3D Highlighter: Localizing Regions on 3D Shapes via Text Descriptions

Dale Decatur
University of Chicago
ddecatur@uchicago.edu

Itai Lang
University of Chicago
itailang@uchicago.edu

Rana Hanocka
University of Chicago
ranahanocka@uchicago.edu

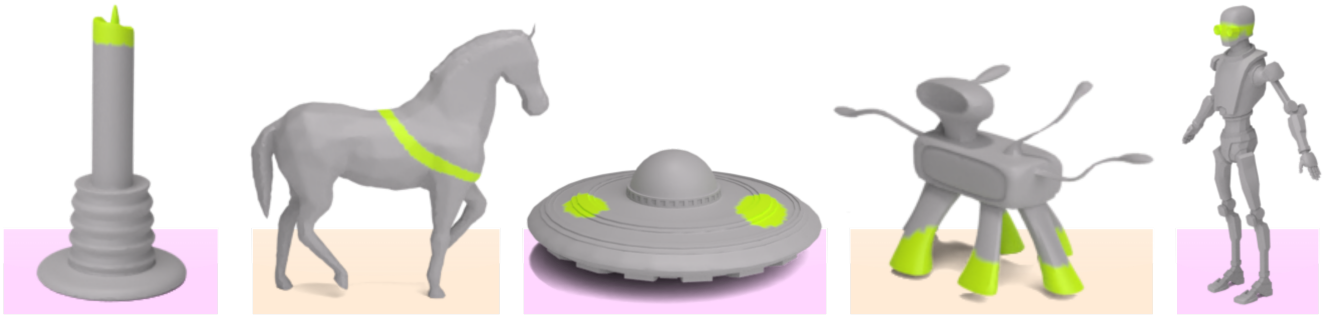

Figure 1. 3D Highlighter localizes semantic regions on a shape using text as input. Our technique reasons about *where* to place seemingly unrelated concepts in semantically meaningful locations on the 3D shape, such as a 'necklace' on a horse or 'shoes' on an alien.

Abstract

We present 3D Highlighter, a technique for localizing semantic regions on a mesh using text as input. A key feature of our system is the ability to interpret "out-of-domain" localizations. Our system demonstrates the ability to reason about where to place non-obviously related concepts on an input 3D shape, such as adding clothing to a bare 3D animal model. Our method contextualizes the text description using a neural field and colors the corresponding region of the shape using a probability-weighted blend. Our neural optimization is guided by a pre-trained CLIP encoder, which bypasses the need for any 3D datasets or 3D annotations. Thus, 3D Highlighter is highly flexible, general, and capable of producing localizations on a myriad of input shapes. Our code is publicly available at https://github.com/threedle/3DHighlighter.

1. Introduction

Semantic localization of regions on 3D meshes is an important problem in computer graphics and vision with broad applications. One such application is the incorporation of semantic information into the 3D modeling process. A particularly challenging aspect of this task emerges when 3D geometric signals are insufficient for performing segmentation, *e.g.* where to add a shirt to a bare 3D human model.

We propose *3D Highlighter*, a method for automatically localizing *fine-grained* semantic regions on a shape based on only a text description. Our system contextualizes the text prompt and *highlights* the corresponding shape region using the network-predicted probabilities. Using *only* text, users are able to semantically identify regions on a shape. Our system takes meshes as input, making it compatible with 3D modeling workflows and tools.

This highlighting task requires both object-level and part-level understanding. 3D Highlighter demonstrates the ability to reason about *where* to place seemingly unrelated concepts on the 3D shape, such as a hat on a candle (Fig. 1). Our system localizes attributes that are *geometrically absent* from a shape, which we refer to as *hallucinated highlighting*. Understanding a part's global shape context is challenging even when relying on salient geometric features [17, 27], let alone without them.

We optimize the weights of a neural network to produce probabilities that are used to color a given 3D shape in accordance with the specified text. We leverage a pre-trained vision-language model (CLIP [31]) to guide the neural optimization towards the text-specified region. This neural optimization formulation is flexible, bypassing the need for any 3D datasets, 3D annotations, or 3D pre-training. Our system is not bound to a specific set of classes, and, as shown in Fig. 2, is not limited to object parts defined by salient geometric features.

We encode the part selection as a *neural field* [44] over the mesh surface. Our network learns to map each point on the surface to a probability of belonging to the text-specified region. We translate the inferred probabilities to a visual at-

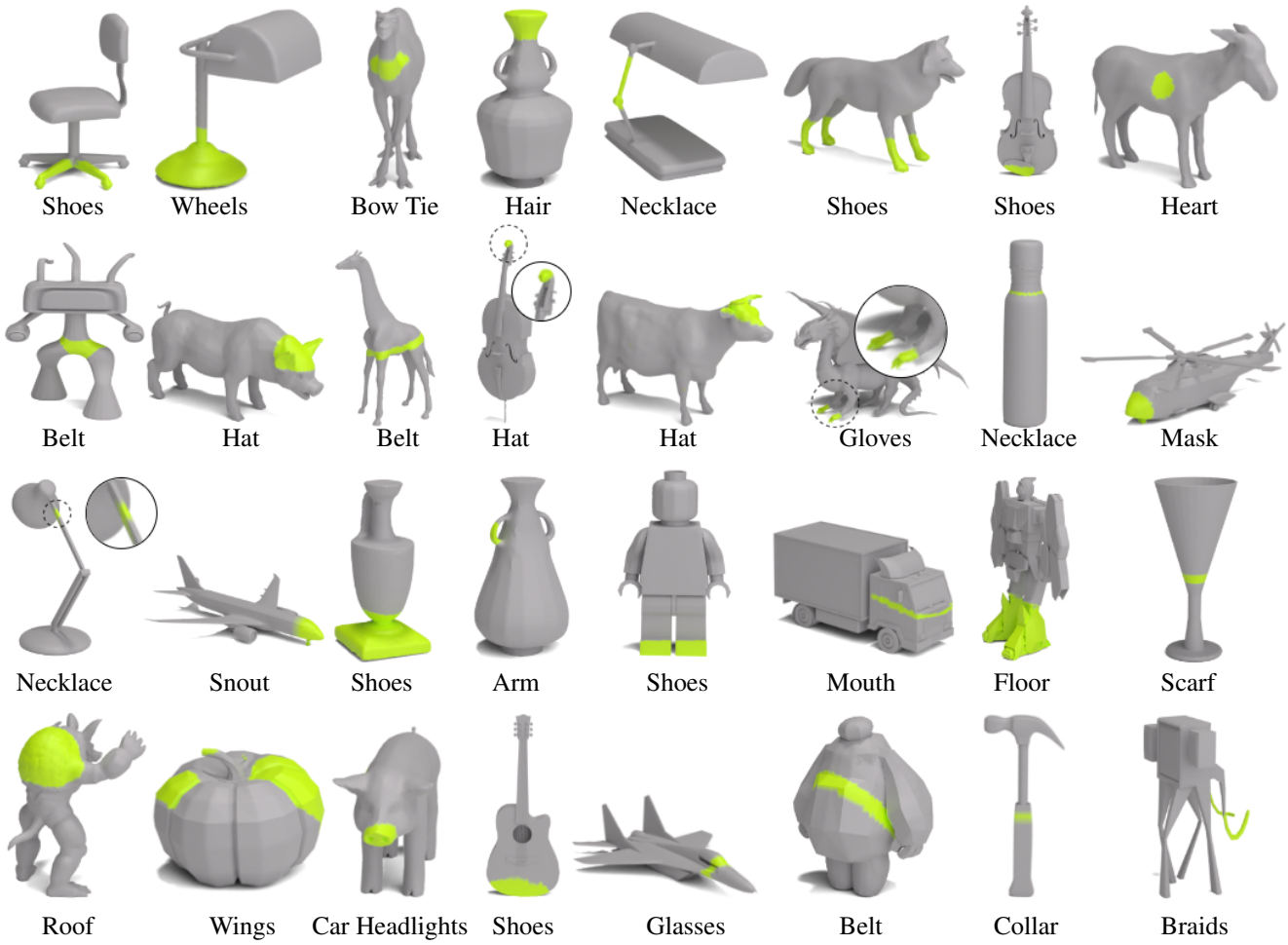

Figure 2. **Hallucinated part highlighting**. Our system is able to reason about *where* to highlight a *geometrically-absent* region on shapes. The resulting localizations demonstrate global understanding and localized part-awareness.

tribute on the mesh surface, which can be rendered and visually understood. The network-predicted probabilities act as a soft-selection operator which *blends* the highlighter color onto the mesh. The network weights are updated by encouraging the CLIP [31] embedding of the 2D renders of the highlighted mesh to adhere to the specified text. As a result, the network implicitly learns to segment the object to adhere to the text prompt.

We make several design choices that are key to the success of 3D Highlighter. Our network does not directly color the mesh. Rather, we predict a *probability of being inside the text-specified highlight*, which is used to blend colors on the mesh. The network is initialized such that points have roughly a 50% probability of being highlighted, resulting in a mesh with albedo halfway between the highlight and background color. During optimization, the relative blend weight of the highlight color directly corresponds to the highlight probability. This blending enables the network to naturally and smoothly increase or decrease the segmentation probability in accordance with the text specification of the target region.

In summary, we present a method for localizing semantic regions on 3D shapes. The localization is specified by a textual description, which is intuitive, flexible, and not limited to a specific training dataset. We demonstrate applications of our method to shape editing and stylization. Furthermore, our field formulation enables the 3D Highlighter to work with different mesh resolutions and triangulations. A key feature of our system is the ability to interpret out-of-domain localizations. For example, 3D Highlighter is able to figure out where to place a 'hat' on a candle as seen in Fig. 1, demonstrating the ability to reason about *where* to place seemingly unrelated concepts on the 3D shape.

2. Related Work

Geometry-driven segmentation. Traditional works in geometry processing use low-level geometric features (such as surface area, curvature, or geodesic distance) in or-

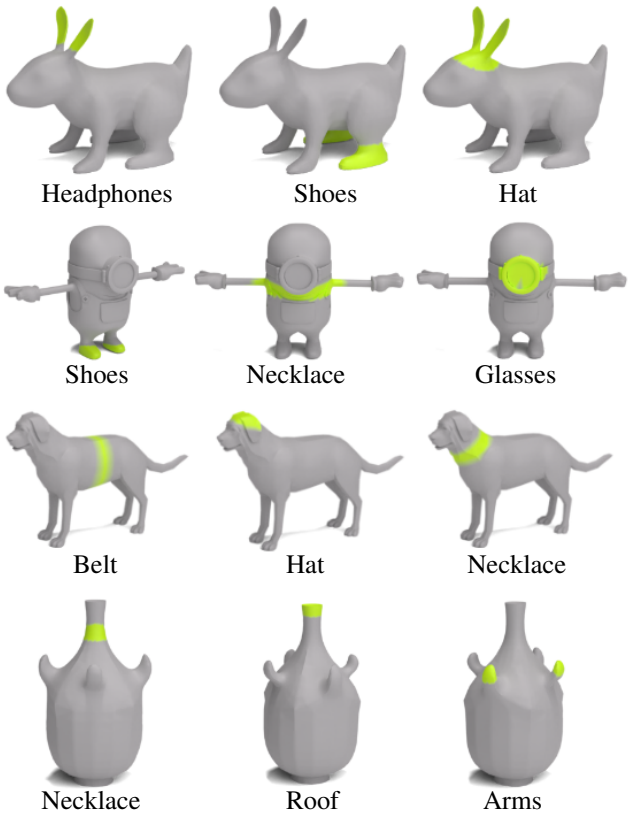

Figure 3. Our method is able to highlight different parts on the same object. For target selections that correspond to distinct regions, 3D Highlighter produces selections that are semantically meaningful and spatially separated without signal from underlying geometry.

der to infer high-level semantic attributes for segmenting shapes [35]. In particular, decomposing shapes into smaller parts or segments often corresponds with physical 3D semantic parts [13, 35]. One approach is to partition shapes based on convexity, or an approximation of convexity [1, 23]. The medial axis carries topological information, which may also be used as a guideline for segmentation [6, 8, 35, 47].

The underlying assumption in these works is that processing the local geometry can be used to understand the semantics for segmentation. By contrast, a key aspect of our work is the ability to perform hallucinated highlights: segmentations that can not necessarily be inferred by geometry alone. See example highlights in Fig. 2 (e.g., localizing a heart on a goat).

Data-driven segmentation. In the deep learning era, the 3D part segmentation task has been widely tackled by neural network models [11, 15, 20, 26, 36, 45]. Training such a model is typically done in a fully-supervised manner on a large dataset of shapes annotated with a given set of part classes. For example, MeshCNN [11] was trained on a human-body segmentation dataset [24] for learning semantic part segmentation. To alleviate the need for 3D annotations, unsupervised learning schemes utilize large collections of unlabelled data [5, 7, 14, 37, 49]. For example, Hong *et al*. [14] inferred part-segmentation through question answering on rendered images from PartNet [46].

In contrast to existing deep learning approaches for shape segmentation, we do not rely on any 3D dataset, nor are we bounded to a specific shape category or set of parts. Instead, we specify the desired localization using text and a pre-trained CLIP model which encompasses rich semantic object understanding. Thus, our 3D Highlighter is capable of localizing various semantic regions on a wide variety of 3D shapes.

Text-guidance. Recent works have leveraged pre-trained vision-language embedding spaces, such as CLIP [31], for analysis, synthesis, and editing. Some techniques leverage pre-trained image encoders for achieving semantic segmentation in images and neural radiance fields [2, 19, 21]. Such techniques are capable of segmenting *entire* objects within a scene based on text, *e.g*., a chair inside a room. However, they may struggle to segment *parts* within an object; *e.g*., failing to distinguish a window (part) from a house (object) [21].

Our work is inspired by the emergent analysis in text-driven synthesis techniques for 3D data [10, 16, 18, 25, 30, 42]. Specifically, Text2Mesh [25] devised a framework for text-driven stylization of 3D meshes, observing that the resulting textures consider part-aware semantics. Yet, since Text2Mesh directly synthesizes stylizations, there is no obvious way to extract any underlying semantic analysis. To address this, we opt to use a highlighter color only as a means for visualizing the network-predicted segmentations.

3. Method

An illustration of our method is shown in Fig. 5. The inputs to our system are a mesh M, represented by vertices $V \in \mathbb{R}^{n\times 3}$ and faces $F \in \{1, ..., n\}^{m\times 3}$, and a text description T. Our neural network, referred to as *neural highlighter*, is optimized to map vertex positions $v \in V$ to a

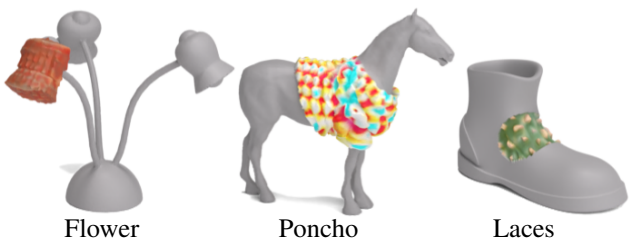

Figure 4. **Localized editing.** We incorporate textures and displacements to a region highlighted with 3D Highlighter. Used styles: Brick (left), Colorful Crochet (middle), Cactus (right).

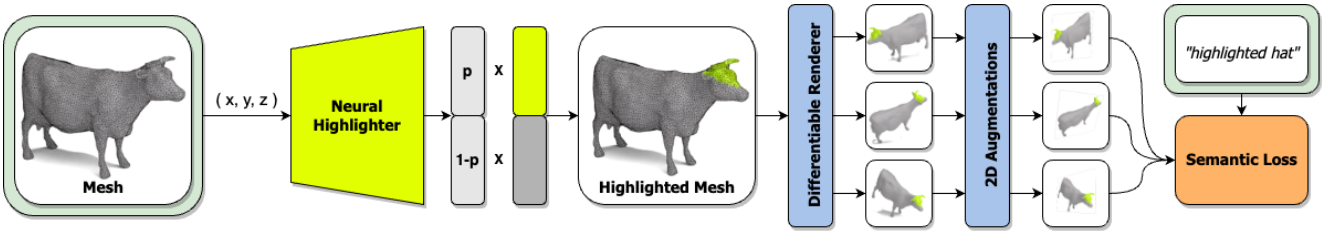

Figure 5. **Overview of 3D Highlighter.** The Neural Highlighter maps each point on the input mesh to a *probability*. The mesh is colored using a probability-weighted blend and then rendered from multiple views. The neural highlighter weights are guided by the similarity between the CLIP embeddings of the 2D augmented images and the input text.

probability p of belonging to the text-specified region. Each vertex on the mesh is colored according to a probability-weighted blend between the highlighter color and a gray background color. The resulting highlighted mesh M' is rendered from multiple views, and we apply 2D augmentations to obtain a set of images. We supervise the network optimization by comparing the CLIP-embedded images to the CLIP embedding of the desired text.

3.1. Neural Highlighter

Our neural highlighter is a neural field [44] mapping coordinates $\mathbf{x} \in \mathbb{R}^3$ to $p \in [0, 1]$, where p is the probability that $\mathbf{x}$ belongs to the text-specified region. The neural highlighter is represented as a multi-layer perceptron (MLP) $\mathcal{F}_\theta$ that takes an input vertex v in the form of a 3D coordinate $\mathbf{x}_v = (x, y, z)$ and predicts a highlight probability p_v, $\mathcal{F}_\theta(\mathbf{x}_v) = p_v$. This formulation allows us to query the neural field to obtain meaningful highlight probabilities for any 3D point on (or near) the mesh surface. Thus, once optimized, the network weights conveniently transfer the localization to different meshings of the same object without requiring further optimization (Fig. 9).

Representing our neural highlighter as an MLP produces contiguous localizations and reduces artifacts. MLPs have been shown to exhibit a spectral bias towards smooth solutions [32], especially on low-dimensional inputs such as 3D coordinates [38]. The bias towards low-frequency outputs encourages our 3D Highlighter to predict contiguous localizations with sharp boundaries and discourages noisy highlights (Fig. 7). For this reason, our approach does not utilize positional encoding. See supplemental material for additional details.

Figure 6. **Viewpoint robustness.** Our system produces consistent results even when using different *primary* viewpoints. Results for three different primary viewpoints for the target text 'necklace'.

3.2. Mesh Color Blending

We leverage the per-point highlight probability to color the mesh in a continuous, differentiable manner, generating semantically meaningful renders for CLIP supervision. We use a probability-weighted blend, where each vertex color C_v is a linear combination of the highlight color H and gray color G weighted by the network-predicted highlight probability $C_v = p_v \cdot H + (1 - p_v) \cdot G$.

At the start of the optimization process, all vertex probabilities are initialized near 0.5 and thus the entire mesh is half-highlighted. As the optimization progresses, vertices smoothly transition towards gray or highlighter color (based on the network predictions) such that vertices predicted to be highlighted adhere to the text-specified region. This formulation translates each step of the optimization to a colored mesh that is semantically meaningful to CLIP. Our method provides continuous gradients, in contrast to coloring vertices according to the argmax of the highlight probability. Our blending scheme results in a smoother optimization landscape and reduces highlight artifacts (Fig. 7).

This formulation is also important for downstream applications that wish to use the localizations, *e.g.* editing and stylization. Predicting per-point highlight probabilities provides an explicit representation of the highlight region on the mesh surface. An alternative approach, optimizing the surface color directly, would only provide a visual result without explicit information about which vertices belong to the localization.

3.3. Unsupervised Guidance

We guide our neural optimization using the joint vision-language embedding space of CLIP [31]. We formulate the desired highlight by describing the association between the input mesh [object] and target localization [region]. Specifically, we design our target text T to be: "a gray [object] with highlighted [region]." We render the highlighted geometry from multiple views using differentiable rendering [4]. At each optimization step, we randomly sample n views from a Gaussian distribution centered around a primary view. This

ensures that the underlying object is recognizable in the majority of views shown to CLIP.

In a preliminary viewpoint prediction stage, we render 360° views of the mesh and measure the CLIP similarity to the target text prompt. We select the primary view to be the render with the highest CLIP similarity. We found that there exist many possible viewpoints which produce desirable highlighter results (see Fig. 6). More details about how the primary view is selected can be found in the supplemental material.

For each view ψ, we render a 2D image I_ψ and apply a random perspective 2D augmentation ϕ, as done in previous works [9, 25]. We then encode each of the augmented images into the CLIP embedding space (in $\mathbb{R}^{768}$) using CLIP's image encoder, denoted as E_I. Our final aggregate image representation e_I is the average CLIP encoding over all views:

$$\mathsf{e}_I = \frac{1}{n} \sum_{\psi} E_I(\phi(I_\psi)) \in \mathbb{R}^{768}. \tag{1}$$

Similarly, we encode the target selection text T with CLIP's text encoder E_T to get the encoded target representation $\mathsf{e}_T = E_T(T) \in \mathbb{R}^{768}$. Our loss $\mathcal{L}$ for optimizing the neural highlighter parameters θ is formulated as the negative cosine similarity between the aggregate image embedding and the text embedding:

$$\underset{\theta}{\operatorname{argmin}}\, \mathcal{L}(\theta) = -\frac{\mathsf{e}_I \cdot \mathsf{e}_T}{|\mathsf{e}_I| \cdot |\mathsf{e}_T|}. \tag{2}$$

When the loss is minimized, the CLIP embedding of the rendered highlighted mesh becomes similar to the target text embedding. Thus, the localized region will reflect the target text region.

4. Experiments

In this section we examine various capabilities of 3D Highlighter. First, we demonstrate the fidelity of our highlighter localization in Sec. 4.1, including qualitative and quantitative evaluations. As far as we can ascertain, our method is the first technique to perform text-driven localization on 3D shapes without pre-training on 3D data. Thus, we adapt an existing language-guided segmentation technique for 2D images to serve as a baseline [21]. Moreover, we demonstrate the robustness of 3D Highlighter in Sec. 4.2. Then we explore several applications of our method in Sec. 4.3, such as selective editing, localized manipulation, and segmentation. Finally, in Sec. 4.4 we evaluate the influence of key components of 3D Highlighter and discuss its limitations in Sec. 4.5.

We apply our method to a large variety of meshes from different sources: COSEG [41], Turbo Squid [40], Thingi10K [48], Toys4k [34], ModelNet [43], and ShapeNet [3]. 3D Highlighter does not impose any restrictions on the mesh quality; many of the meshes used contain artifacts, such as elements that are non-manifold, unoriented, and contain boundaries or self-intersections. Our PyTorch [29] implementation optimization takes around 5 minutes to run on an Nvidia A40 GPU. In our experiments, we used CLIP ViT-L/14 at 224×224 resolution.

4.1. Generality and Fidelity of 3D Highlighter

Highlight generality. 3D Highlighter is not restricted to any particular category for either the input mesh or the text-specified localization, since it does not rely on a 3D dataset or 3D pre-training. In Fig. 2, we see our method achieves accurate localization for a diverse collection of meshes from various domains such as humanoids, animals, and manufactured objects. 3D Highlighter is capable of localizing a wide variety of diverse attributes even when the context of these target attributes is entirely unrelated to the input mesh. Moreover, 3D Highlighter demonstrates that it can perform *hallucinated highlighting*, where it selects regions on meshes with no underlying geometric signal (such as a bow tie on a camel or a hat on a pig).

Highlight specificity. In Fig. 3, we observe that semantic differences are reflected in the network-predicted highlight. 3D Highlighter is able to successfully localize different text-specified regions on the same mesh. Our framework demonstrates the nuanced understanding required to disambiguate different target regions, such as headphones and hat on the rabbit. Finally, the ability to identify many different regions on a single mesh allows users intuitive, comprehensive, and fine-grained control over part localization.

Quantitative evaluation. 3D Highlighter is the first system to select semantic regions on 3D shapes using text guidance, without any 3D datasets. Since there are no quantitative benchmarks to evaluate the quality of our highlights, we do so with a perceptual user study.

Moreover, since there are no existing approaches for text-based segmentation in 3D, we create two baselines by

full	*direct*	*no blend*	*no augs*
0.332	0.319	0.297	0.287

Figure 7. **Ablation experiments.** We present ablation results for target text 'shoes' using our system (***full***), direct optimization (*direct*), without probability-weighted blending (*no blend*), and without 2D augmentations (*no augs*). Resulting CLIP scores shown below each image.

Method	Control	LSeg	Text2LIVE	Ours
Average Score ↑	1.00	1.26	2.23	**4.38**

Table 1. **Perceptual study.** We extend two image-based approaches LSeg [21] (segmentation) and Text2LIVE [2] (localized editing) to the highlighting task and report mean user rating.

extending two different 2D image-based approaches. The first baseline extends LSeg [21] which directly predicts a segmentation in 2D, while the second baseline extends Text2LIVE [2] which infers an edit mask for 2D image manipulation. To evaluate these baselines, we render a bare mesh from a view where the target localization region is clearly visible. We extract the 2D segmentation produced by the image baselines and use it to color the rendered image. Then we ask users to rate the highlight quality of both baselines and our 3D Highlighter result rendered from the same view in our perceptual study.

Our perceptual study reports quantitative results on the quality of highlights from both 3D Highlighter and baselines. Users were asked to rate each result from 1-5 on how effectively the highlight represents "an [object] with a region corresponding to a [region] highlighted." Visual examples from our study are shown in the supplemental material (Fig. 21). In total, 33 users evaluated each method on 5 mesh and region combinations.

Our 3D Highlighter achieved the highest ratings compared to the baselines (Tab. 1). LSeg is built for text-driven semantic segmentation and excels at segmenting entire objects within a scene. However, LSeg struggles to identify parts within a single object, leading to subpar performance on our highlighting task. Text2LIVE was not explicitly built for segmentation, however it does rely on inferring a continuously-valued edit mask (*i.e.* a soft-segmentation) when performing localized image editing. The edit mask is designed to produce high-quality image manipulations; however, it is not directly suitable for identifying the sharp segmentation boundaries required for our highlighting task. Qualitative comparisons and an additional quantitative comparison using a modified CLIP R-Precision metric are discussed in the supplemental material.

4.2. Robustness of 3D Highlighter

Localization transfer. An important benefit of formulating 3D Highlighter as a neural field optimization is the ability to trivially transfer localization results between different meshings. This ability is useful for many tasks in geometry processing which require an object to be re-triangulated, simplified, subdivided, or otherwise remeshed. Localization transfer is possible since our neural highlighter is represented as a field over the shape and is independent of any

Figure 8. **Controlled stylization.** Given three different stylizations of the same object, we use 3D Highlighter to select different regions and combine them together (Ours). Attempting to achieve this composition with a holistic approach leads to an undesirable result (Text2Mesh [25]).

specific meshing. Although the neural highlighter is trained on mesh vertices, the resulting network encodes a smooth field and produces meaningful outputs for any 3D point on (or near) the mesh surface.

In Fig. 9, we show an optimization of the 3D Highlighter on a single mesh triangulation (original) for the prompt 'shoes'. We then apply the already-optimized neural highlighter to remeshed (middle) and subdivided (right) versions of the original mesh, showing the transferability of the selected region to different triangulations. This result demonstrates how 3D Highlighter is independent of the input mesh and that, once we have a localization for one mesh, we can trivially transfer it to any other meshing of the same object.

Viewpoint robustness. Our method is robust to the primary view choice. This property is important for our localization task, as we may not know *a priori* which view is ideal. In Fig. 6, we perform our optimization using three different primary viewpoints: $0°$, $90°$, and $-90°$ (viewpoints shown in blue). We then present predicted localizations, showing that for all three views, 3D Highlighter is able to accurately identify the target localization region, regardless of whether that region is visible from the primary view.

From the $-90°$ primary view, the target region (the neck) is not visible. However, is is still visible with a low probability for views sampled from the Gaussian distribution

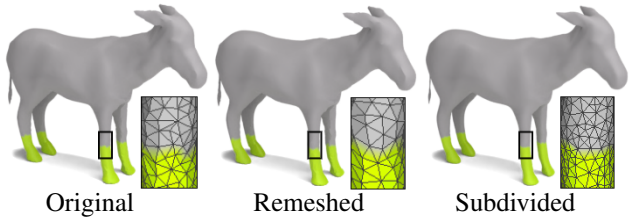

Figure 9. **Localization transfer.** We optimize our neural highlighter on one mesh (original) for the prompt 'shoes'. Once optimized, the network weights transfer the localization to different meshings of the same object (remeshed and subdivided).

around the primary view. This means that over the course of optimization, regions other than the neck are mostly seen while the target region is rarely visible. Nonetheless, our method manages to highlight the desired region, which implies its robustness to how frequently the target region for localization is seen. Furthermore, it shows that oversampling views where the target region is not visible does not negatively influence the optimization.

4.3. Applications of 3D Highlighter

Selective editing. In Fig. 4, we show that it is possible to use 3D Highlighter to selectively edit a 3D object within a semantic region. This is applicable to techniques which incorporate global texture or material properties over the entire shape, such as in Text2Mesh [25] or MatCap [39]. Starting with different bare input meshes, we edit the entire shape using a global stylization technique [25]. Then, we use 3D Highlighter to select a text-specified region and incorporate the modifications *only* in the selected area. Thus 3D Highlighter provides direct control over where to stylize shapes, enabling users to obtain localized stylizations based on semantic cues.

Controlled stylization via composition. Achieving compositionality with language models is a challenging task [33]. For example, starting with a human mesh and using Text2Mesh [25] to stylize '*Iron Man with the head of Steve Jobs and Yeti legs*', leads to muddled and undesirable results (Fig. 8, rightmost). Our method enables compositionality between different shape modifications by chaining simple concepts together (Fig. 8). Specifically, we decompose the desired modification into three separate attainable targets ('Iron Man', 'Steve Jobs', and 'Yeti'), which we stylize individually with Text2Mesh. We then utilize our 3D Highlighter to localize the text-specified regions. We achieve the desired composition by combining the highlighted regions together, obtaining clear boundaries between stylizations.

Semantic segmentation. In Fig. 10, we show that our technique is not restricted to *hallucinated highlighting* and is capable of localizing semantically-specified geometric regions. These text-driven localizations identify unique geometric parts *without* utilizing any 3D datasets or part labels.

4.4. Components of 3D Highlighter

Ablation study. Several components are key for facilitating 3D Highlighter. We provide ablation results in Fig. 7 to demonstrate the effect of our design choices. First, using a direct optimization of the vertex color (*direct*) instead of optimizing a neural field results in splotchy highlight artifacts. Since the neural field has a spectral bias towards smooth solutions [32], omitting it leads to an undesired noisy output. Second, removing the probability weighted blending (*no blend*) and instead coloring vertices using only

Figure 10. **Semantic Segmentation.** 3D Highlighter produces semantic segmentations for unique geometric parts *without* any 3D dataset or annotations.

two distinct values also produces a noisy highlight pattern. Without a continuous color blend, the gradients become ill-conditioned and unstable, leading to highlight artifacts and irregular localization boundaries. Lastly, similar to previous works [9, 25], we observe that without 2D perspective augmentations (*no augs*), 3D Highlighter outputs degenerate solutions. The ablation study emphasizes the importance of our key design choices in 3D Highlighter for its ability to highlight a coherent and localized region on the input shape.

Prompt formulation and CLIP understanding. Our prompt formulation combined with our coloring scheme results in the correct association between objects and their properties, a known challenge when using CLIP [33]. In Fig. 12, we analyze the CLIP score for two different prompts: '*gray chair with highlighted back*' (left) and '*blue chair with red back*' (right). For each prompt, we measure the CLIP similarity to renders of both the correct assignment and flipped assignment.

We observe that our prompt formulation ('*gray chair with highlighted back*') results in a higher average CLIP score for the correct assignment. In contrast, when specifying colors in the prompt ('*blue chair with red back*') and styling the mesh accordingly, we see *higher* CLIP scores for the *flipped* association. Using the same gray and yellow renders (left), we also compare to a prompt specifying colors ('gray chair with yellow back') and find that the higher

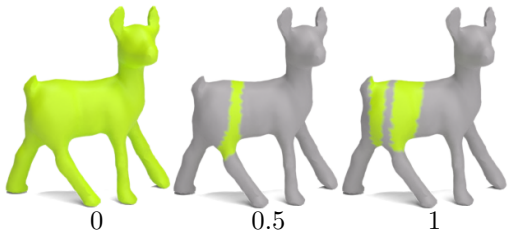

Figure 11. **Network initialization**. We optimize 3D Highlighter for the text prompt 'belt' using different initialization methods: using a default initialization where all output probabilities are near 0.5 (middle) or altering the final layer so that all outputs are 0 (left) or 1 (right). Initializing with 0 or 1 leads to an undesirable result.

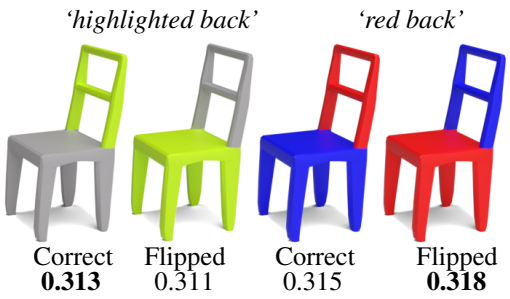

Figure 12. **CLIP understanding.** We examine CLIP similarity scores for several prompt formulations targeting the '*back*' of the chair while using the correct color assignment and where the coloring is flipped. For the prompt '*gray chair with highlighted back*' (left) we observe that the CLIP score is higher for the **correct** assignment. For the the prompt '*blue chair with red back*' (right) the CLIP score is higher for the **flipped** (incorrect) assignment.

CLIP score corresponds to the flipped selection (data not shown).

We also measure the CLIP scores for our standard prompt formulation: 'gray chair with highlighted back', replacing the yellow color in the rendering with other colors, such as red and blue, and find that the correct selection has a higher CLIP score (data not shown). To conclude, our prompt formulation (i.e., the use of the term 'highlighted') coincides with CLIP's understanding and 3D Highlighter is robust to the highlight color.

Network initialization. Initializing the network such that the object is partially highlighted (i.e., with highlight probability equal to 0.5) is important for obtaining desirable results. In Fig. 11, we show the optimization of our method for the target text prompt 'belt' using three different initializations. Our method (middle) initializes all output probabilities near 0.5 by random weight initialization of the network. We compare to initializing the output probabilities to 0 (left) or 1 (right), in which we set the weights of the last layer to 0, and the bias to 0 or 1, respectively.

For the initialization to both 0.5 and 1, a highlight color is uniformly present on the styled mesh, whereas with 0, the mesh is gray with no highlight. Consequently, we hypothesize that the presence of highlight color at initialization is important for CLIP's supervision.

4.5. Limitations

3D Highlighter is robust to variations of the object specification in the target prompt. However, there should still be a logical connection between the 3D shape and its description. Fig. 13 shows results for a camel mesh and the target highlight 'shinguards'. For each optimization, we use a slightly different target prompt by varying the *object* specification. The prompts are of the form "[*object*] *with highlighted shinguards*", where [*object*] is replaced with *camel*, *pig*, *animal*, or *chair*.

In Fig. 13, we observe that with object specifications that resemble the geometry of camel, such as pig and animal, 3D Highlighter accurately localizes the desired region. However, for a description that is incompatible with the object's geometry (i.e., referring to a camel as a chair), our method does not produce meaningful results. This result sheds light on 3D Highlighter's robustness to text descriptions: 3D Highlighter is able to reason about a mesh even when its description is not perfectly accurate, provided that it is sufficiently similar to the true description (i.e., referring to a camel mesh as a pig).

5. Conclusions

We present a technique for *highlighting* semantic regions on meshes using text as input, without any 3D datasets or 3D pre-training. 3D Highlighter can *reason* about where to place a non-obviously related part on a 3D object (i.e. a hat on a candle). The ability to combine unconnected parts and objects together is reminiscent of ideas from *image analogies* [12, 22]. In this work, we show that we can identify part-concepts that are *geometrically absent* from a shape, giving rise to our *hallucinated highlighting* capability.

During neural optimization, our neural network infers a *probability* which we use to blend the highlight color onto the mesh. The network-predicted probabilities are general, and provide a soft-segmentation which we show can be used for a variety of different applications (Figs. 4 and 8). In the future, we are interested in extending our framework to obtain part correspondence between shapes that differ topologically but are semantically related.

6. Acknowledgments

We thank the University of Chicago for providing the AI cluster resources, services, and the professional support of the technical staff. This work was also supported in part by gifts from Adobe Research. Finally, we would like to thank Richard Liu, Avery Zhou, and the members of 3DL for their thorough and insightful feedback on our work.

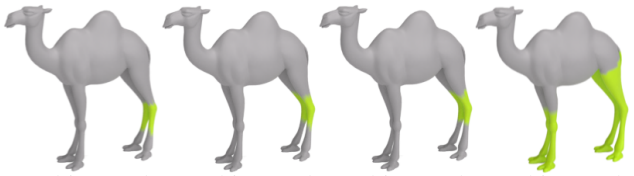

Figure 13. **Prompt generality**. Our system is robust to certain variations in object specifications. We achieve desirable results for the text input '*camel with highlighted shinguards*' (left), as well as for other variations ('*pig*' and '*animal*'). If the object specification, such as '*chair*', is incompatible with the input geometry, 3D Highlighter no longer produces meaningful results.

References

[1] Shmuel Asafi, Avi Goren, and Daniel Cohen-Or. Weak convex decomposition by lines-of-sight. *Computer graphics forum*, 32(5):23–31, 2013. 3

[2] Omer Bar-Tal, Dolev Ofri-Amar, Rafail Fridman, Yoni Kasten, and Tali Dekel. Text2live: Text-driven layered image and video editing. *arXiv preprint arXiv:2204.02491*, 2022. 3, 6, 12, 13

[3] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 5

[4] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[5] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8490–8499, 2019. 3

[6] Nicu D Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *IEEE Transactions on visualization and computer graphics*, 13(3):530, 2007. 3

[7] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. 3

[8] Tamal K Dey and Wulue Zhao. Approximating the medial axis from the voronoi diagram with a convergence guarantee. *Algorithmica*, 38(1):179–200, 2004. 3

[9] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *ArXiv*, abs/2106.14843, 2021. 5, 7

[10] Rao Fu, Xiao Zhan, Yiwen Chen, Daniel Ritchie, and Srinath Sridhar. Shapecrafter: A recursive text-conditioned 3d shape generation model. *arXiv preprint arXiv:2207.09446*, 2022. 3

[11] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: A network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):90:1–90:12, 2019. 3

[12] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 327–340, 2001. 8

[13] Donald D Hoffman and Whitman A Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984. 3

[14] Yining Hong, Yilun Du, Chunru Lin, Josh Tenenbaum, and Chuang Gan. 3d concept grounding on neural fields. In *Annual Conference on Neural Information Processing Systems*, 2022. 3

[15] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Junxiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R. Martin. Subdivision-based mesh convolution networks. *ACM Trans. Graph.*, 41(3):25:1–25:16, 2022. 3

[16] Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 867–876, 2022. 3, 12

[17] Oliver Van Kaick, Noa Fish, Yanir Kleiman, Shmuel Asafi, and Daniel Cohen-Or. Shape segmentation by approximate convexity analysis. *ACM Transactions on Graphics (TOG)*, 34(1):1–11, 2014. 1

[18] Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Popa Tiberiu. Clip-mesh: Generating textured meshes from text using pretrained image-text models. *SIGGRAPH Asia 2022 Conference Papers*, December 2022. 3

[19] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. *arXiv*, 2022. 3

[20] Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020. 3

[21] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *International Conference on Learning Representations*, 2022. 3, 5, 6, 12, 13

[22] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *arXiv preprint arXiv:1705.01088*, 2017. 8

[23] Jyh-Ming Lien and Nancy M Amato. Approximate convex decomposition of polyhedra. In *Proceedings of the 2007 ACM symposium on Solid and physical modeling*, pages 121–131, 2007. 3

[24] Haggai Maron, Meirav Galun, Noam Aigerman, Miri Trope, Nadav Dym, Ersin Yumer, Vladimir G Kim, and Yaron Lipman. Convolutional neural networks on surfaces via seamless toric covers. *ACM Trans. Graph.*, 36(4):71–1, 2017. 3

[25] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 3, 5, 6, 7, 11

[26] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:952–963, 2020. 3

[27] Kaichun Mo, Shilin Zhu, Angel X. Chang, Li Yi, Subarna Tripathi, Leonidas J. Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding, 2018. 1

[28] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 12

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 5

[30] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv*, 2022. 3

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 1, 2, 3, 4

[32] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5301–5310. PMLR, 09–15 Jun 2019. 4, 7

[33] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 7

[34] James Matthew Rehg. Toys4k 3d object dataset, 2022. https://github.com/rehg-lab/lowshot-shapebias/tree/main/toys4k. 5

[35] Ariel Shamir. A survey on mesh segmentation techniques. *Computer graphics forum*, 27(6):1539–1556, 2008. 3

[36] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. Diffusionnet: Discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 3

[37] Weiwei Sun, Andrea Tagliasacchi, Boyang Deng, Sara Sabour, Soroosh Yazdani, Geoffrey E Hinton, and Kwang Moo Yi. Canonical capsules: Self-supervised capsules in canonical pose. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24993–25005. Curran Associates, Inc., 2021. 3

[38] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 4, 12

[39] Hideki Todo, Ken Anjyo, and Shun'Ichi Yokoyama. Lit-sphere extension for artistic rendering. *Vis. Comput.*, 29(6–8):473–480, jun 2013. 7

[40] TurboSquid. Turbosquid 3d model repository, 2021. https://www.turbosquid.com/. 5

[41] Oliver van Kaick, Andrea Tagliasacchi, Oana Sidi, Hao Zhang, Daniel Cohen-Or, Lior Wolf, and Ghassan Hamarneh. Prior knowledge for part correspondence. *Computer Graphics Forum*, 30(2):553–562, 2011. 5

[42] Can Wang, Menglei Chai, Mingming He, Dongdong Chen, and Jing Liao. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3835–3844, 2022. 3

[43] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015. 5

[44] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. 1, 4, 12

[45] Li Yi, Hao Su, Xingwen Guo, and Leonidas J Guibas. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2282–2290, 2017. 3

[46] Fenggen Yu, Kun Liu, Yan Zhang, Chenyang Zhu, and Kai Xu. Partnet: A recursive part decomposition network for fine-grained and hierarchical shape segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9491–9500, 2019. 3

[47] Qian Zheng, Zhuming Hao, Hui Huang, Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Skeleton-intrinsic symmetrization of shapes. *Computer Graphics Forum*, 34(2):275–286, 2015. 3

[48] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797*, 2016. 5

[49] Chenyang Zhu, Kai Xu, Siddhartha Chaudhuri, Li Yi, Leonidas J Guibas, and Hao Zhang. Adacoseg: Adaptive shape co-segmentation with group consistency loss. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8543–8552, 2020. 3