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 [25]), 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).

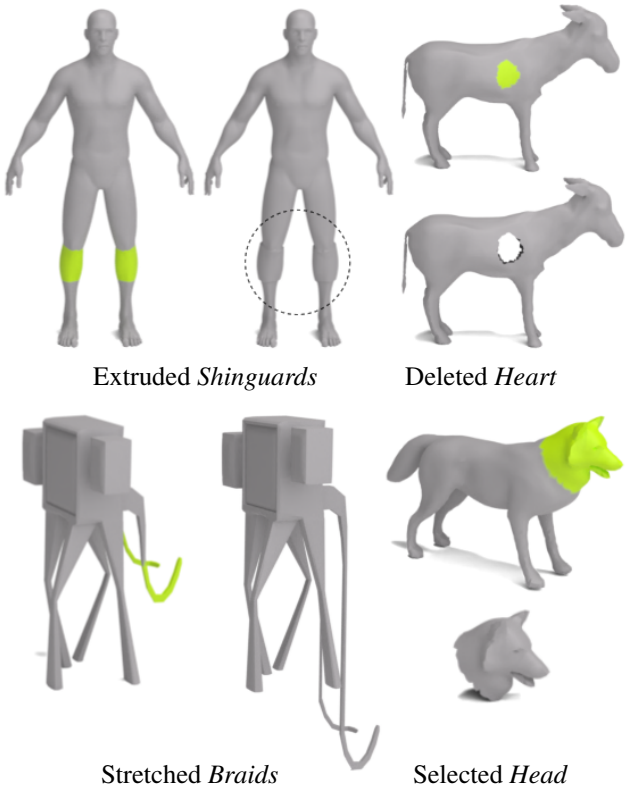


Figure 14. **Geometric edits**. Using regions selected with 3D Highlighter, we can perform localized geometric edits such as extrusion, stretching, deletion, and selection.

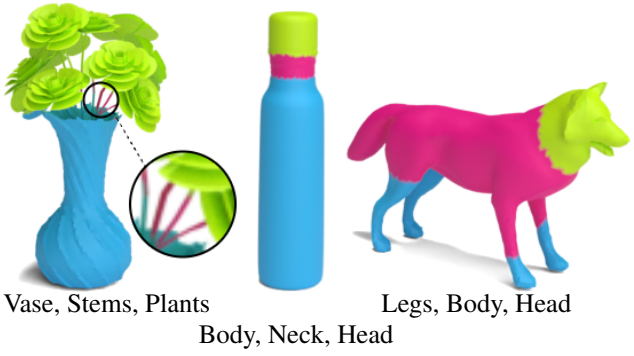


Figure 15. **Multi-class semantic segmentation**. 3D Highlighter can be used to obtain semantic segmentations of 3D objects with unique geometric parts not found in any 3D dataset or annotations.

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

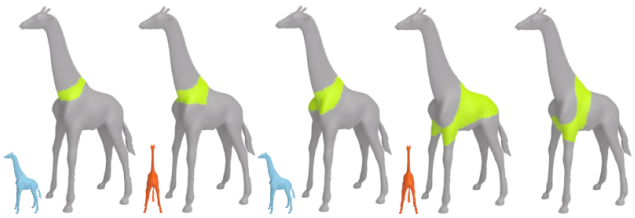


Figure 16. **Viewpoint sampling**. Results using different center view selection and sampling techniques for the target text: 'necklace'. Our primary view sampling procedure produces better results than uniform sampling for center views where the selection region **is** (side on) and **is not** (facing 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 [38] and is commonly used in neural fields [44], 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

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.

CLIP Model	ViT-L/14 ↑	ViT-B/16 ↑	ViT-B/32 ↑
LSeg [21]	18.75	6.25	6.25
Text2LIVE [2]	43.75	31.25	12.5
Ours	**81.25**	**43.75**	**25**

Table 2. **Highlight fidelity.** We extend two image-based approaches LSeg [21] (segmentation) and Text2LIVE [2] (localized editing) to the highlighting task and report CLIP R-Precision.

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.

Quantitative evaluation details. In addition to our perceptual user study, we created a quantitative evaluation strategy inspired by Dream Fields [16, 28]. In our work, we build a specialized CLIP R-Precision metric based on text-retrieval with text prompts created for different localizations of semantic regions. We evaluate both baselines and our 3D Highlighter result rendered from the same view using this specialized CLIP R-Precision. We run this evaluation on the ViT-L/14, ViT-B/16, and ViT-B/32 CLIP models and observe that our method outperforms both baselines (Tab. 2).

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 above, we report the CLIP R-Precision for all methods using the ViT-L/14, ViT-B/16, and ViT-B/32 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. 2, our method achieves higher CLIP R-Precision than both baselines. In addition to these quantitative results, Fig. 18 shows qualitative differences between

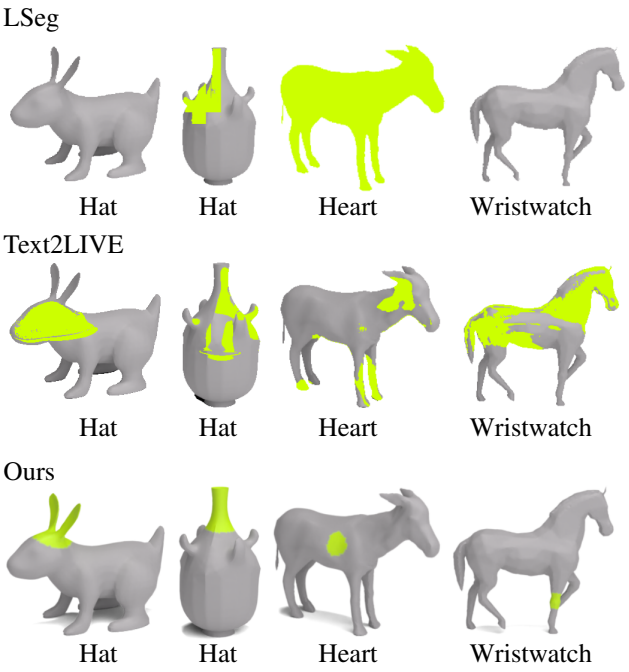


Figure 18. **Qualitative Evaluation Examples**. Highlights on different meshes and prompts for LSeg [21], Text2LIVE [2], and 3D Highlighter (ours). Both baselines struggle on many mesh/prompt combinations. LSeg often outputs no selection region (as seen in the LSeg horse).

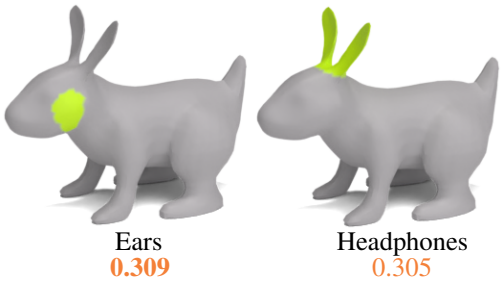


Figure 19. **CLIP understanding**. The result of optimizing CLIP towards '*headphones*' (right) results in a more '*ear-like*' result compared to optimizing towards *ears* (left). The **CLIP similarity** between the '*ears*' text prompt and both highlighted meshes also confirms that CLIP's semantic association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

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

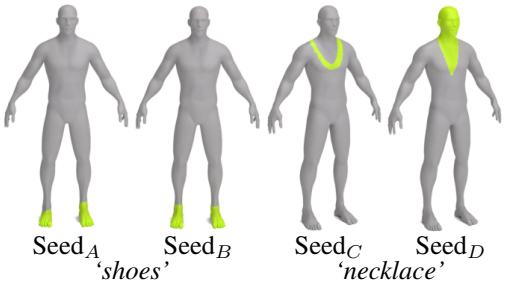
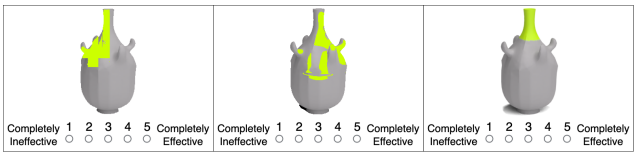


Figure 20. **Optimization sensitivity**. We observe that the results of 3D Highlighter are robust to different seeds when using certain target prompts (such as shoes, left); while other target prompts may produce more variable results (such as necklace, right).

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 provide 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 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.

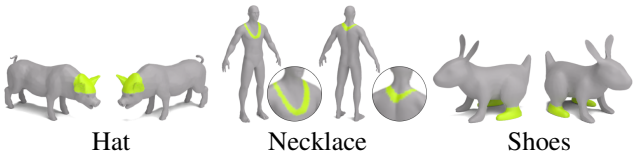
Optimization consistency and sensitivity. Depending on the combination of mesh and target localization region, 3D Highlighter's optimization can vary in its sensitivity to non-determinism and thus its consistency (Fig. 20). For some combinations of meshes and prompts, the supervision signal is very strong. 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.

View consistency. The viewing angles used during optimization are sampled from a Gaussian distribution centered at the primary view with a standard deviation of $\sigma_a = \frac{\pi}{2}$

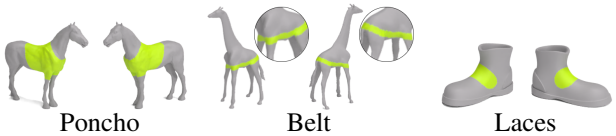


LSeg Text2LIVE Ours

Figure 21. **Perceptual study examples.** We show visual examples from our perceptual study for the question "a vase with a region corresponding to a hat highlighted."



Hat Necklace Shoes



Poncho Belt Laces

Figure 22. **View consistency.** Our highlights are view consistent.

Setting	Direct	No blend	No augs	Pos enc	Ours
ViT-L/14	37.5	37.5	25	18.75	**81.25**
ViT-B/16	12.5	18.75	6.25	12.5	**43.75**
ViT-B/32	0	0	0	6.25	**25**

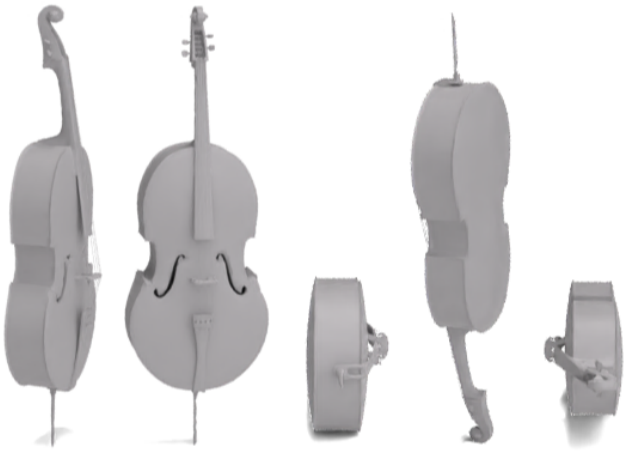
Table 3. **Quantitative ablation study.** We report a quantitative ablation of the components of our method using CLIP R-Precision.

(azimuth) and $\sigma_e = \frac{\pi}{4}$ (elevation). Due to our view sampling, 3D Highlighter sees all sides of the shape with reasonable frequency and thus produces view-consistent localizations (see Fig. 22).

Quantitative ablation. In addition to our qualitative ablation in Fig. 7, we report a quantitative ablation using CLIP R-Precision on a collection of mesh and prompt combinations and observe that our full system produces the highest scores (Tab. 3).

B. Implementation

Choice of primary view. We use CLIP to automatically select our primary view (see Fig. 23). 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.



0.34 0.33 0.24 0.23 0.18

Figure 23. **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.

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 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. 24 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. 25 depicts highlights on animal meshes while Fig. 26 shows highlights on object meshes.

Figure 24. **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.

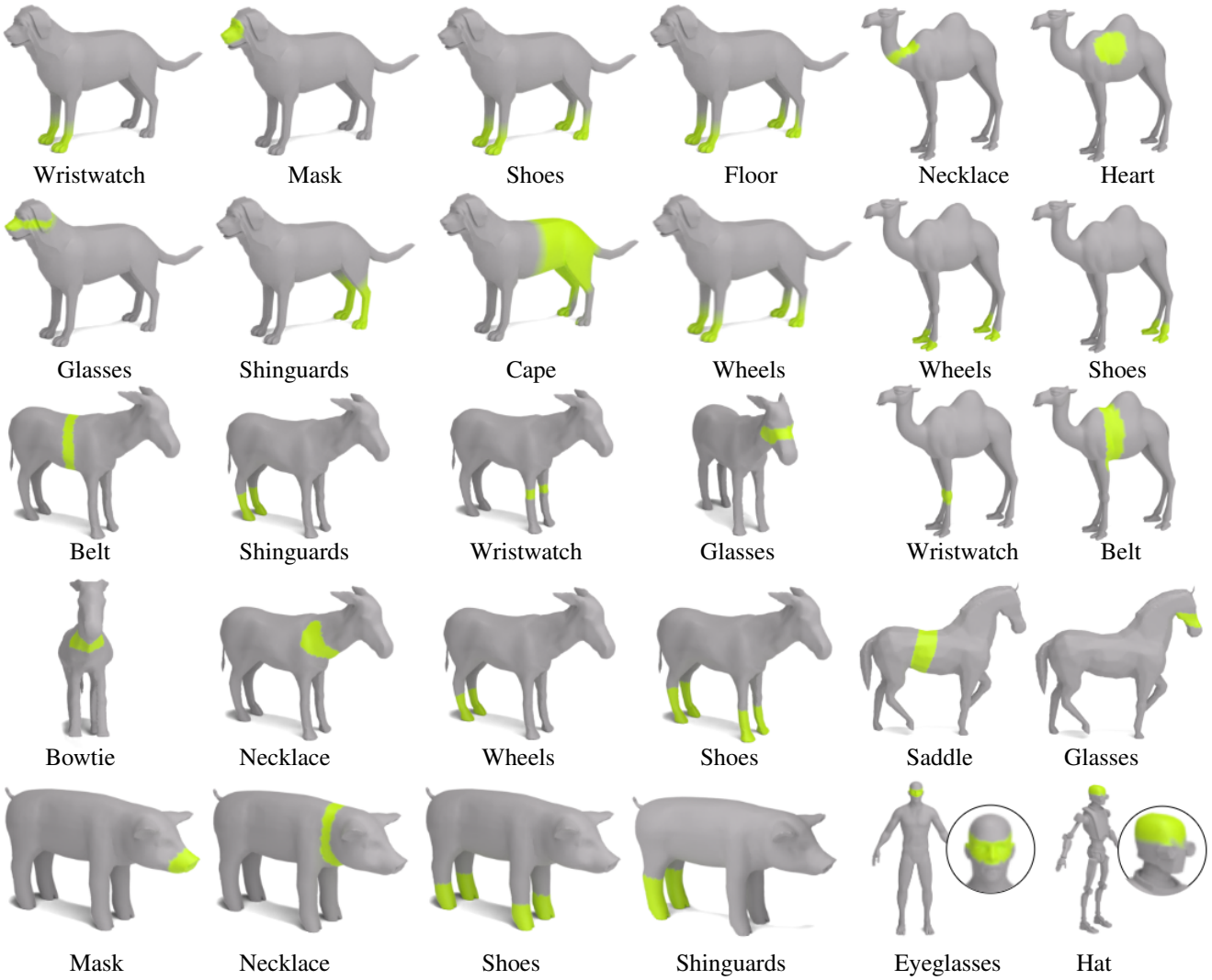


Figure 25. **Animal gallery**. Example highlights on meshes of dogs, goats, camels, horses, pigs, humans, and robots.

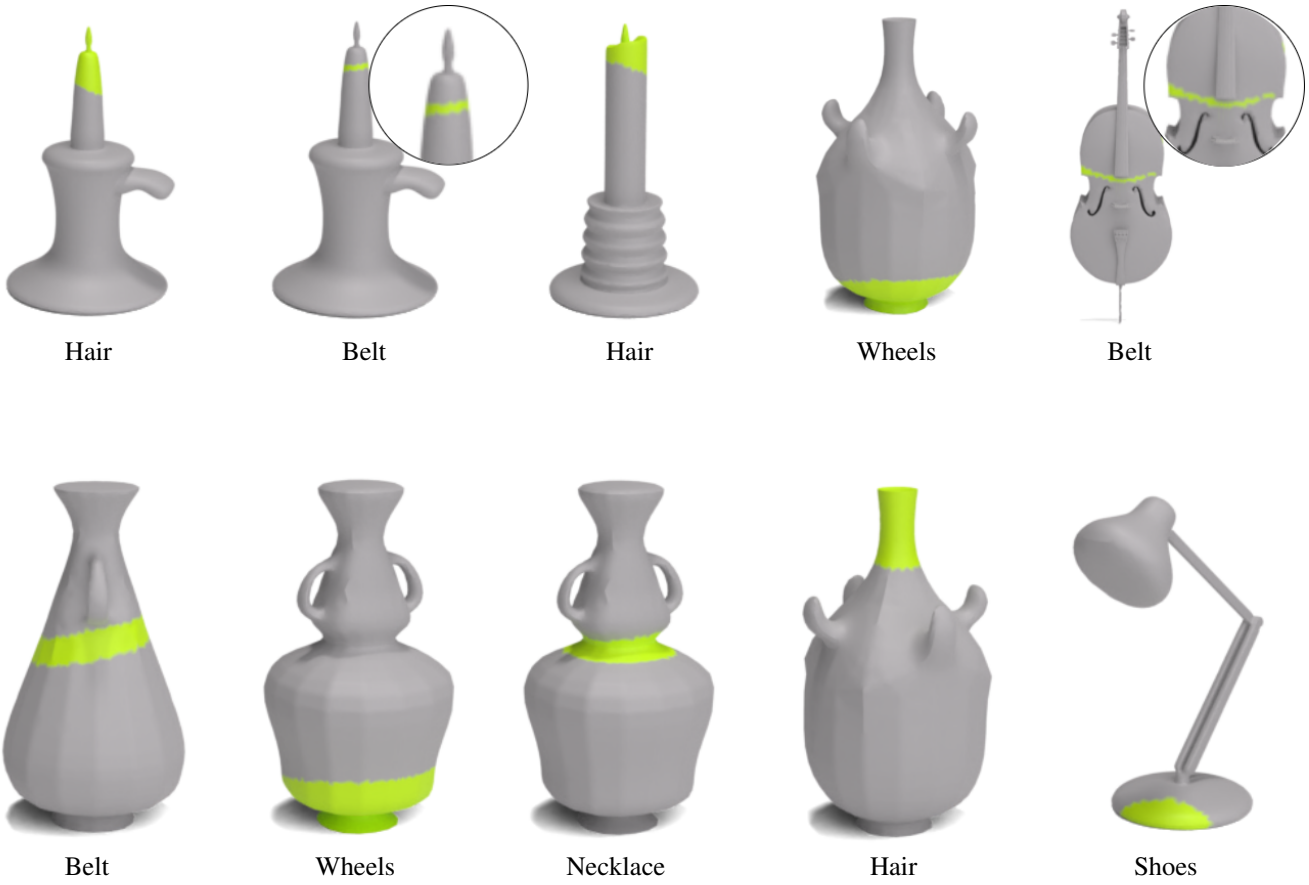


Figure 26. **Object gallery**. Example highlights on meshes of candles, vases, instruments, and lamps.