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

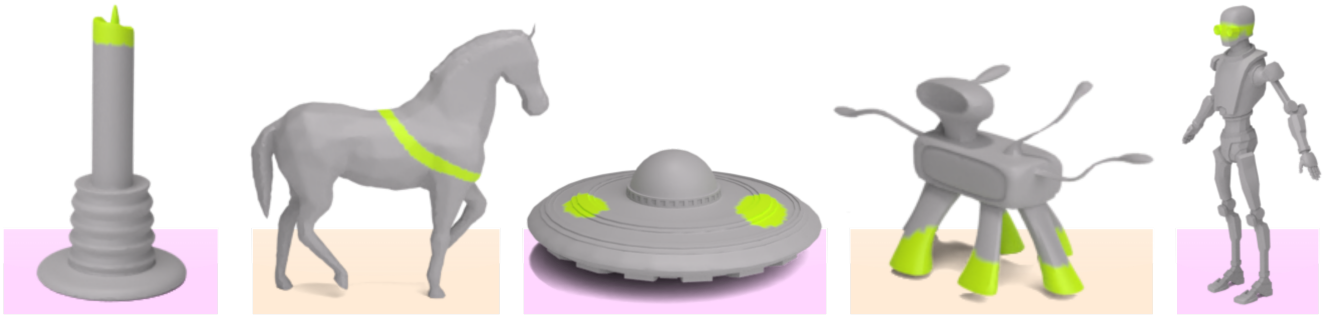

Figure 1. 3D Highlighter localizes semantic regions on a shape using text as input. Our technique reasons about *where* to place seemingly unrelated concepts in semantically meaningful locations on the 3D shape, such as a 'necklace' on a horse or 'shoes' on an alien.

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-

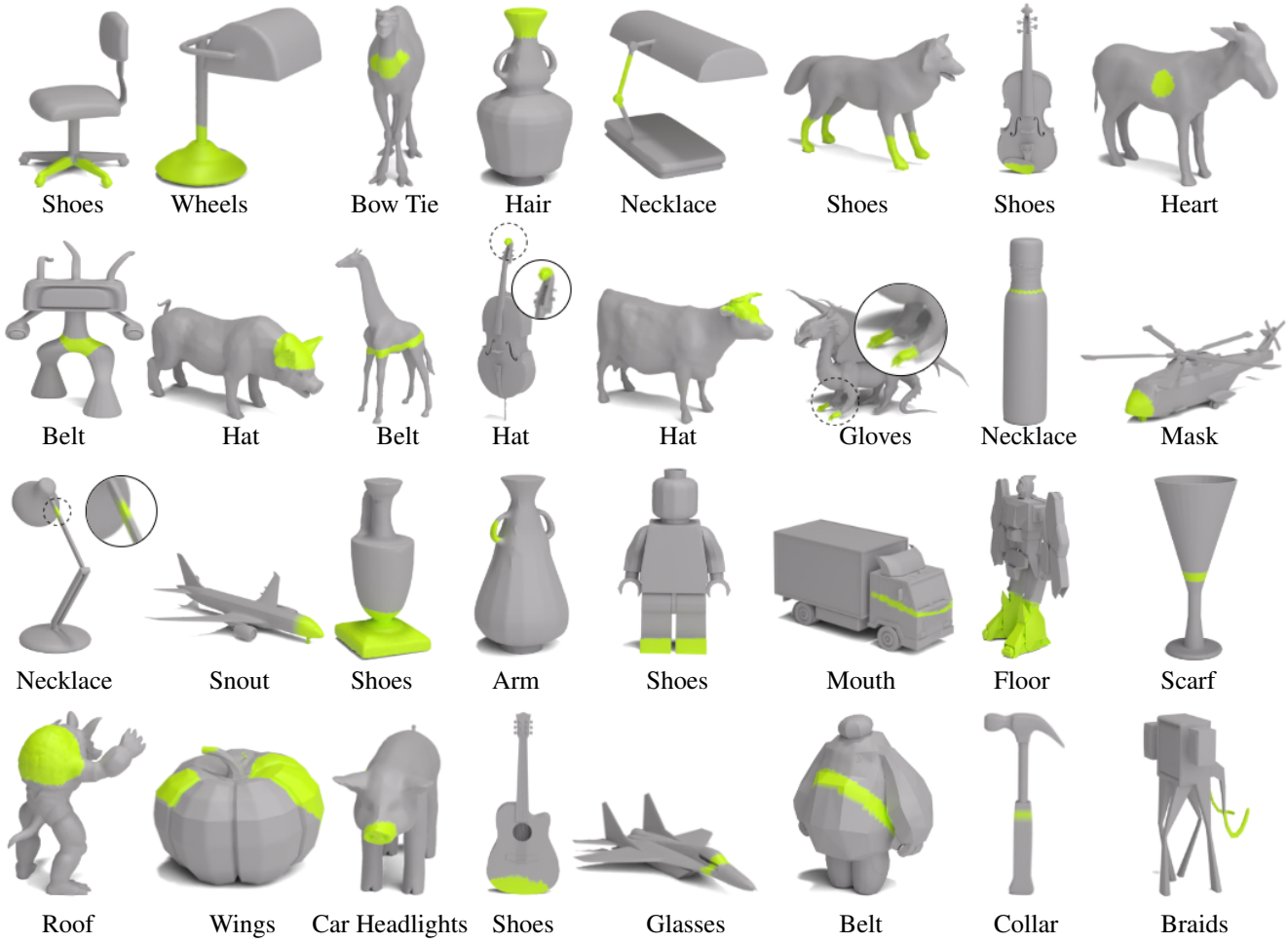

Figure 2. **Hallucinated part highlighting**. Our system is able to reason about *where* to highlight a *geometrically-absent* region on shapes. The resulting localizations demonstrate global understanding and localized part-awareness.

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-

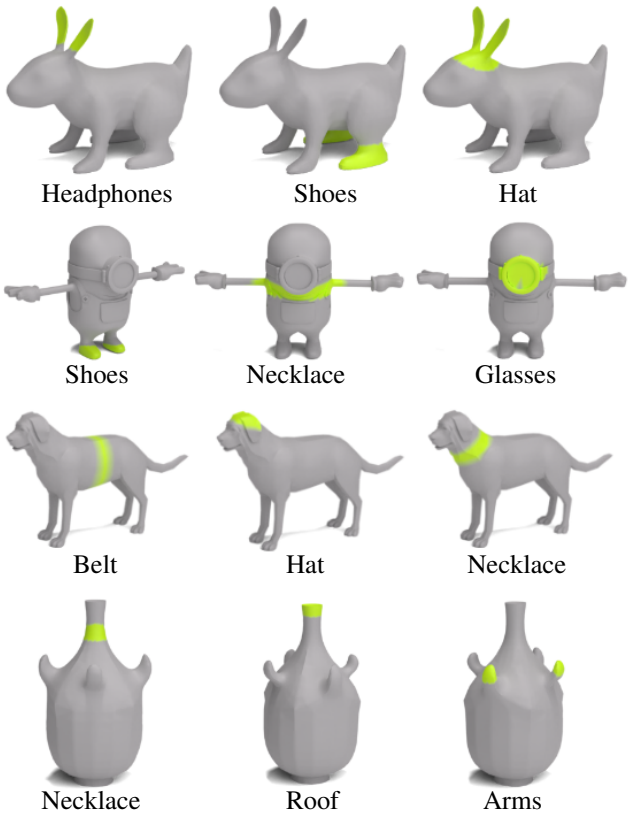

Figure 3. Our method is able to highlight different parts on the same object. For target selections that correspond to distinct regions, 3D Highlighter produces selections that are semantically meaningful and spatially separated without signal from underlying geometry.

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

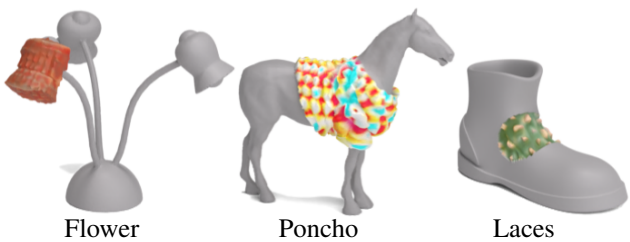

Figure 4. **Localized editing.** We incorporate textures and displacements to a region highlighted with 3D Highlighter. Used styles: Brick (left), Colorful Crochet (middle), Cactus (right).

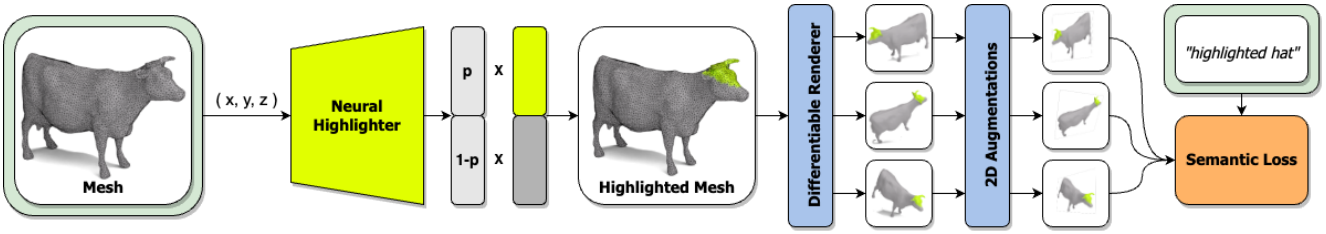

Figure 5. **Overview of 3D Highlighter.** The Neural Highlighter maps each point on the input mesh to a *probability*. The mesh is colored using a probability-weighted blend and then rendered from multiple views. The neural highlighter weights are guided by the similarity between the CLIP embeddings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

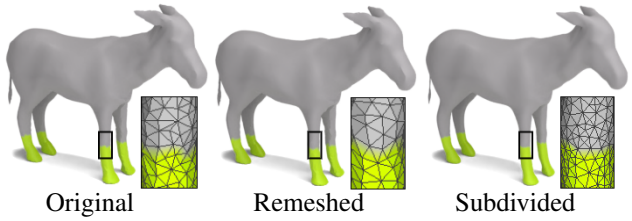

Figure 9. **Localization transfer.** We optimize our neural highlighter on one mesh (original) for the prompt 'shoes'. Once optimized, the network weights transfer the localization to different meshings of the same object (remeshed and subdivided).

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 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.

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

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

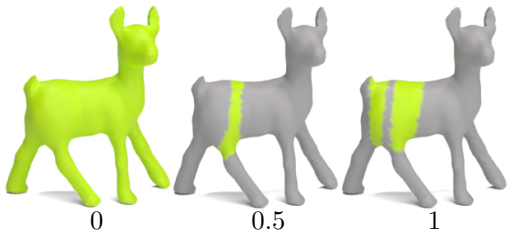

Figure 11. **Network initialization**. We optimize 3D Highlighter for the text prompt 'belt' using different initialization methods: using a default initialization where all output probabilities are near 0.5 (middle) or altering the final layer so that all outputs are 0 (left) or 1 (right). Initializing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

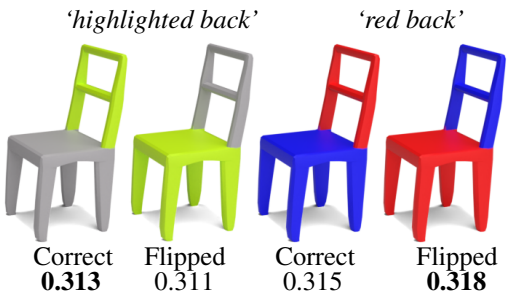

Figure 12. **CLIP understanding.** We examine CLIP similarity scores for several prompt formulations targeting the '*back*' of the chair while using the correct color assignment and where the coloring is flipped. For the prompt '*gray chair with highlighted back*' (left) we observe that the CLIP score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

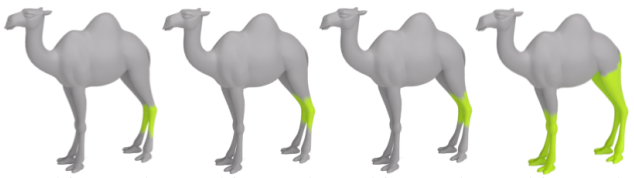

Figure 13. **Prompt generality**. Our system is robust to certain variations in object specifications. We achieve desirable results for the text input '*camel with highlighted shinguards*' (left), as well as for other variations ('*pig*' and '*animal*'). If the object specification, such as '*chair*',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

Supplementary Material for 3D Highlighter: Localizing Regions on 3D Shapes via Text Descriptions

We provide additional information about our method. Appendix A shows additional results and experiments we conducted with 3D Highlighter. Appendix B elaborates on our implementation details, including how we accomplish primary view selection, our network architecture, and our optimization scheme. Appendix C shows highlights on additional mesh and prompt combinations.

A. Additional Experiments and Details

Geometric edits. 3D Highlighter can be applied to create localized geometric edits by performing extrusion, stretching, deletion, and selection on localized regions (see Fig. 14). For extrusion, we scale vertices in the localized region along their normal vectors. With stretching, we shift the vertices in the localized region by some constant value. For deletion, we remove all vertices in the localized region as well as the faces they make up. For selection, we render only faces comprised entirely of vertices in the localized region. This application enables users to alter the geometry of semantic regions of 3D objects using only text.

Multi-class semantic segmentation. Our method can be used to obtain multi-class semantic segmentations of 3D objects (see Fig. 15). This application takes advantage of 3D Highlighter's ability to identify semantic regions. First we localize each semantic class on the object individually. Then we initialize a graph cut segmentation algorithm using our predicted probabilities for all classes. This algorithm outputs a segmentation of the vertices that is based on our predictions, but is smooth and conforms to the geometry of the mesh. This extension of 3D Highlighter allows users to acquire multi-class semantic segmentations for meshes with geometric parts not found in 3D datasets.

Viewpoint sampling. Our primary viewpoint sampling procedure is tailored to our specific localization task allowing it to produce more accurate highlights than other sampling methods. We evaluate three different viewpoint sampling procedures: ours (primary), anchor, and uniform sampling (Fig. 16). Primary view sampling is described in 3.3 in the main paper. Our primary view sampling is a variant of anchor view sampling (used in [24]), in which we sample all n views from a Gaussian centered on the center view. The anchor sampling approach uses the center (anchor) view in each iteration along with $n-1$ Gaussian samples. Uniform sampling samples n random views uniformly, independent of any center view. For the non-uniform approaches (primary and anchor) we evaluate each with two different center views, once where the target selection region is visible (blue) and once where it is not (orange).

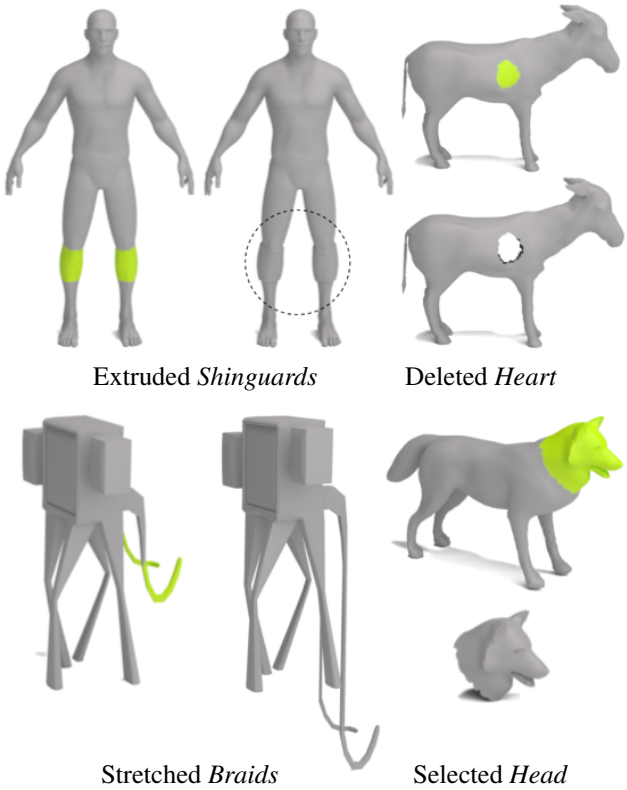

Figure 14. **Geometric edits**. Using regions selected with 3D Highlighter, we can perform localized geometric edits such as extrusion, stretching, deletion, and selection.

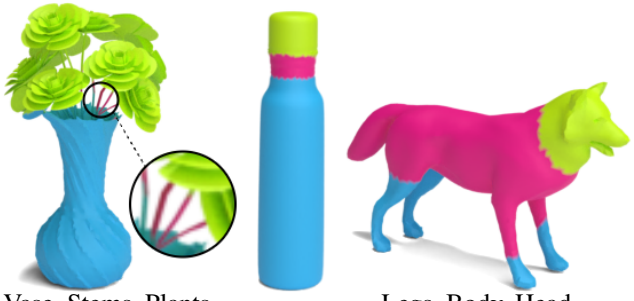

Figure 15. **Multi-class semantic segmentation**. 3D Highlighter can be used to obtain semantic segmentations of 3D objects with unique geometric parts not found in any 3D dataset or annotations.

We observe that our method produces similar results to the anchor method when using a center view where the text-specified target region is visible. However, when the target region is not visible, our method achieves more desirable results compared to anchor view sampling. This is

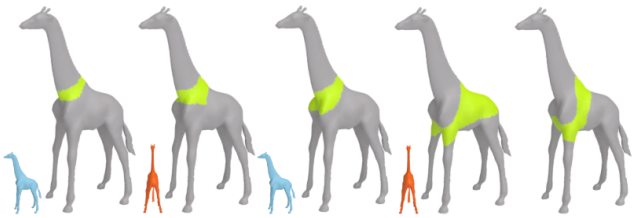

Figure 16. **Viewpoint sampling**. Results using different center view selection and sampling techniques for the target text: 'necklace'. Our primary view sampling procedure produces better results than uniform sampling for center views where the selection region is (side on) and is not (facing away) visible. With a good center view, both our primary view method and the anchor view method produce accurate selections. However, with a bad center view our method produces more desirable results than the anchor view approach.

because the anchor view sampling approach over-samples views near the anchor view and when this anchor view does not include the target region, it does not sample enough views of that region to effectively localize. Our method also produces better results than uniform sampling since uniform sampling results in many views from angles where the the shape is not recognizable to CLIP. By centering our sampling on a view where the shape is recognizable, we avoid impeding the optimization. Thus, our primary sampling approach strikes a balance in that we sample widely enough to sufficiently learn the target region while avoiding problematic views that are unrecognizable to CLIP and impede the optimization.

Impact of positional encoding. As specified in Sec. 3.1 in the paper, we choose not to use a positional encoding. Although positional encoding has been shown to allow networks to learn high frequency features [37] and is commonly used in neural fields [42], our task actually benefits from low frequency predictions. In Fig. 17, we optimize 3D Highlighter on the target region 'belt' both with and without a positional encoding. Using the positional encoding (right) gives the network too much freedom. This creates noisy highlight artifacts across the mesh. In extreme cases, the network can even hallucinate letters onto the mesh as seen in the figure. Without the positional encoding (left), the spectral bias results in a contiguous highlight region without any highlight artifacts.

Full (ours) **0.331** | *Positional Encoding* 0.312

Figure 17. **Positional encoding impact**. We optimize 3D Highlighter for the target localization '*belt*' with (*Positional Encoding*) and without (***Standard***) a positional encoding.

Quantitative evaluation details. In our quantitative evaluation, we measure the CLIP R-Precision of different localization methods on the highlighter task. Our CLIP R-Precision is defined as follows. We designate the set T_p to be a set of 10 possible target localizations. We also use 16 distinct meshes to create a dataset D of 16 different mesh/target localization pairs in which each mesh M_i has a corresponding target localization L_i where $L_i \in T_p$. To evaluate a method, we compute highlights for all pairs $(M_1, L_{i_1}), (M_2, L_{i_2}), ...(M_{16}, L_{i_{16}})$ in D. Next, we use CLIP to attempt to retrieve the original target localization that was used to generate each highlight. To do so, we choose the target localization text in T_p that has the highest CLIP similarity to the highlight. If this chosen target localization matches the target localization used to generate the highlight, then we consider that to be a successful retrieval. To obtain the CLIP R-Precision for a method, we report the percentage of mesh/target localization pairs in D that were successfully retrieved.

As specified in Sec. 4.1, we report the CLIP R-Precision for all methods using both the ViT-L/14 and ViT-B/16 CLIP models. Additionally, since our highlight is represented in 3D, we have to render our highlighted mesh into 2D. We do so using a different renderer than the one we use during optimization. By evaluating on different CLIP models and using a new renderer, we show that our highlight localizations are robust across different CLIP models and renderers.

As shown in Tab. 1, our method achieves higher CLIP R-Precision than both baselines. In addition to these quantitative results, Fig. 18 shows qualitative differences between the highlights of the different methods. From this figure, we can see that 3D Highlighter produces more accurate localizations than the baselines. Both Text2LIVE and LSeg also struggle to produce contiguous highlight regions. Additionally, LSeg frequently produces empty localization regions such as seen in the first example in the top left corner of Fig. 18. The results of the baselines demonstrate the difficulty of the highlighter task.

CLIP understanding. 3D Highlighter relies on CLIP for supervision and thus is limited by biases in CLIP. There are cases where CLIP's understanding does not align with human visual understanding of where the region should be localized. In Fig. 19, we use 3D Highlighter to localize the region '*ears*' (left) on a bunny mesh. To a human observer, it is clear that the localization does not contain the bunny's ears: instead, the localization is a region on the side of the bunny's head. This is likely a result of CLIP more strongly associating ears with being placed on the side of a head than on top. In such cases, 3D Highlighter will pro-

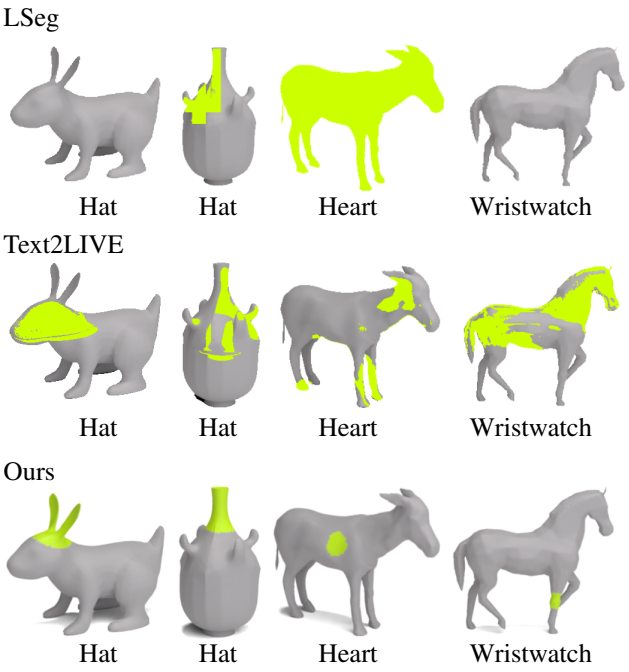

Figure 18. **Qualitative Evaluation Examples**. Highlights on different meshes and prompts for LSeg [20], Text2LIVE [2], and 3D Highlighter (ours). Both baselines struggle on many mesh/prompt combinations. LSeg often outputs no selection region (as seen in the LSeg horse).

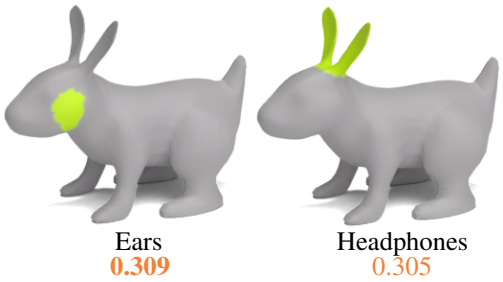

Figure 19. **CLIP understanding**. The result of optimizing CLIP towards '*headphones*' (right) results in a more '*ear-like*' result compared to optimizing towards *ears* (left). The **CLIP similarity** between the '*ears*' text prompt and both highlighted meshes also confirms that CLIP's semantic association may not always correspond with the user's semantic association.

vide poor localizations since it is based on CLIP's preferences. However, if we use 3D Highlighter to localize the region '*headphones*' (right) on the same bunny mesh, we get a localization that has good visual correspondence to both '*headphones*' and '*ears*' (since the ideal localization for these two prompts on a bunny should look very similar). If we measure the CLIP similarity of both results to the text '*gray bunny with highlighted ears*', we find that the '*ears*' localization has higher CLIP similarity even though it has less visual correspondence. This explains why the '*ears*' target region does not produce a localization like the one

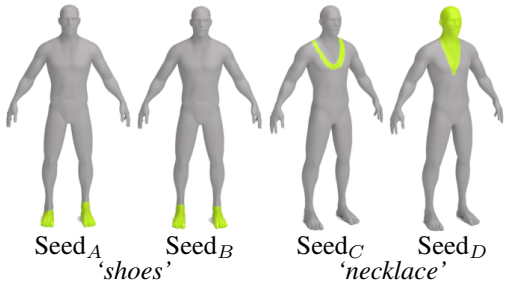

Figure 20. **Optimization sensitivity**. We observe that the results of 3D Highlighter are robust to different seeds when using certain target prompts (such as shoes, left); while other target prompts may produce more variable results (such as necklace, right).

produced for the target region '*headphones*'. Thus, when CLIP's biases lead to poor results on a given target region, we can often still obtain a good localization by running 3D Highlighter on a different specification of the target region.

Optimization consistency and sensitivity. Depending on the combination of mesh and target localization region, 3D Highlighter's optimization can vary in its sensitivity to non-determinism and thus its consistency (Fig. 20). For some combinations of meshes and prompts, the supervision signal is very strong. As such, non-determinism has little to no impact and the optimization produces nearly identical results every run. However, for some mesh and prompt combinations, the supervision signal is weaker. As a result, the optimization is more sensitive to non-determinism and we see that the highlighted regions can differ significantly between runs.

B. Implementation

Choice of primary view. We use CLIP to automatically select our primary view (see Fig. 21). To sample our views during the rendering step, we need to choose a *primary* view to center our view sampling distribution on. We want this view to correspond with CLIP's understanding of the underlying object as well as the target localization region. As such, we sample views uniformly around the object and for each rendered view, we encode it with CLIP and compare it to the encoded text target 'A 3D render of a gray [object] with highlighted [target region]'. We then choose the view with the highest CLIP similarity to the text target to be our primary view.

Neural highlighter architecture and implementation. Our neural highlighter is represented by an MLP with 6 linear layers. The input dimension is 3, encoding (x, y, z) coordinates. Each linear layer has a width of 256. After each of the first 5 linear layers we apply a ReLU activation followed by a layer norm. After our 6th and final linear layer, we instead apply a softmax activation that converts our output into a vector of probabilities. Thus, our final

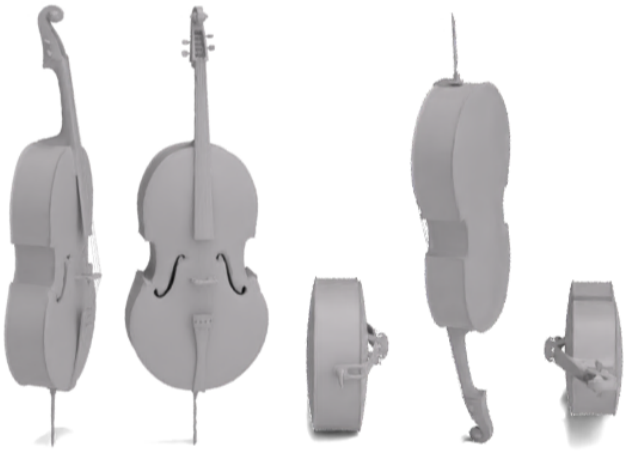

Figure 21. **Automatic primary view selection**. We select our primary view by sampling views uniformly and choosing the view with the highest CLIP similarity to the text target. We show sampled views and their CLIP similarity score to the target prompt.

layer outputs an n dimensional tensor where n is the number of classes. Each element of the output tensor corresponds to the probability of the vertex belonging to that class. For the standard highlighter task, there are only 2 classes: target region and not target region. Thus, there are 2 elements in the output vector and we can use the element of the output vector corresponding to the probability of belonging to the target region as the highlight probability described in the main paper.

Optimization. We optimize the parameters of our neural highlighter using PyTorch's Adam optimizer with a constant learning rate of $1e^{-4}$. We train for 2500 iterations on a single A40 GPU which takes around 5 minutes to complete. See Fig. 22 for a visualization of the progression of predictions during the optimization process.

C. More Visual Results

We show highlights for additional combinations of meshes and prompts: Fig. 23 depicts highlights on animal meshes while Fig. 24 shows highlights on object meshes.

Figure 22. **Optimization visualization**. We optimize 3D Highlighter on a mesh of a dog with the target localization '*hat*' and visualize the predicted highlighted region at five steps throughout the optimization. We also report the CLIP similarity score at each step.

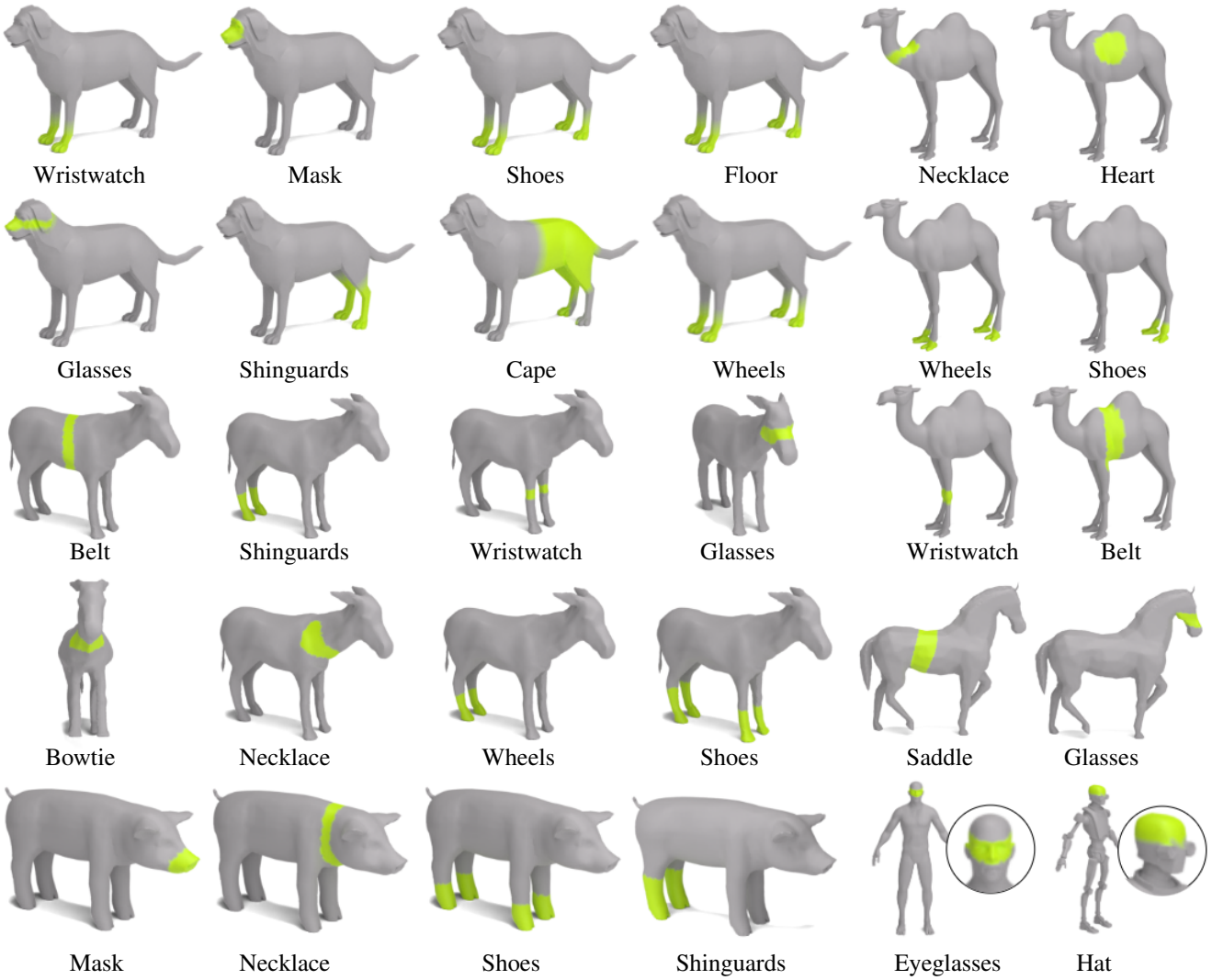

Figure 23. **Animal gallery**. Example highlights on meshes of dogs, goats, camels, horses, pigs, humans, and robots.

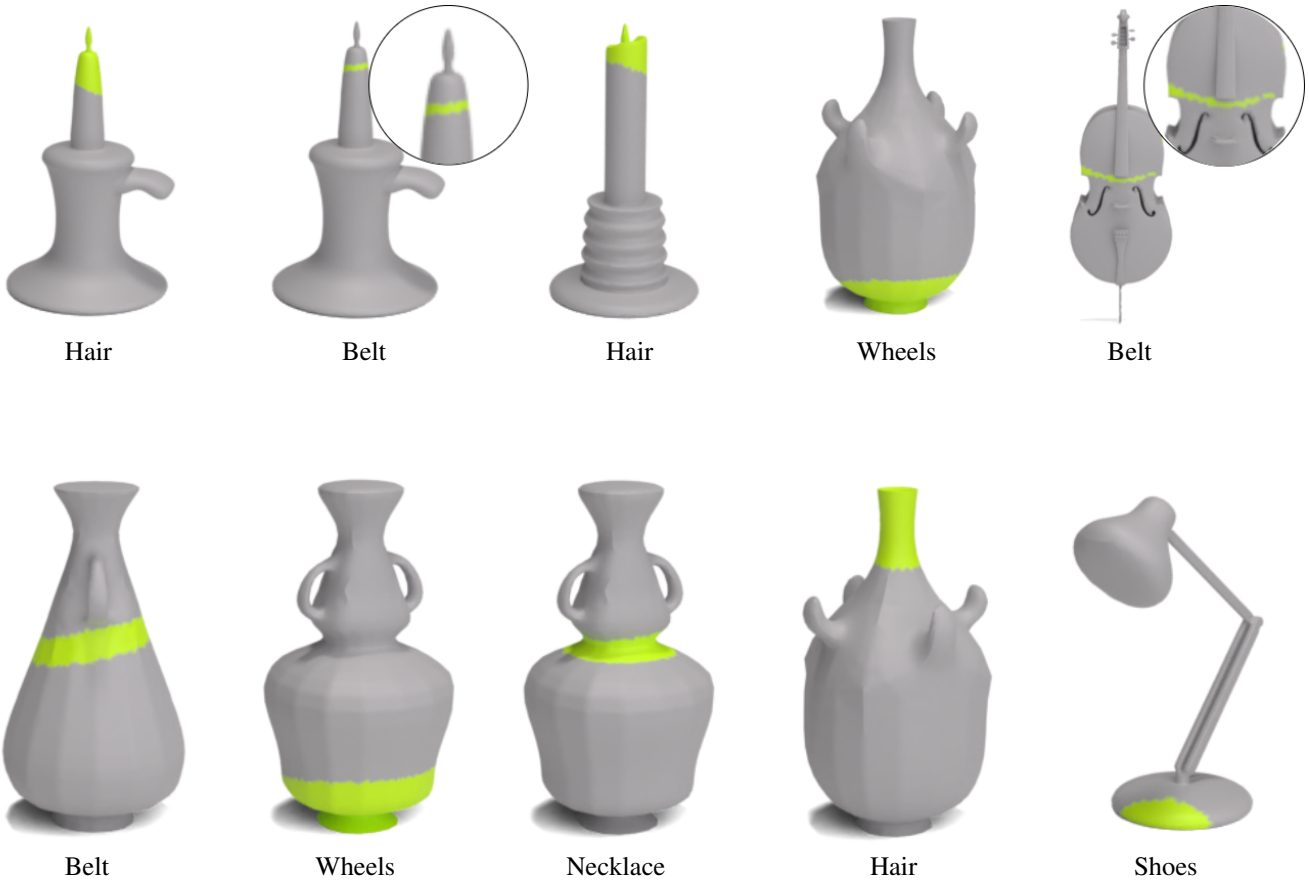

Figure 24. **Object gallery**. Example highlights on meshes of candles, vases, instruments, and lamps.