *CVPR ’05*. 2005 (cit. on pp. 24, 41, 88, 90, 91).

[23] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. “Suggestive contours for conveying shape”. In: *ACM Transactions on Graphics (TOG)*. Vol. 22. 3. ACM. 2003, pp. 848–855 (cit. on p. 82).

[24] Johanna Delanoy, Mathieu Aubry, Phillip Isola, Alexei A Efros, and Adrien Bousseau. “3d sketching using multi-view deep volumetric prediction”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–22 (cit. on pp. 6, 11, 15, 26).

[25] Yu Deng, Jiaolong Yang, and Xin Tong. “Deformed implicit field: Modeling 3d shapes with learned dense correspondence”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 10286–10296 (cit. on p. 19).

[26] Daniel Dixon, Manoj Prasad, and Tracy Hammond. “icandraw: Using sketch recognition and corrective feedback to assist a user in drawing human faces”. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 2010, pp. 897–906 (cit. on pp. 7, 17).

[27] Dong Du, Xiaoguang Han, Hongbo Fu, et al. “SAniHead: Sketching animal-like 3D character heads using a view-surface collaborative mesh generative network”. In: *IEEE Transactions on Visualization and Computer Graphics* (2020) (cit. on pp. 58, 64).

[28] Jérémie Dumas, An Lu, Sylvain Lefebvre, Jun Wu, and Christian Dick. “By-example synthesis of structurally sound patterns”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–12 (cit. on pp. 10, 20).

[29] Helin Dutagaci, Chun Pan Cheung, and Afzal Godil. “A benchmark for best view selection of 3D objects”. In: *Proceedings of the ACM workshop on 3D object retrieval*. 2010, pp. 45–50 (cit. on p. 10).

[30] Mathias Eitz, James Hays, and Marc Alexa. “How do humans sketch objects?” In: *ACM Trans. Graph.* 31.4 (2012), pp. 44–1 (cit. on pp. 18, 25, 82).

[31] Guoxin Fang, Tianyu Zhang, Sikai Zhong, et al. “Reinforced FDM: Multi-axis filament alignment with controlled anisotropic strength”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–15 (cit. on pp. 10, 20).

[32] Jakub Fišer, Paul Asente, Stephen Schiller, and Daniel Sỳkora. “Advanced drawing beautification with shipshape”. In: *Computers & Graphics* 56 (2016), pp. 46–58 (cit. on pp. 7, 17, 41).

[33] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. “Upright orientation of man-made objects”. In: *ACM SIGGRAPH 2008 papers*. 2008, pp. 1–7 (cit. on p. 58).

[34] Lin Gao, Jie Yang, Tong Wu, et al. “SDM-NET: Deep generative network for structured deformable mesh”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), pp. 1–15 (cit. on pp. 10, 39).

[35] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256 (cit. on pp. 39, 64, 86).

[36] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, et al. “Generative Adversarial Nets”. In: *International Conference on Neural Information Processing Systems*. 2014 (cit. on p. 26).

[37] Lapo Governi, Rocco Furferi, Matteo Palai, and Yary Volpe. “3D geometry reconstruction from orthographic views: A method based on 3D image processing and data fitting”. In: *Computers in Industry* 64.9 (2013), pp. 1290–1300 (cit. on pp. 14, 26).

[38] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, et al. “OpenSketch: a richly-annotated dataset of product design sketches”. In: *ACM Transactions on Graphics (TOG)* 38.6 (2019), p. 232 (cit. on pp. 4, 6, 16, 26, 75, 85).

[39] Benoit Guillard, Edoardo Remelli, Pierre Yvernay, and Pascal Fua. “Sketch2Mesh: Reconstructing and Editing 3D Shapes from Sketches”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 13023–13032 (cit. on pp. 11, 12, 21).

[40] David Ha and Douglas Eck. “A neural representation of sketch drawings”. In: *arXiv preprint arXiv:1704.03477* (2017) (cit. on p. 25).

[41] David Ha and Douglas Eck. “A Neural Representation of Sketch Drawings”. In: *International Conference on Learning Representations*. 2018 (cit. on pp. 8, 18, 19).

[42] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. “MatchNet: Unifying feature and metric learning for patch-based matching”. In: *Proc. IEEE CVPR*. 2015, pp. 3279–3286 (cit. on p. 24).

[43] Heiko Hirschmuller and Daniel Scharstein. “Evaluation of stereo matching costs on images with radiometric differences”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.9 (2008), pp. 1582–1599 (cit. on pp. 15, 24).

[44] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851 (cit. on p. 23).

[45] Haibin Huang, Evangelos Kalogerakis, Siddhartha Chaudhuri, et al. “Learning local shape descriptors from part correspondences with multiview convolutional networks”. In: *ACM Transactions on Graphics (TOG)* 37.1 (2018), p. 6 (cit. on pp. 25, 26, 82, 96).

[46] Zhe Huang, Hongbo Fu, and Rynson WH Lau. “Data-driven segmentation and labeling of freehand sketches”. In: *ACM Transactions on Graphics (TOG)* 33.6 (2014), pp. 1–10 (cit. on pp. 18, 30).

[47] TJR HUGHES. “The Finite Element Method”. In: *Linear Static and Dynamic Finite Element Analysis* (1987) (cit. on p. 20).

[48] Takeo Igarashi, Satoshi Matsuoka, Sachiko Kawachiya, and Hidehiko Tanaka. “Interactive beautification: A technique for rapid geometric design”. In: *ACM SIGGRAPH 2007 courses*. 2007, 18–es (cit. on p. 17).

[49] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1125–1134 (cit. on pp. 2, 12, 22, 65).

[50] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. “Spatial transformer networks”. In: *Advances in neural information processing systems* 28 (2015) (cit. on p. 38).

[51] Jianbo Jiao, Ying Cao, Manfred Lau, and Rynson Lau. “Tactile sketch saliency”. In: *Proceedings of the 28th ACM International Conference on Multimedia*. 2020, pp. 3072–3080 (cit. on p. 22).

[52] Gabe Johnson, Mark Gross, Ellen Yi-Luen Do, and Jason Hong. “Sketch it, make it: sketching precise drawings for laser cutting”. In: *CHI’12 Extended Abstracts on Human Factors in Computing Systems*. 2012, pp. 1079–1082 (cit. on pp. 1, 10).

[53] Tero Karras, Samuli Laine, Miika Aittala, et al. “Analyzing and improving the image quality of stylegan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119 (cit. on p. 8).

[54] Michel Keller, Zetao Chen, Fabiola Maffra, Patrik Schmuck, and Margarita Chli. “Learning Deep Descriptors With Scale-Aware Triplet Networks”. In: *CVPR ’18*. 2018 (cit. on p. 24).

[55] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on p. 86).

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *NIPS*. 2012, pp. 1097–1105 (cit. on p. 26).

[57] Vijay Kumar B G, Gustavo Carneiro, and Ian Reid. “Learning Local Image Descriptors With Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions”. In: *Proc. IEEE CVPR*. 2016 (cit. on p. 24).

[58] Timothy Langlois, Ariel Shamir, Daniel Dror, Wojciech Matusik, and David IW Levin. “Stochastic structural analysis for context-aware design and fabrication”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), pp. 1–13 (cit. on pp. 2, 10, 20).

[59] Chang Ha Lee, Amitabh Varshney, and David W Jacobs. “Mesh saliency”. In: *ACM Transactions on Graphics (TOG)* 24.3 (2005), pp. 659–666 (cit. on p. 100).

[60] Yong Jae Lee, C Lawrence Zitnick, and Michael F Cohen. “Shadowdraw: real-time user guidance for freehand drawing”. In: *ACM Transactions on Graphics (TOG)* 30.4 (2011), pp. 1–10 (cit. on pp. 7, 17, 31).

[61] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. “Free2CAD: parsing freehand drawings into CAD commands”. In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–16 (cit. on p. 6).

[62] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. “Sketch2cad: Sequential cad modeling by sketching in context”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–14 (cit. on p. 6).

[63] Changjian Li, Hao Pan, Yang Liu, et al. “Robust flow-guided neural prediction for sketch-based freeform surface modeling”. In: *ACM Transactions on Graphics*. ACM. 2018, p. 238 (cit. on pp. 11, 15, 26).

[64] Jun Li, Kai Xu, Siddhartha Chaudhuri, et al. “Grass: Generative recursive autoencoders for shape structures”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–14 (cit. on p. 29).

[65] Lei Li, Hongbo Fu, and Chiew-Lan Tai. “Fast sketch segmentation and labeling with deep learning”. In: *IEEE computer graphics and applications* 39.2 (2018), pp. 38–51 (cit. on pp. 18, 30, 81, 98).

[66] Lei Li, Changqing Zou, Youyi Zheng, et al. “Sketch-R2CNN: An Attentive Network for Vector Sketch Recognition”. In: *arXiv preprint arXiv:1811.08170* (2018) (cit. on p. 25).

[67] Lei Li, Changqing Zou, Youyi Zheng, et al. “Sketch-r2cnn: An rnn-rasterization-cnn architecture for vector sketch recognition”. In: *IEEE transactions on visualization and computer graphics* 27.9 (2020), pp. 3745–3754 (cit. on p. 19).

[68] Chenxi Liu, Enrique Rosales, and Alla Sheffer. “Strokeaggregator: Consolidating raw sketches into artist-intended curve drawings”. In: *ACM Transactions on Graphics (TOG)* 37.4 (2018), pp. 1–15 (cit. on pp. 6, 17).

[69] Shubao Liu and David B Cooper. “Ray markov random fields for image-based 3d modeling: Model and efficient inference”. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE. 2010, pp. 1530–1537 (cit. on pp. 15, 23).

[70] David G Lowe. “Distinctive image features from scale-invariant keypoints”. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110 (cit. on pp. 23, 24).

[71] Lin Lu, Andrei Sharf, Haisen Zhao, et al. “Build-to-last: Strength to weight 3D printed objects”. In: *ACM Transactions on Graphics (ToG)* 33.4 (2014), pp. 1–10 (cit. on pp. 10, 20).

[72] Zhaoliang Lun, Matheus Gadelha, Evangelos Kalogerakis, Subhransu Maji, and Rui Wang. “3d shape reconstruction from sketches via multi-view convolutional networks”. In: *2017 International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 67–77 (cit. on pp. 6, 11, 26).

[73] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. “Efficient deep learning for stereo matching”. In: *CVPR ’16*. 2016, pp. 5695–5703 (cit. on p. 15).

[74] Zixin Luo, Tianwei Shen, Lei Zhou, et al. “Geodesc: Learning local descriptors by integrating geometry constraints”. In: *ECCV ’18*. 2018, pp. 168–183 (cit. on pp. 2, 24).

[75] Chenlin Meng, Yutong He, Yang Song, et al. “Sdedit: Guided image synthesis and editing with stochastic differential equations”. In: *International Conference on Learning Representations*. 2021 (cit. on p. 23).

[76] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. “Occupancy networks: Learning 3d reconstruction in function space”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 4460–4470 (cit. on pp. 2, 19, 21).

[77] Masaaki Miki, Takeo Igarashi, and Philippe Block. “Parametric self-supporting surfaces via direct computation of airy stress functions”. In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pp. 1–12 (cit. on p. 10).

[78] Anastasiia Mishchuk, Dmytro Mishkin, Filip Radenovic, and Jiri Matas. “Working hard to know your neighbor’s margins: Local descriptor learning loss”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 4826–4837 (cit. on pp. 15, 24, 88, 98).

[79] Dmytro Mishkin, Filip Radenovic, and Jiri Matas. “Repeatability is not enough: Learning affine regions via discriminability”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 284–300 (cit. on p. 24).

[80] Kaichun Mo, Shilin Zhu, Angel X Chang, et al. “Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 909–918 (cit. on pp. 10, 39).

[81] Pablo Navarro, José Ignacio Orlando, Claudio Delrieux, and Emmanuel Iarussi. “SketchZooms: Deep multi-view descriptors for matching line drawings”. In: *arXiv preprint arXiv:1912.05019* (2019) (cit. on pp. 26, 82, 96).

[82] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, et al. “Glide: Towards photorealistic image generation and editing with text-guided diffusion models”. In: *arXiv preprint arXiv:2112.10741* (2021) (cit. on p. 23).

[83] Gen Nishida, Ignacio Garcia-Dorado, Daniel G Aliaga, Bedrich Benes, and Adrien Bousseau. “Interactive sketching of urban procedural models”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 130 (cit. on pp. 15, 26).

[84] Gioacchino Noris, Daniel Sỳkora, A Shamir, et al. “Smart scribbles for sketch segmentation”. In: *Computer Graphics Forum*. Vol. 31. 8. Wiley Online Library. 2012, pp. 2516–2527 (cit. on p. 98).

[85] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. “Deep Metric Learning via Lifted Structured Feature Embedding”. In: *Proc. IEEE CVPR*. 2016 (cit. on p. 24).

[86] Gunay Orbay and Levent Burak Kara. “Beautification of design sketches using trainable stroke clustering and curve fitting”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.5 (2011), pp. 694–708 (cit. on pp. 6, 17).

[87] Julian Panetta, Abtin Rahimian, and Denis Zorin. “Worst-case stress relief for microstructures”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–16 (cit. on pp. 10, 20).

[88] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. “Deepsdf: Learning continuous signed distance functions for shape representation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 165–174 (cit. on pp. 2, 19).

[89] Adam Paszke, Sam Gross, Francisco Massa, et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in neural information processing systems*. 2019, pp. 8026–8037 (cit. on pp. 39, 64, 86).

[90] Theo Pavlidis and Christopher J Van Wyk. “An automatic beautifier for drawings and illustrations”. In: *ACM SIGGRAPH Computer Graphics* 19.3 (1985), pp. 225–234 (cit. on p. 17).

[91] Romain Prévost, Emily Whiting, Sylvain Lefebvre, and Olga Sorkine-Hornung. “Make it stand: balancing shapes for 3D fabrication”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–10 (cit. on pp. 10, 20).

[92] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. “Hierarchical text-conditional image generation with clip latents”. In: *arXiv preprint arXiv:2204.06125* (2022) (cit. on p. 23).

[93] Alec Rivers, Frédo Durand, and Takeo Igarashi. “3D Modeling with Silhouettes”. In: *ACM Trans. Graph.* 29.4 (July 2010) (cit. on p. 26).

[94] Shunsuke Saito, Zeng Huang, Ryota Natsume, et al. “Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 2304–2314 (cit. on p. 19).

[95] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. “The sketchy database: learning to retrieve badly drawn bunnies”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 119 (cit. on p. 25).

[96] Rosália G. Schneider and Tinne Tuytelaars. “Example-Based Sketch Segmentation and Labeling Using CRFs”. In: *ACM Trans. Graph.* 35.5 (July 2016) (cit. on p. 98).

[97] Rosália G Schneider and Tinne Tuytelaars. “Sketch classification and classification-driven analysis using fisher vectors”. In: *ACM Transactions on graphics (TOG)* 33.6 (2014), pp. 1–9 (cit. on p. 18).

[98] Nadav Schor, Oren Katzir, Hao Zhang, and Daniel Cohen-Or. “CompoNet: Learning to generate the unseen by part synthesis and composition”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 8759–8768 (cit. on pp. 7, 18).

[99] Florian Schroff, Dmitry Kalenichenko, and James Philbin. “Facenet: A unified embedding for face recognition and clustering”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 815–823 (cit. on pp. 3, 85).

[100] Christian Schumacher, Bernd Bickel, Jan Rys, et al. “Microstructures to control elasticity in 3D printing”. In: *ACM Transactions on Graphics (Tog)* 34.4 (2015), pp. 1–13 (cit. on p. 10).

[101] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. “Structure recovery by part assembly”. In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), p. 180 (cit. on pp. 16, 82, 85).

[102] Philip Shilane and Thomas Funkhouser. “Distinctive regions of 3D surfaces”. In: *ACM Transactions on Graphics (TOG)* 26.2 (2007), p. 7 (cit. on p. 25).

[103] Oana Sidi, Oliver van Kaick, Yanir Kleiman, Hao Zhang, and Daniel Cohen-Or. “Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering”. In: *Proceedings of the 2011 SIGGRAPH Asia Conference*. 2011, pp. 1–10 (cit. on pp. 10, 39, 64).

[104] E. Simo-Serra, E. Trulls, L. Ferraz, et al. “Discriminative Learning of Deep Convolutional Feature Point Descriptors”. In: *ICCV ’15*. 2015, pp. 118–126 (cit. on p. 24).

[105] Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. “Mastering sketching: adversarial augmentation for structured prediction”. In: *ACM Transactions on Graphics (TOG)* 37.1 (2018), pp. 1–13 (cit. on pp. 6, 7, 17, 41).

[106] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. “Learning to simplify: fully convolutional networks for rough sketch cleanup”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–11 (cit. on p. 7).

[107] Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. “Learning Manifold Patch-Based Representations of Man-Made Shapes”. In: *International Conference on Learning Representations*. 2020 (cit. on p. 6).

[108] Dmitriy Smirnov, Mikhail Bessmeltsev, and Justin Solomon. “Learning Manifold Patch-Based Representations of Man-Made Shapes”. In: *International Conference on Learning Representations*. 2021 (cit. on pp. 11, 12).

[109] Noah Snavely, Steven M Seitz, and Richard Szeliski. “Photo tourism: exploring photo collections in 3D”. In: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM. 2006, pp. 835–846 (cit. on p. 23).

[110] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2256–2265 (cit. on p. 23).

[111] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, et al. “Score-based generative modeling through stochastic differential equations”. In: *arXiv preprint arXiv:2011.13456* (2020) (cit. on p. 23).

[112] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, et al. “Laplacian surface editing”. In: *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*. 2004, pp. 175–184 (cit. on p. 41).

[113] Ondrej Stava, Juraj Vanek, Bedrich Benes, Nathan Carr, and Radomír Měch. “Stress relief: improving structural strength of 3D printable objects”. In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), pp. 1–11 (cit. on pp. 2, 10, 20).

[114] David Stutz and Andreas Geiger. “Learning 3d shape completion under weak supervision”. In: *International Journal of Computer Vision* 128.5 (2020), pp. 1162–1181 (cit. on p. 58).

[115] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. “Multi-view convolutional neural networks for 3d shape recognition”. In: *ICCV ’15*. 2015, pp. 945–953 (cit. on pp. 26, 81).

[116] Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. “Live sketch: Video-driven dynamic deformation of static drawings”. In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, pp. 1–12 (cit. on p. 6).

[117] Qingkun Su, Wing Ho Andy Li, Jue Wang, and Hongbo Fu. “EZ-sketching: three-level optimization for error-tolerant image tracing.” In: *ACM Trans. Graph.* 33.4 (2014), pp. 54–1 (cit. on p. 18).

[118] Wanchao Su, Dong Du, Xin Yang, Shizhe Zhou, and Hongbo Fu. “Interactive sketch-based normal map generation with deep neural networks”. In: *Proceedings of the ACM on Computer Graphics and Interactive Techniques* 1.1 (2018), pp. 1–17 (cit. on pp. 22, 78).

[119] Wanchao Su, Xin Yang, and Hongbo Fu. “Sketch2normal: deep networks for normal map generation”. In: *SIGGRAPH Asia 2017 Posters*. 2017, pp. 1–2 (cit. on p. 6).

[120] Ivan E Sutherland. “Sketchpad a man-machine graphical communication system”. In: *Simulation* 2.5 (1964), R–3 (cit. on p. 17).

[121] Alexandru Telea and Andrei Jalba. “Voxel-based assessment of printability of 3D shapes”. In: *International symposium on mathematical morphology and its applications to signal and image processing*. Springer. 2011, pp. 393–404 (cit. on pp. 2, 20).

[122] Yurun Tian, Bin Fan, and Fuchao Wu. “L2-Net: Deep learning of discriminative patch descriptor in euclidean space”. In: *CVPR ’17*. 2017, pp. 661–669 (cit. on pp. 2, 15, 16, 24, 25, 80, 84, 98).

[123] Yurun Tian, Xin Yu, Bin Fan, et al. “SOSNet: Second order similarity regularization for local descriptor learning”. In: *CVPR ’19*. 2019, pp. 11016–11025 (cit. on pp. 2, 15, 24, 98, 99).

[124] Federico Tombari, Stefano Mattoccia, Luigi Di Stefano, and Elisa Addimanda. “Classification and evaluation of cost aggregation methods for stereo correspondence”. In: *CVPR ’08*. 2008, pp. 1–8 (cit. on pp. 15, 24).

[125] Tony Tung, Shohei Nobuhara, and Takashi Matsuyama. “Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo”. In: *ICCV ’09*. IEEE. 2009, pp. 1709–1716 (cit. on pp. 15, 23).

[126] Erva Ulu, James Mccann, and Levent Burak Kara. “Lightweight structure design under force location uncertainty”. In: *ACM Transactions on Graphics (TOG)* 36.4 (2017), pp. 1–13 (cit. on pp. 2, 10, 12, 13, 20, 22, 56, 59, 77).

[127] Dave Pagurek Van Mossel, Chenxi Liu, Nicholas Vining, Mikhail Bessmeltsev, and Alla Sheffer. “StrokeStrip: joint parameterization and fitting of stroke clusters”. In: *ACM Transactions on Graphics (TOG)* 40.4 (2021), pp. 1–18 (cit. on pp. 6, 17).

[128] Beirong Wang, Jian Sun, and Beryl Plimmer. “Exploring sketch beautification techniques”. In: *Proceedings of the 6th ACM SIGCHI New Zealand chapter’s international conference on Computer-human interaction: making CHI natural*. 2005, pp. 15–16 (cit. on p. 6).

[129] Hao Wang, Nadav Schor, Ruizhen Hu, et al. “Global-to-local generative model for 3d shapes”. In: *ACM Transactions on Graphics (TOG)* 37.6 (2018), pp. 1–10 (cit. on p. 29).

[130] Nannan Wang, Dacheng Tao, Xinbo Gao, Xuelong Li, and Jie Li. “A comprehensive survey to face hallucination”. In: *International journal of computer vision* 106.1 (2014), pp. 9–30 (cit. on p. 25).

[131] Nanyang Wang, Yinda Zhang, Zhuwen Li, et al. “Pixel2mesh: Generating 3d mesh models from single rgb images”. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 52–67 (cit. on pp. 11, 12, 21).

[132] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, et al. “High-resolution image synthesis and semantic manipulation with conditional gans”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8798–8807 (cit. on pp. 2, 12, 22, 57, 65).

[133] Weiming Wang, Tuanfeng Y Wang, Zhouwang Yang, et al. “Cost-effective printing of 3D objects with skin-frame structures”. In: *ACM Transactions on Graphics (ToG)* 32.6 (2013), pp. 1–10 (cit. on p. 10).

[134] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, et al. “Active co-analysis of a set of shapes”. In: *ACM Transactions on Graphics (TOG)* 31.6 (2012), pp. 1–10 (cit. on p. 58).

[135] Xing Wei, Yue Zhang, Yihong Gong, and Nanning Zheng. “Kernelized Subspace Pooling for Deep Local Descriptors”. In: *CVPR ’18*. 2018 (cit. on p. 24).

[136] S. Winder and M. Brown. “Learning Local Image Descriptors”. In: *Proc. IEEE CVPR*. 2007 (cit. on p. 24).

[137] Rundi Wu, Yixin Zhuang, Kai Xu, Hao Zhang, and Baoquan Chen. “PQ-NET: A Generative Part Seq2Seq Network for 3D Shapes”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2020, pp. 826–835 (cit. on p. 29).

[138] Wayne Wu, Kaidi Cao, Cheng Li, Chen Qian, and Chen Change Loy. “Transgaga: Geometry-aware unsupervised image-to-image translation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 8012–8021 (cit. on p. 8).

[139] Kun Xu, Kang Chen, Hongbo Fu, Wei-Lun Sun, and Shi-Min Hu. “Sketch2Scene: sketch-based co-retrieval and co-placement of 3D models”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 123 (cit. on p. 81).

[140] Peng Xu, Yongye Huang, Tongtong Yuan, et al. “Sketchmate: Deep hashing for million-scale human sketch retrieval”. In: *CVPR ’18*. 2018, pp. 8090–8098 (cit. on pp. 19, 25).

[141] Chuan Yan, David Vanderhaeghe, and Yotam Gingold. “A benchmark for rough sketch cleanup”. In: *ACM Transactions on Graphics (TOG)* 39.6 (2020), pp. 1–14 (cit. on p. 17).

[142] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. “Hierarchical Deep Stereo Matching on High-Resolution Images”. In: *CVPR ’19*. 2019, pp. 5515–5524 (cit. on p. 15).

[143] Lumin Yang, Jiajie Zhuang, Hongbo Fu, et al. “SketchGNN: Semantic Sketch Segmentation with Graph Neural Networks”. In: *ACM Transactions on Graphics (TOG)* 40.3 (2021), pp. 1–13 (cit. on pp. 18, 30, 103).

[144] Miaojun Yao, Zhili Chen, Linjie Luo, Rui Wang, and Huamin Wang. “Level-set-based partitioning and packing optimization of a printable model”. In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), pp. 1–11 (cit. on p. 10).

[145] Hui Ye, Kin Chung Kwan, and Hongbo Fu. “3D curve creation on and around physical objects with mobile AR”. In: *IEEE Transactions on Visualization & Computer Graphics* 01 (2021), pp. 1–1 (cit. on p. 6).

[146] Li Yi, Vladimir G Kim, Duygu Ceylan, et al. “A scalable active framework for region annotation in 3d shape collections”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 210 (cit. on pp. 15, 82, 85).

[147] Deng Yu, Lei Li, Youyi Zheng, et al. “Sketchdesc: Learning local sketch descriptors for multi-view correspondence”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.5 (2020), pp. 1738–1750 (cit. on pp. 6, 52).

[148] Qian Yu, Yongxin Yang, Feng Liu, et al. “Sketch-a-net: A deep neural network that beats humans”. In: *International Journal of Computer Vision* 122.3 (2017), pp. 411–425 (cit. on pp. 16, 25).

[149] Qian Yu, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. “Sketch-a-Net that Beats Humans”. In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press. 2015, pp. 1–12 (cit. on pp. 8, 19).

[150] Sergey Zagoruyko and Nikos Komodakis. “Learning to Compare Image Patches via Convolutional Neural Networks”. In: *CVPR ’15*. 2015 (cit. on p. 24).

[151] Jonas Zehnder, Stelian Coros, and Bernhard Thomaszewski. “Designing structurally-sound ornamental curve networks”. In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), pp. 1–10 (cit. on pp. 10, 20).

[152] Song-Hai Zhang, Yuan-Chen Guo, and Qing-Wen Gu. “Sketch2Model: View-aware 3d modeling from single free-hand sketches”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 6012–6021 (cit. on pp. 11, 12, 21).

[153] Tongjie Y Zhang and Ching Y. Suen. “A fast parallel algorithm for thinning digital patterns”. In: *Communications of the ACM* 27.3 (1984), pp. 236–239 (cit. on pp. 31, 41).

[154] Xu Zhang, Felix X. Yu, Sanjiv Kumar, and Shih-Fu Chang. “Learning Spread-Out Local Feature Descriptors”. In: *ICCV ’17*. 2017 (cit. on p. 24).

[155] Qingnan Zhou, Julian Panetta, and Denis Zorin. “Worst-case structural analysis.” In: *ACM Trans. Graph.* 32.4 (2013), pp. 137–1 (cit. on pp. 2, 10, 20).

[156] Mingrui Zhu, Jie Li, Nannan Wang, and Xinbo Gao. “A deep collaborative framework for face photo–sketch synthesis”. In: *IEEE transactions on neural networks and learning systems* 30.10 (2019), pp. 3096–3108 (cit. on p. 25).

[157] Mingrui Zhu, Nannan Wang, Xinbo Gao, Jie Li, and Zhifeng Li. “Face Photo-Sketch Synthesis via Knowledge Transfer.” In: *IJCAI*. 2019, pp. 1048–1054 (cit. on p. 25).

[158] C Lawrence Zitnick. “Handwriting beautification using token means”. In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), pp. 1–8 (cit. on p. 6).

A

Publications Related to This Dissertation

1. Deng Yu, Chufeng Xiao, Manfred Lau, and Hongbo Fu. "Sketch2Stress: Sketching with Structural Stress Awareness". (*Under Review*).
2. Deng Yu, Lin Gao, Manfred Lau, and Hongbo Fu. "Sketch Beautification: Learning Part Beautification and Structure Refinement for Sketches of Man-made Objects". (*Under Review*).
3. Chufeng Xiao, Deng Yu, Xiaoguang Han, Youyi Zheng, and Hongbo Fu. "SketchHairSalon: Deep Sketch-based Hair Image Synthesis". In ACM Transactions on Graphics (TOG). (*Proceedings of ACM SIGGRAPH Asia 2021*)
4. Deng Yu, Lei Li, Youyi Zheng, Manfred Lau, Yi-Zhe Song, Chiew-Lan Tai, and Hongbo Fu."SketchDesc: Learning local sketch descriptors for multi-view correspondence." In IEEE Transactions on Circuits and Systems for Video Technology (TCSVT), 31(5), (2020): 1738 - 1750.

B

Implementation Details and Sketch Beautification Results

The implementation details and more results of our sketch beautification approach are provided in this appendix chapter. Figures B.1 and B.2 illustrate the parametric network structures of the part beautification module and structure refinement module in our sketch beautification approach. Figures B.3, B.4, and B.5 show more beautification results of our approach.

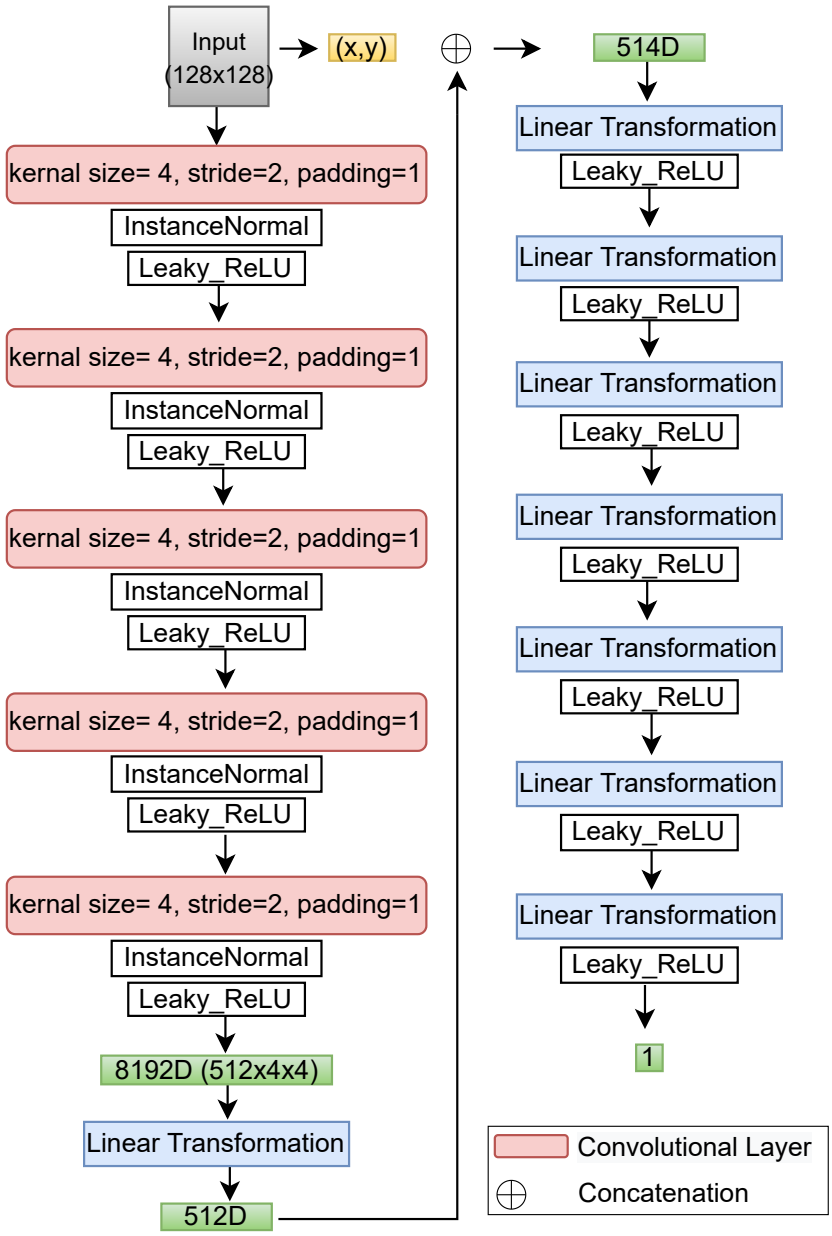


Fig. B.1. Network structure of the sketch implicit model.

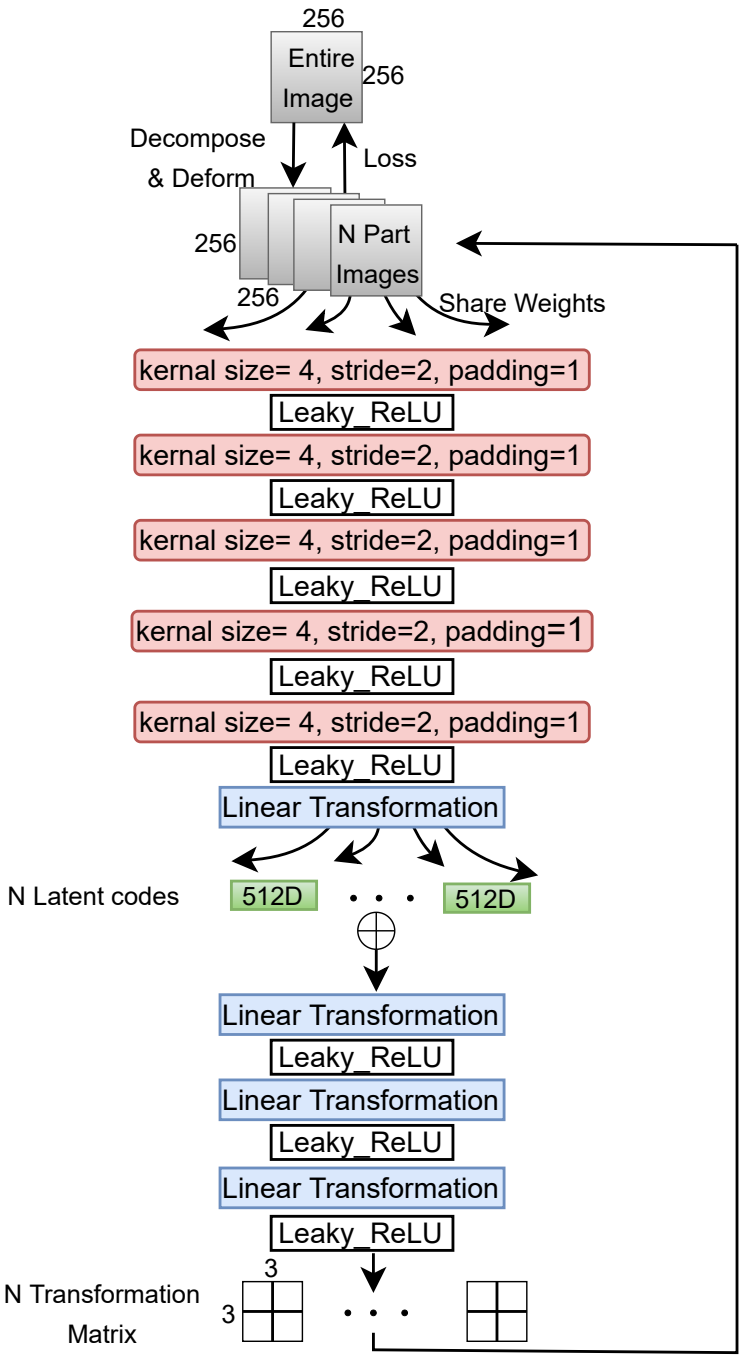


Fig. B.2. Spatial transformation network (STN) for sketch assembly.

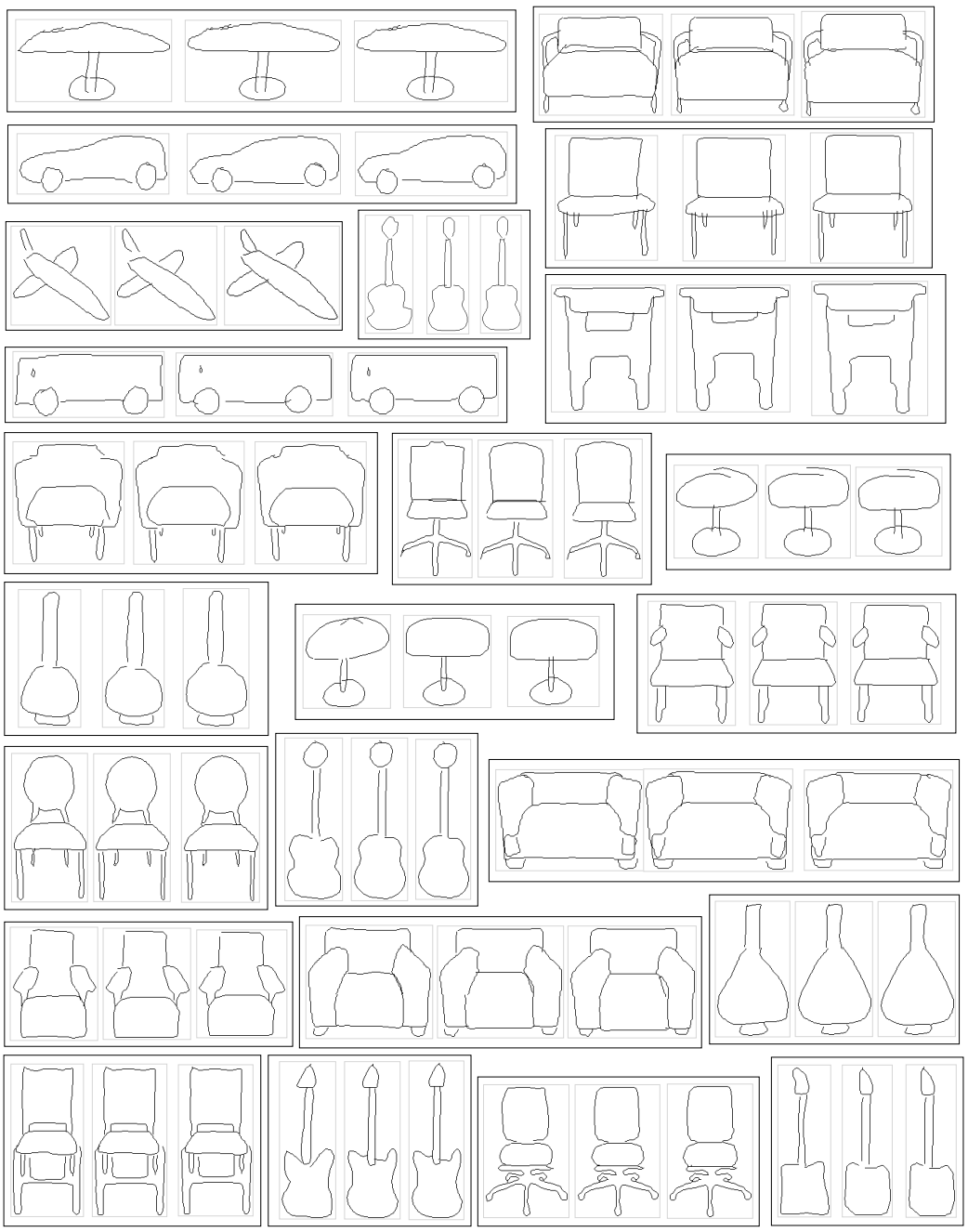


Fig. B.3. Visual results of our method. Each triplet contains an input sketch (Left), the sketch after part beautification (Middle), and the final result after structure beautification (Right). Please zoom in for better visualization.

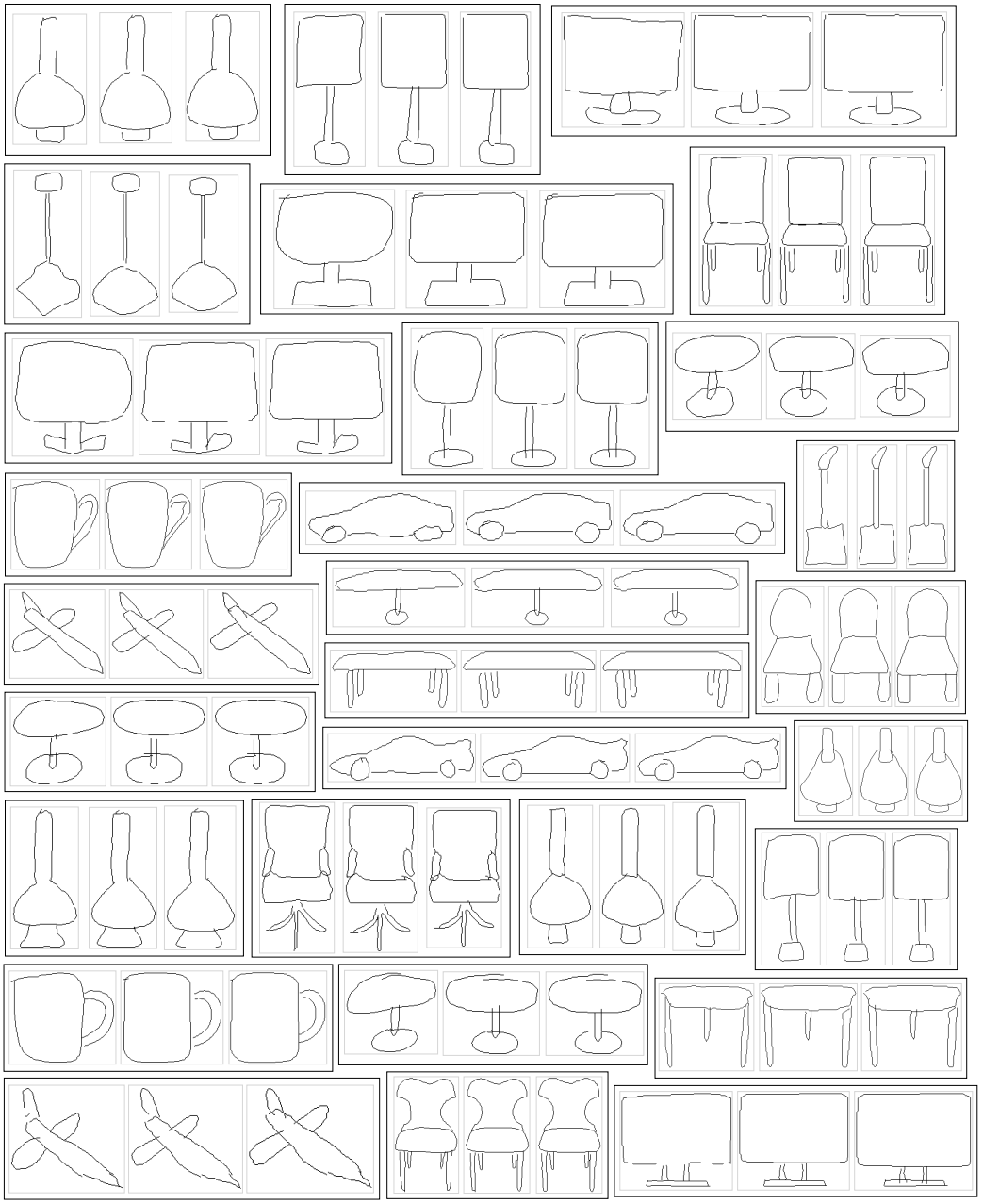


Fig. B.4. Visual results of our method. Each triplet contains an input sketch (Left), the sketch after part beautification (Middle), and the final result after structure beautification (Right). Please zoom in for better visualization.

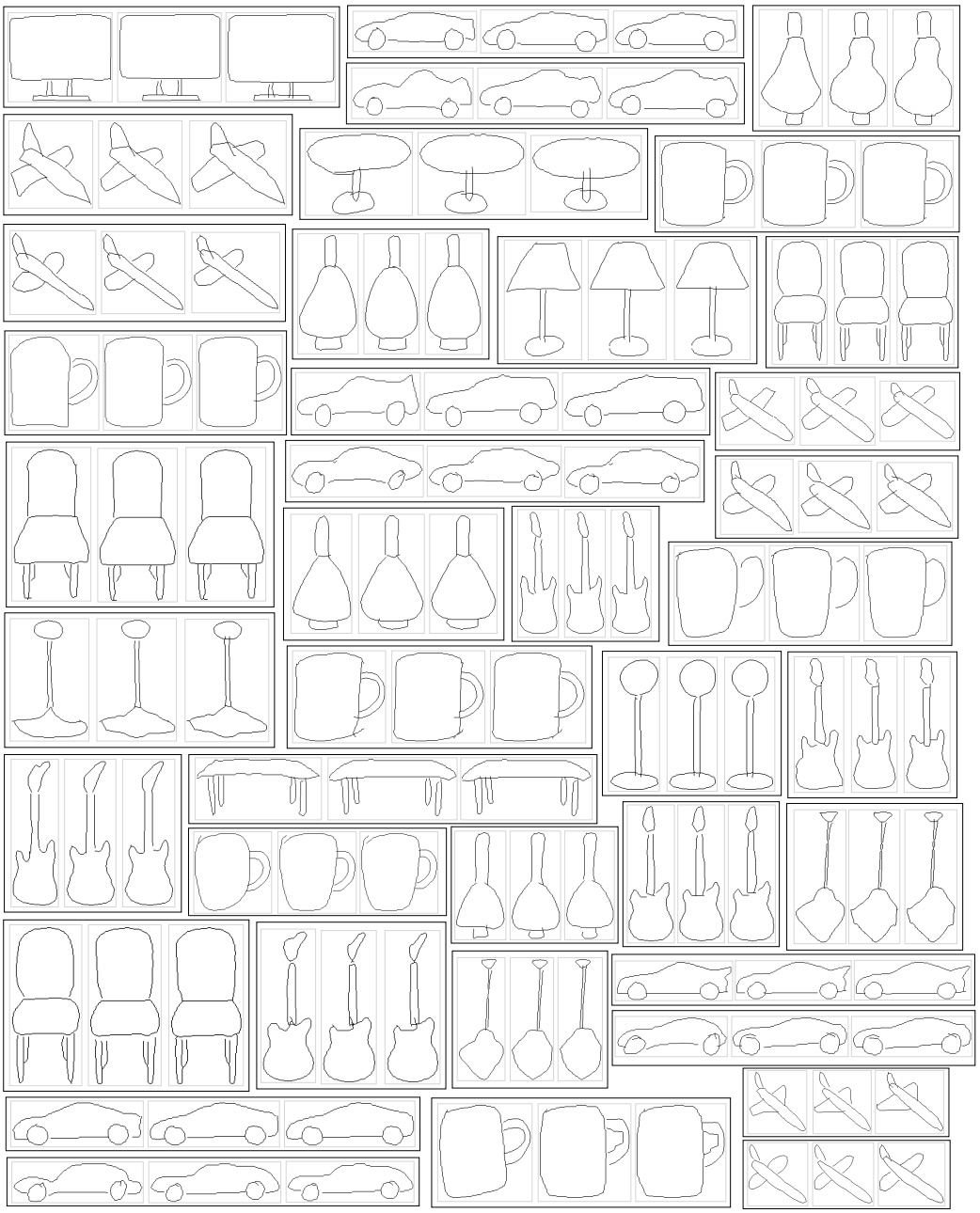


Fig. B.5. Visual results of our method. Each triplet contains an input sketch (Left), the sketch after part beautification (Middle), and the final result after structure beautification (Right). Please zoom in for better visualization.

C

Network Details and More Evaluations of *Sketch2Stress*

The parametric network details and more evaluations of *Sketch2Stress* are provided in this appendix chapter. Figure C.1 illustrates the parametric network structures of our *Sketch2Stress* approach. Figures C.2, C.3, C.4 further present more qualitative results of compared methods across different categories.

Network Details We adopt a framework of a two-branch generator and multi-scale discriminators to synthesize the structural stress map directly from the inputs of sketches and the user's specified force configurations. The two-branch generators are designed as an encoder-decoder architecture, as shown in Figure C.1. For multi-scale discriminators, we adopted the same pixel discriminators as pix2pixHD to distinguish the real/fake multi-scale normal maps and stress maps.

More Evaluations To demonstrate the performance of all competitors in the sketch-based structural stress generation task, we display more qualitative results in Figure C.2, Figure C.3, and Figure C.4. Note that all the methods are tested on the unseen data.

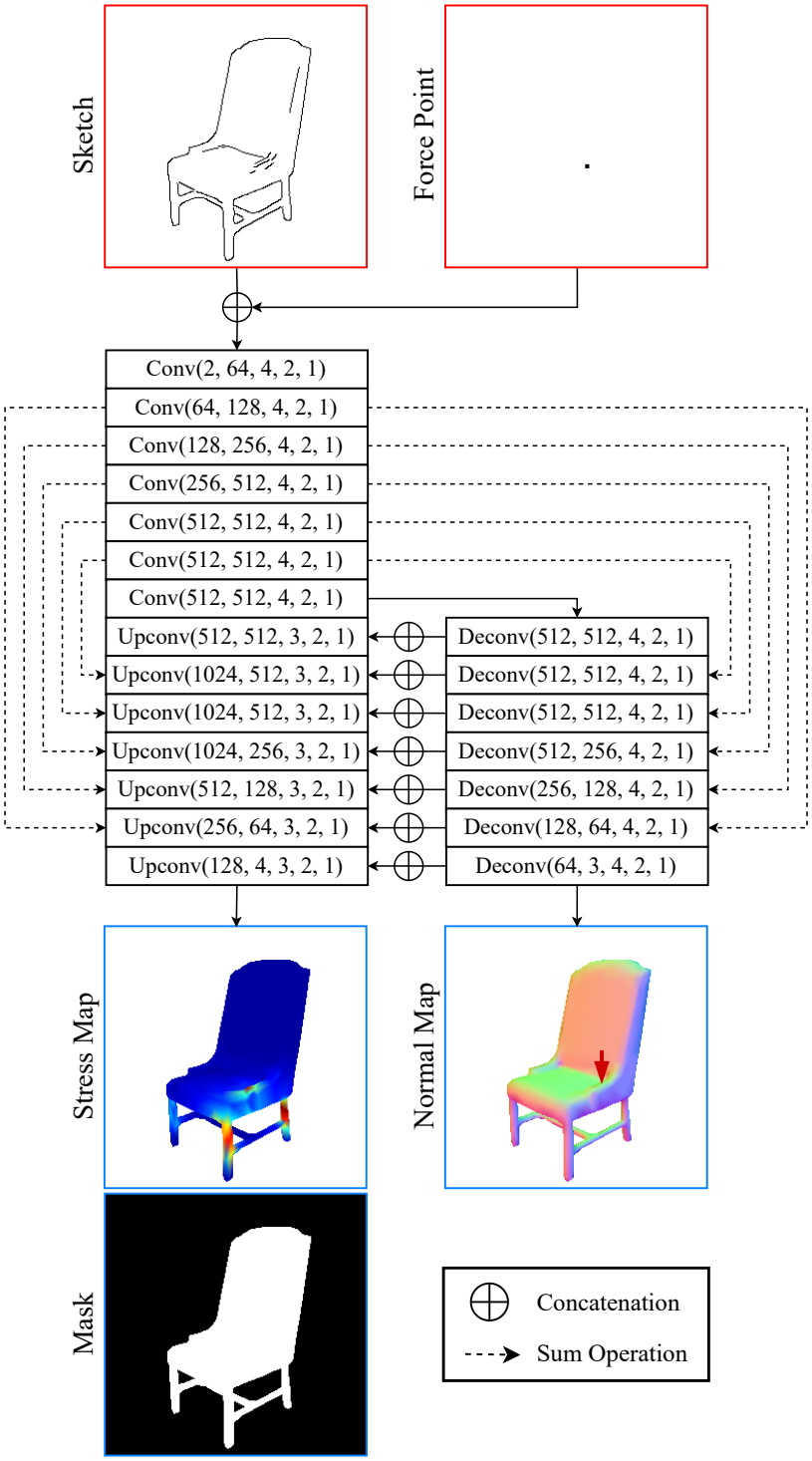


Fig. C.1. The details of our two-branch *Sketch2Stress* network. The block of *Conv/Upconv/Deconv(input_channel_number, output_channel_number, kernel_size, stride, padding_width)* respectively represents Convolution, Upsampling + Convolution, and Deconvolution layer. Note that each Conv/Upconv/Deconv layer is followed by leaky ReLU and batch normalization, which are omitted in the figure for simplicity.

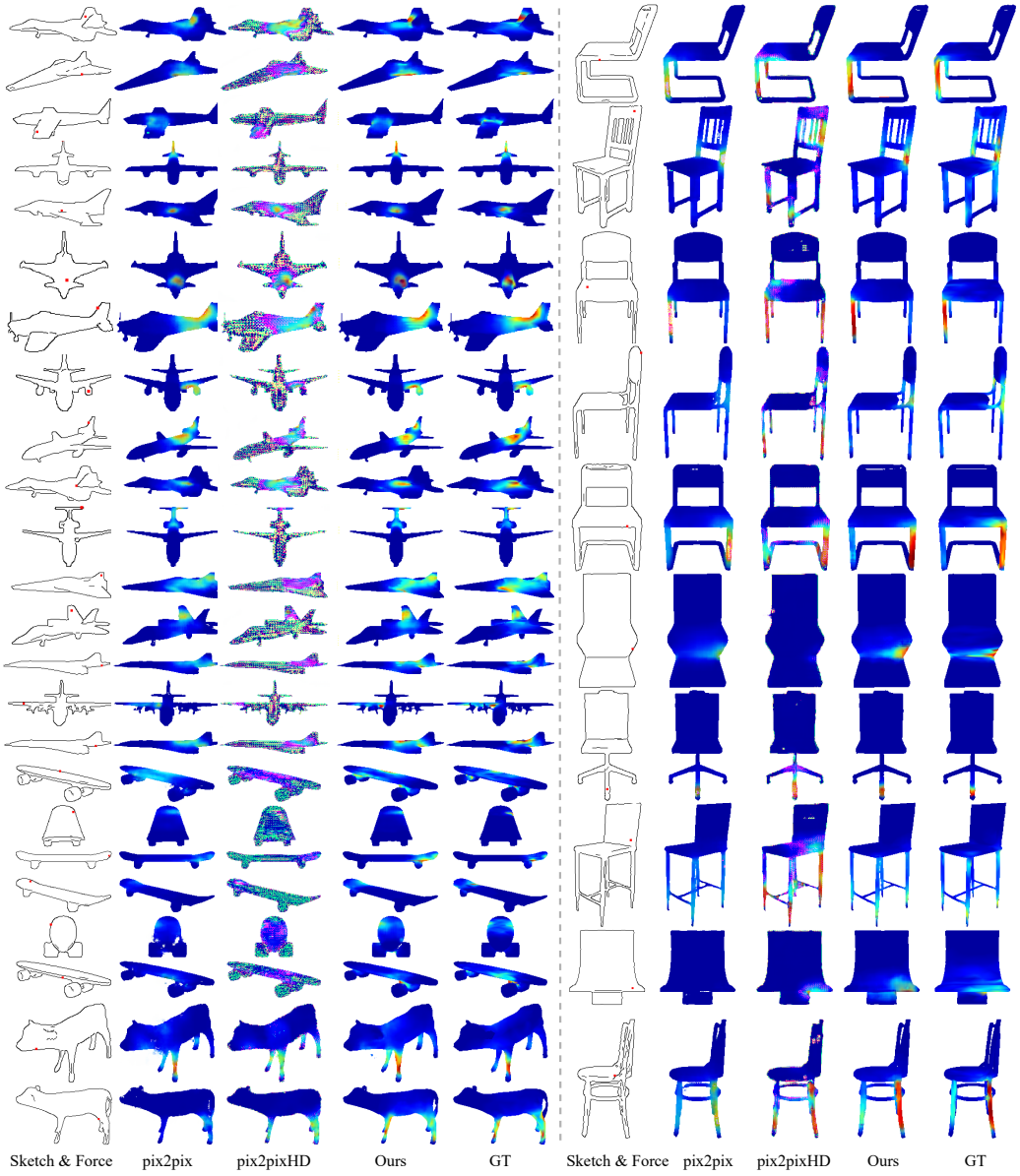


Fig. C.2. Qualitative comparison of results generated by different methods of pix2pix, pix2pixHD, our method, and ground truth. The leftmost column shows the input sketches and the external forces (plotted as red dots on sketches).

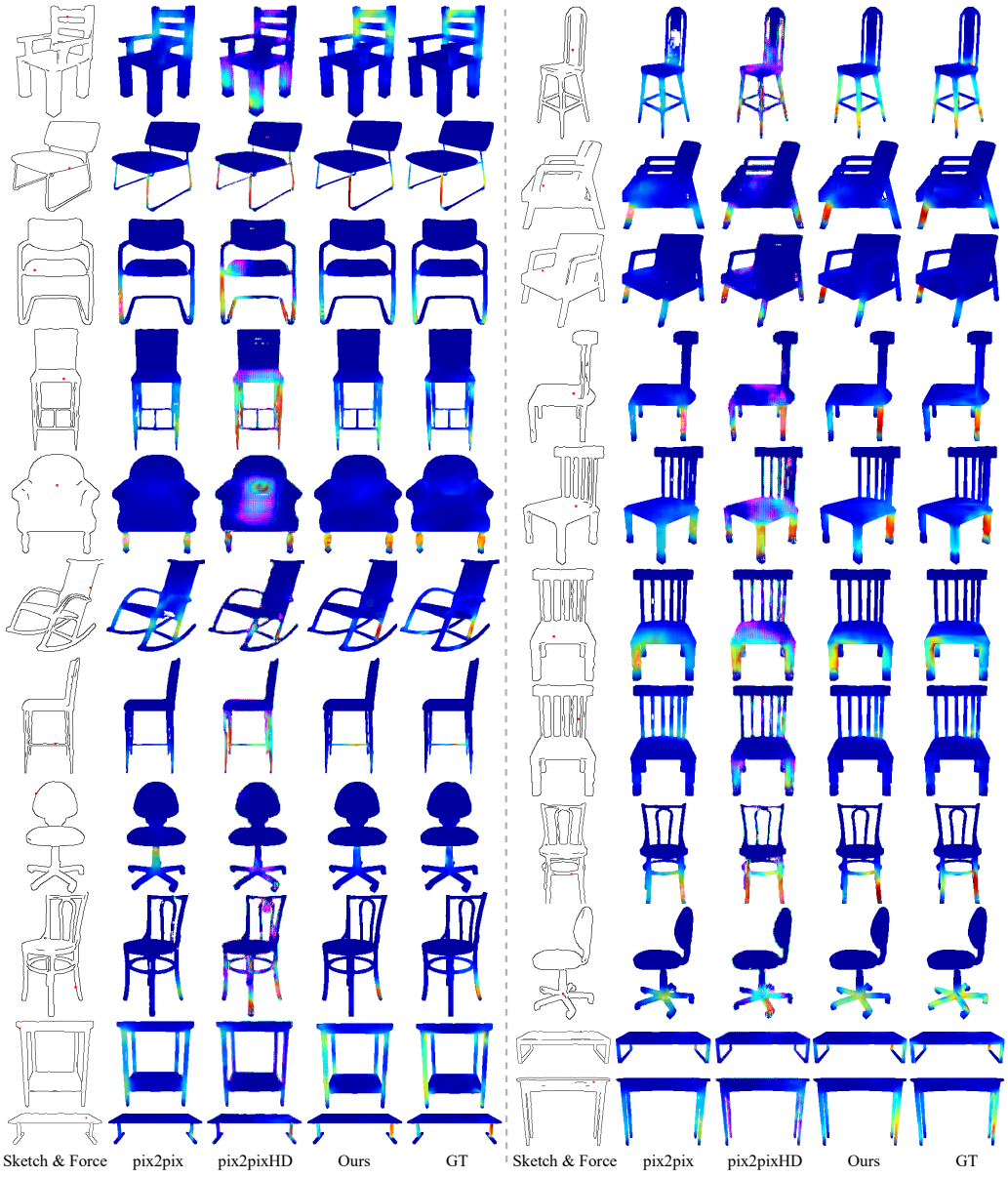


Fig. C.3. Qualitative comparison of results generated by different methods of pix2pix, pix2pixHD, our method, and ground truth. The leftmost column shows the input sketches and the external forces (plotted as red dots on sketches).

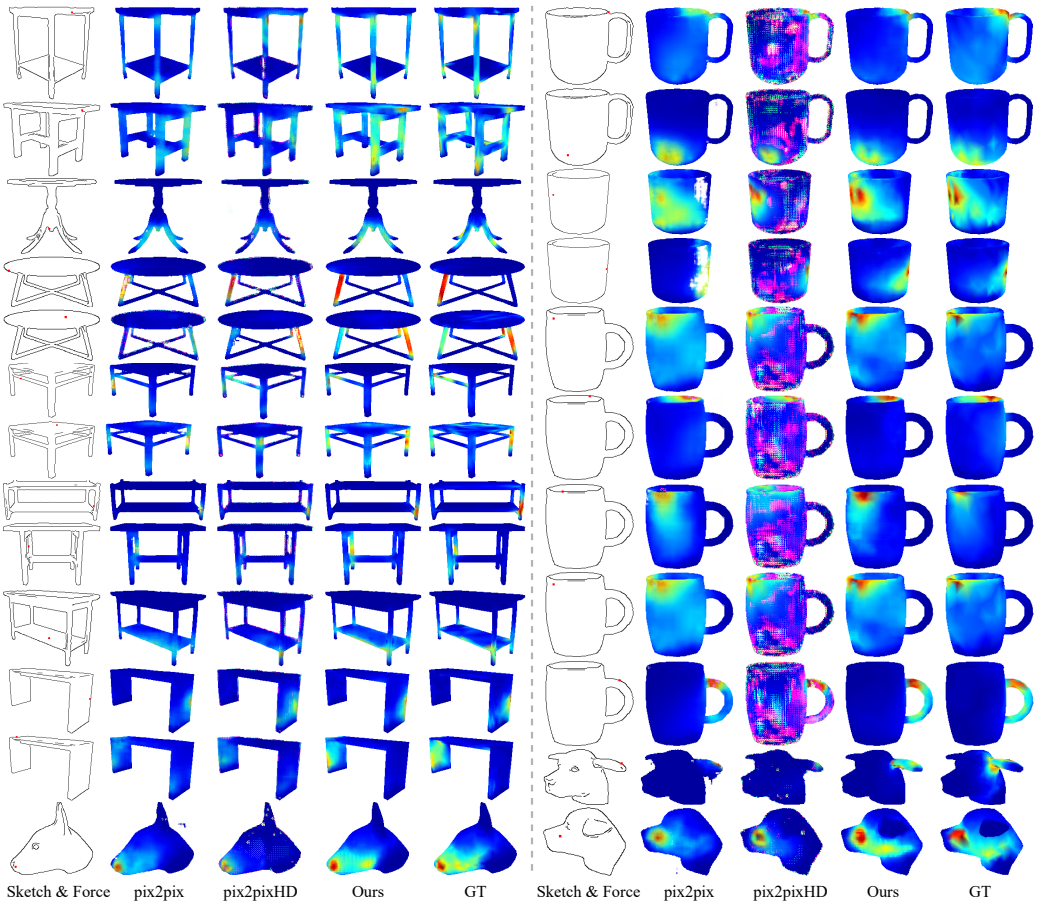


Fig. C.4. Qualitative comparison of results generated by different methods of pix2pix, pix2pixHD, our method, and ground truth. The leftmost column shows the input sketches and the external forces (plotted as red dots on sketches).

D

Correspondence Results of *SketchDesc* Descriptors

This appendix chapter provides more correspondence matching results on multi-view sketches with our *SketchDesc* descriptors. Figures D.1 and D.2 display the performance of our *SketchDesc* on the product sketches created by professional designers and the freehand sketches drawn by volunteers, respectively. And Figure D.3 illustrates the performance of different approaches on the synthesized multi-view sketches rendered from the existing 3D repositories.

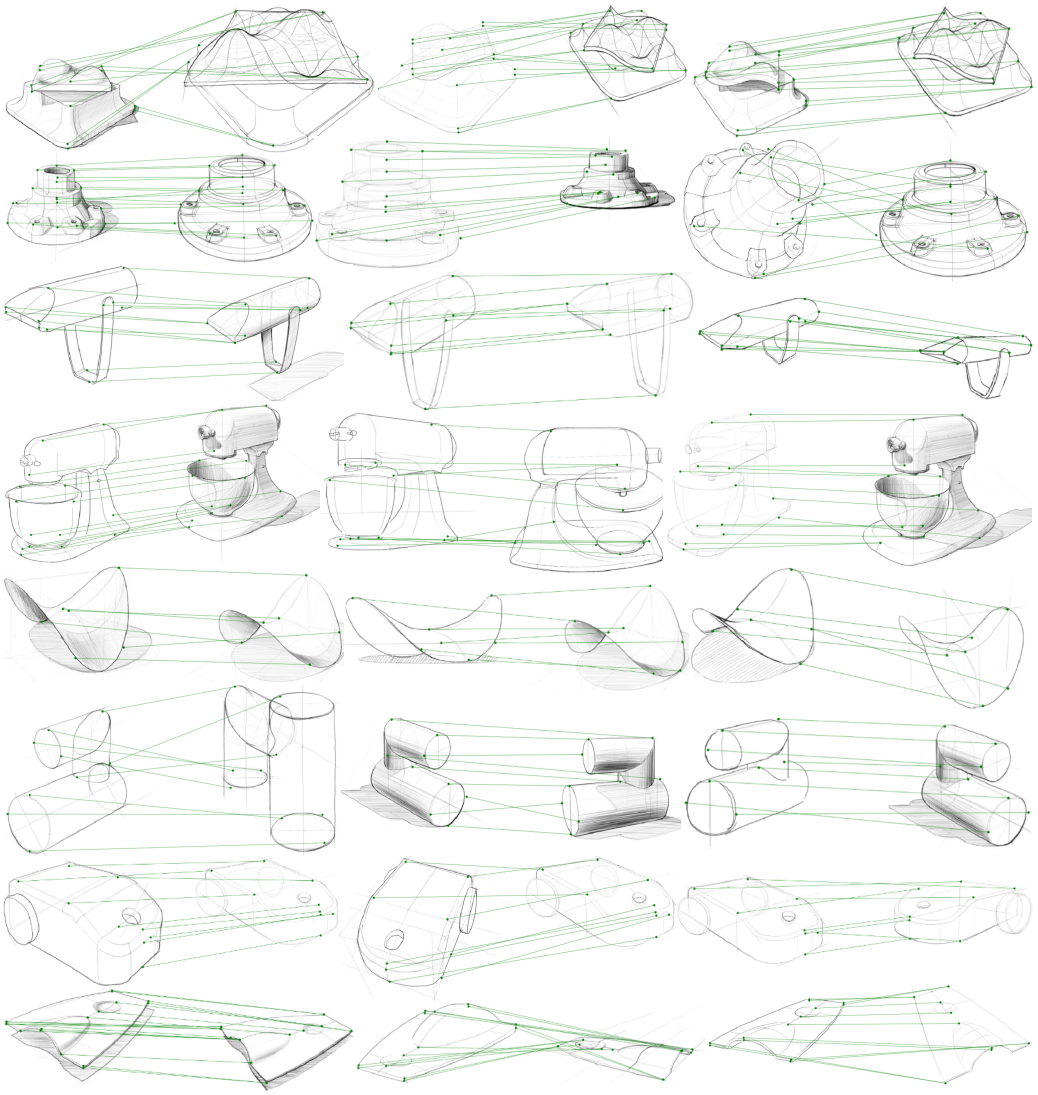


Fig. D.1. Correspondence of multi-view sketches built by *SketchDesc* descriptors. All of the multi-view sketches are from the OpenSketch dataset.

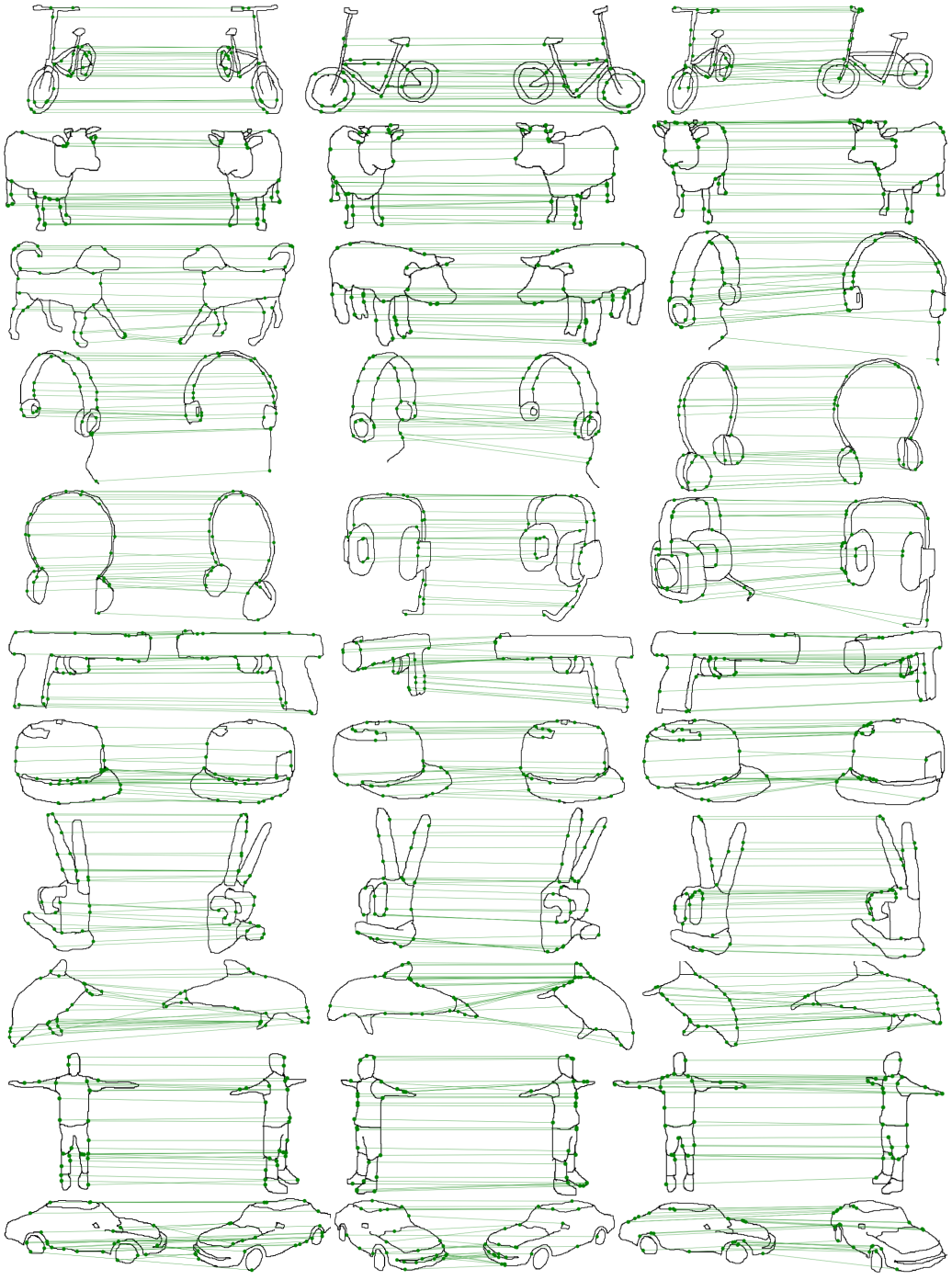


Fig. D.2. Correspondence of multi-view sketches built by *SketchDesc* descriptors. All of these multi-view sketches are created by volunteers.

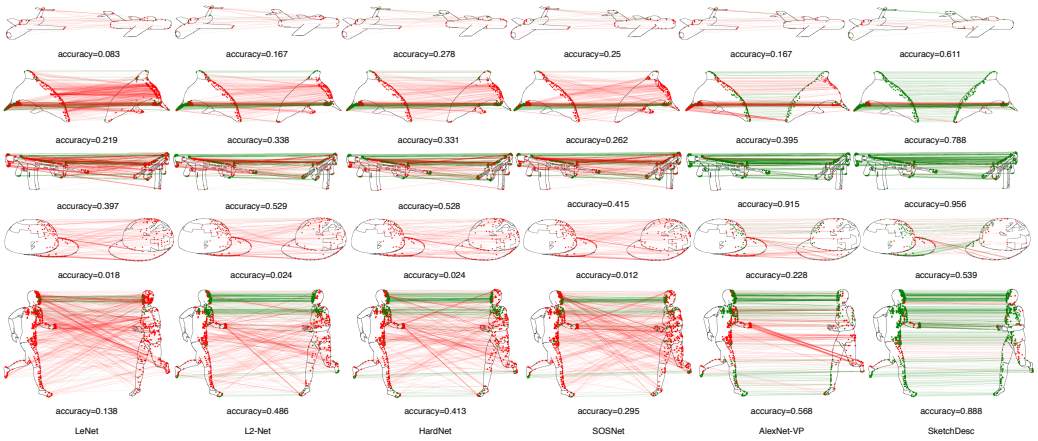


Fig. D.3. Sketch correspondence for multi-view sketches (synthesized). The red lines indicate the failed matching and the green lines show the matching correspondences among multi-view sketches.