Vox-E: Text-guided Voxel Editing of 3D Objects
— Supplementary Material —

Etai Sella[1] Gal Fiebelman[1] Peter Hedman[2] Hadar Averbuch-Elor[1]
[1]Tel Aviv University [2]Google Research

Contents

We refer readers to the interactive visualizations at `index.html` that show fly-through results and comparisons. In this document, we provide additional details (Section 1), ablations (Section 2) and visualizations (Section 3).

1. Additional Details

1.1. Implementation Details

Below we provide all the implementation details of our method, detailed in Section 3 in the main paper.

Grid-Based Volumetric Representation

We use 100 images uniformly sampled from upper hemisphere poses along with corresponding camera intrinsic and extrinsic parameters to train our initial grid. We follow the standard ReLU Fields [2] training process using their default settings aside from two modifications:

1. We change the result grid size from the standard 128^3 to 160^3 to increase the output render quality.
2. As detailed in the main paper, we limit the order of spherical harmonics to be zero order only to avoid undesirable view-dependent effects (we further illustrate these effects in Section 2.2).

Text-guided Object Editing

We perform 8000 training iterations during the object editing optimization stage. During each iteration, a random pose is uniformly sampled from an upper hemisphere and an image is rendered from our edited grid G_e according to the sampled pose and the rendering process described in ReLU Fields [2]. Noise is then added to the rendered image according to the time-step sampled from the fitting distribution.

We use an annealed SDS loss which gradually decreases the maximal time-step we draw t from. Formally, this annealed SDS loss introduces three additional hyper-parameters to our system: a starting iteration i_{start}, an annealing frequency f_a and an annealing factor γ_a. With these hyper-parameters set, we change our time-step distribution to be:

$$t \sim U[t_0 + \varepsilon, t_{final} * k_i + \varepsilon], \tag{1}$$

$$k_i = \begin{cases} 1, & \text{if } i < i_{start} \\ k_{i-1} * \gamma_a, & \text{else if } i \,\%\, f_a = 0 \\ k_{i-1}, & \text{otherwise} \end{cases} \tag{2}$$

In all our experiments, the values we use for ε, i_{start}, f_a and γ_a are 0.02, 4000, 600, and 0.75. Additionally, we stop annealing the time-step when it reaches a value of 0.35. The latent diffusion model we use in our experiments is "StableDiffusion 2.1" by Stability AI.

We use a weight of 200 to balance the two terms (multiplying $\mathcal{L}_{\text{reg3D}}$ by this weight value). The volumetric regularization term operates only on the density features of the editing grid. The optimizer we used in this (and all other stages) is the Adam optimizer [3] with a learning rate of 0.03 and betas 0.9, 0.999. The resolution of the images rendered from our grid is 266×266. We add a "a render of" prefix to all of our editing prompts as we found that this produced more coherent results (and the images the LDM receives are indeed renders).

Spatial Refinement via 3D Cross-Attention

The diffusion model we use for this stage is "StableDiffusion 1.4" by CompVis and it consists of several cross-attention layers at resolutions 32, 16, and 8. To extract a single attention map for each token we interpolate each cross attention map from each layer and attention head to our image resolution (266x266) and take an average per each token. The time-step we use to generate the attention maps is 0.2 (the actual step being 0.2 * N_{steps} = 200).

The cross-attention grids A_e and A_{obj} contain a density feature and an additional one-dimensional feature a, which represents the cross-attention value at a given voxel and can be interpreted and rendered as a grayscale luma value. We initialize the density features in these grids to the density features of the editing grid's (the former stage's output) and freeze them. At each refinement iteration we generate two 2D cross-attention maps from the LDM, one for the object and one for the edit. After obtaining the 2D cross-attention maps, we render gray-scale heatmaps from A_e and A_{obj} and use $L1$ loss to encourage similarity between the rendered attention images and their corresponding attention maps extracted from the diffusion model. We repeat this process for 1500 iterations, sampling a random upper-hemisphere pose each time. As in the former optimization stage, we use the Adam optimizer with a learning rate of 0.03 and betas 0.9 and 0.999 and generate images in 266×266 resolution.

After obtaining the two grids A_e and A_{obj}, we perform element-wise softmax on their a values to obtain probabilities for each voxel belonging to either the object, denoted by $P_{obj}(v)$, or the edit, denoted by $P_e(v)$. We then proceed to calculate the binary refinement volumetric mask. To do this we define a graph in which each non-zero density voxel in our edited grid G_e is a node. We define "edit" and "object" labels as the *source* and *drain* nodes, such that a node connected to the source node is marked as an "edit" node and a node connected to the drain node is marked as an "object" node. We rank the nodes according to their $P_e(v)$ values and connect the top $N_{init-edit}$ nodes to the source node. We then rank the nodes according to their $P_{obj}(v)$ value and connect the top $N_{init-object}$ nodes to the drain node. We then connect the non-terminal nodes to each-other in a 6-neighborhood with the capacity of each edge being w_{pq} as detailed in the main paper.

We set the hyper-parameters $N_{init-edit}$ and $N_{init-object}$ to be 300 and 200. To perform graph-cut [1], we used the PyMaxflow implementation of the max-flow / min-cut algorithm.

1.2. Evaluation Protocol

To evaluate our results quantitatively, we constructed a test set composed of eight scenes: 'White Dog', 'Grey Dog', 'White Cat', 'Ginger Cat', 'Kangaroo', 'Alien', 'Duck' and 'Horse', and six editing prompts: (1) A $\langle object \rangle$ wearing big sunglasses, (2) A $\langle object \rangle$ wearing a Christmas sweater, (3) A $\langle object \rangle$ wearing a birthday party hat, (4) A yarn doll of a $\langle object \rangle$, (5) A wood carving of a $\langle object \rangle$, (6) A claymation $\langle object \rangle$. This yields 18 edited scenes in total. We render each edited scene from 100 different poses distributed evenly along a 360° ring. In addition to these 18 scenes we also render 100 images from the same poses on the initial (reconstruction) grid G_i for each input scene. When comparing our result with other 3D textual editing papers we evaluate our results using two CLIP-based metrics. The CLIP model we used for both of these metrics is ViT-B/32 and the input image text prompts used to calculate the directional CLIP metric is "A render of a $\langle object \rangle$". $CLIP_{Dir}$ is calculated for each edited image in relation to the corresponding image in the reconstruction scene. To quantitatively evaluate ablations we use two additional metrics using FID [4]. For this we use the pytorch implementation given by the authors with the standard settings.

360° ***Real Scenes***

For the 360° *Real Scenes* edits we follow the same implementation details as outlined previously, with three modifications:

1. We alternate between using the DVGO model or the ReLU-Fields model as our 3D representation. Results for both models are presented in Figure 6 of the main paper.

2. Our input poses are created in a spherical manner and when rendering we sample linearly in inverse depth rather than in depth as seen in the official implementation of NeRF .

3. We perform 5000 training iterations during the object editing optimization stage and the values we use for ε, i_{start}, f_a and γ_a are 0.02, 3000, 400, and 0.75.

1.3. 3D Object Editing Techniques

Below we provide additional details on the alternative 3D object editing techniques we compare against. All of the techniques we compare against use only an un-textured mesh and an editing prompt as input. As such, we used the meshes our inputs were rendered from as input for the editing methods. Additionally, we tested an additional scenario in which we imported the 'horse' mesh from the Text2Mesh GitHub repository to blender, added a grey-matte material to it and rendered images of it to use as input for our system. This scenario used four prompts: (1) A wood carving of a horse, (2) A horse wearing a Santa hat, (3) A donkey, (4) A carousel horse, and was used for qualitative comparisons only.

Text2Mesh

When comparing to Text2Mesh we used the code provided by the authors and the input settings given in the "run_horse.sh" demo file.

SketchShape

In this comparison we again use the code provided by the authors. And the input parameters used are the default parameters in the 'train_latent_nerf.py' script 'train_latent_nerf.py' script with 10,000 training steps (as opposed to the default 5,000).

Latent-Paint

We compared our method to Latent-Paint only qualitatively as this method outputs edits that transform only the appearance of the input mesh, rather than appearance and geometry. As in SketchShape we used the code provided by the authors and used the default input settings provided for latent paint, which are given in the 'train_latent_paint.py' script.

DFF + CN

In this comparison we use the code provided by the authors and the default input parameters provided for this method.

1.4. 2D Image Editing Techniques

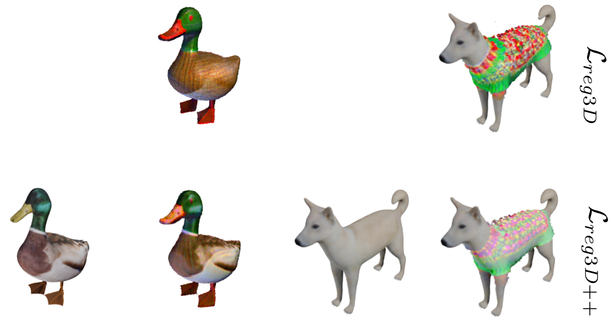
When comparing to InstructPix2Pix and SDEdit we constructed two image sets for each scene / prompt combination we wanted to test. Both sets were created by rendering one of our inputs in evenly spaced poses along a 360° ring, one set was rendered over a white background and the other over a 'realistic' image of a brick wall. We used these sets as input for each 2D editing method along with an editing prompt and compared the results to rendered outputs from our result grids. When comparing to InstructPix2Pix we used the standard InstructPix2Pix pipeline with 16bit floating point precision and 20 inference steps. We used the default guidance scale (1.0) for the images rendered over the 'realistic' background and increased the guidance scale to 3.0 for the images rendered over a white background, as we found it to produce higher quality results specifically for these more 'synthetic' images. When giving prompts to InstructPix2Pix we rephrased our prompts as instructions, for example turning "a dog wearing sunglasses" to "put sunglasses on this dog". When comparing to SDEdit we used the standard SDEdit pipeline with guidance scale of 0.75 and a strength of 0.6.

2. Ablations

In this section, we show a more detailed ablation study which evaluates the effect of our volumetric regularization loss (Section 2.1) and an additional experiment, demonstrating the effect of using high order spherical harmonics coefficients (Section 2.2).

2.1. Alternative Regularization Objectives

Table 1 shows a quantitative comparison over different image-space and volumetric regularizations. Only the image-space L1 loss also appears in the main paper. Below we provide additional details on these ablations.



Input

"A wood carving of a duck"

Input

"A dog wearing a christmas sweater"

Figure 1. **Regularizing RGB colors in addition to volumetric densities**. We show results obtained when using our default regularization objective $\mathcal{L}_{reg3D}$ (top-row) compared against results obtained when using $\mathcal{L}_{reg3D++}$- an alternative version of $\mathcal{L}_{reg3D}$ (bottom-row) in which we penalize the miscorrelation between both density and color features. These results show that regularizing both density and RGB can be limiting, especially when the edit requires a drastic change in color, such as changing the white fur of the dog into a vibrant christmas sweater.

Image-space Regularization In this setting we render images from our editing grid G_e in the poses corresponding to the input images during each iteration of the optimization stage. Rather than using a volumetric regularization, we incur a loss between the images rendered from G_e and the corresponding input image while using the same weight used to balance $\mathcal{L}_{\text{reg3D}}$ with the annealed SDS loss (this weight is set to 200, as detailed in Section 1.1). We evaluate this ablation using L_1 and L_2 image space loss functions.

Alternative Volumetric Regularization Functions In this setting we replace our correlation-based regularization with other functions that encourage similarity between the density features of the grids using the same balancing weight. Namely we compare against $L1$ and $L2$ volumetric loss functions, both penalizing the distance between the density features of G_i and those of G_e. We additionally compare against an alternative version of $\mathcal{L}_{\text{reg3D}}$ in which we penalize the miscorrelation between both density and color features, formally:

	Loss Function	CLIP$_{Sim}$ ↑	CLIP$_{Dir}$ ↑	FID$_{Rec}$ ↓	FID$_{Input}$ ↓
2D	L_1	0.26	0.02	415.96	437.09
	L_2	0.25	0.02	437.68	467.14
3D	L_1	0.36	0.05	222.91	284.86
	L_2	0.35	0.05	240.50	284.83
	$\mathcal{L}_{reg3D++}$	0.34	0.02	**210.46**	**242.73**
	$\mathcal{L}_{\text{reg3D}}$	**0.36**	**0.06**	223.89	272.73

Table 1. **Detailed ablation study**, evaluating the effect of different regularization objectives. We compare the performance using $\mathcal{L}_{\text{reg3D}}$, with image-space (top rows) and volumetric (bottom rows) L_1 and L_2 losses, as well as $\mathcal{L}_{\text{reg3D++}}$, which also penalizes miscorrelations between color features.

$$\mathcal{L}_{reg3D++} = \mathcal{L}_{reg3D} + \left(1 - \frac{Cov(f_i^{rgb}, f_e^{rgb})}{\sqrt{Var(f_i^{rgb})Var(f_e^{rgb})}}\right) \quad (3)$$

We find that using this loss yields better reconstruction scores, at the expense of significantly lower CLIP-based scores (e.g., CLIP$_{Dir}$ scores drop from 0.08 to 0.02). Qualitatively, constraining RGB values as well as density features appears too limiting for our purposes. This can be seen in Figure 1, where we compare results obtained when using $\mathcal{L}_{reg3D++}$ against results obtained when using $\mathcal{L}_{reg3D}$. When observing these results, we can see that the edit integrity is reduced at the expense of the preservation of the origin object's color. This is evident in the duck, for instance, where the brown wooden color of the body is only clearly visible in the $\mathcal{L}_{reg3D}$ example. Furthermore, the colors of the sweater on the dog are significantly faded when regularized with $\mathcal{L}_{reg3D++}$ as the colors of a standard christmas sweater are typically much more vibrant than the white fur of the dog.

2.2. Ablating the Color Representation

As mentioned in Section 3.1 of the main paper, we do not model view dependent effects using higher order spherical harmonics as that leads to undesirable effects. We demonstrate this by observing these effects in examples rendered with 1st and 2nd order spherical harmonic coefficients as color features. These results can be seen in videos available on our project page.

When observing these results we can clearly see how view-dependent colors yield undesirable effects such as the feet of the “yarn kangaroo" varying from green to yellow across views or the head of the dog becoming a birthday party hat when it faces away from the camera. We additionally see the colors become over-saturated, especially when using second-order spherical harmonic coefficients. It is also evident that the added expressive capabilities of the model allow it to over-fit more easily to specific views, creating unrealistic results such as the “cat wearing glasses" in the first and second order coefficient models, where glasses are scattered along various parts of its body. We note that while this expressive power currently produces undesirable effects it does potentially enable higher quality and more realistic renders, and therefore, we believe that constraining this power is an interesting topic for future research.

2.3. Cross-attention Grid Supervision

As explained in Section 1.1, we use a constant timestamp of 0.2 when extracting attention maps for training our attention grids A_e and A_{obj}. This value was chosen empirically as we found that higher time-steps tend to be noisier and less focused, while lower time-steps varied largely from pose to pose producing inferior attention grids. This can be seen qualitatively in Figure 2. As illustrated in the figure, the attention values for the edit region get gradually more smeared and unfocused as the time-steps increase. This is evident, for instance, in warmer regions around the kangaroo's tail or the head of the duck. While perhaps less visually distinct, we can also observe that in lower timestamps the warm regions denoting high attention values cover a smaller area of the region which should be edited. We empirically find that this makes it more challenging for separating the object and edit regions.

3. Additional Visualizations and Results

Visualizing 2D cross-attention maps and images rendered from our 3D cross-attention grids While the attention maps used as ground-truth are inherently unfocused (as they are up-sampled from very low resolutions) and are not guaranteed to be view consistent, we show that learning the projection of these attention maps on to our object's density produces view-consistent heat maps for object and edit regions (Figure 3).

References

[1] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[2] Animesh Karnewar, Tobias Ritschel, Oliver Wang, and Niloy Mitra. Relu fields: The little non-linearity that could. In *ACM SIGGRAPH 2022 Conference Proceedings*, pages 1–9, 2022.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[4] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. `https://github.com/mseitzer/pytorch-fid`, 08 2020. Version 0.2.1.

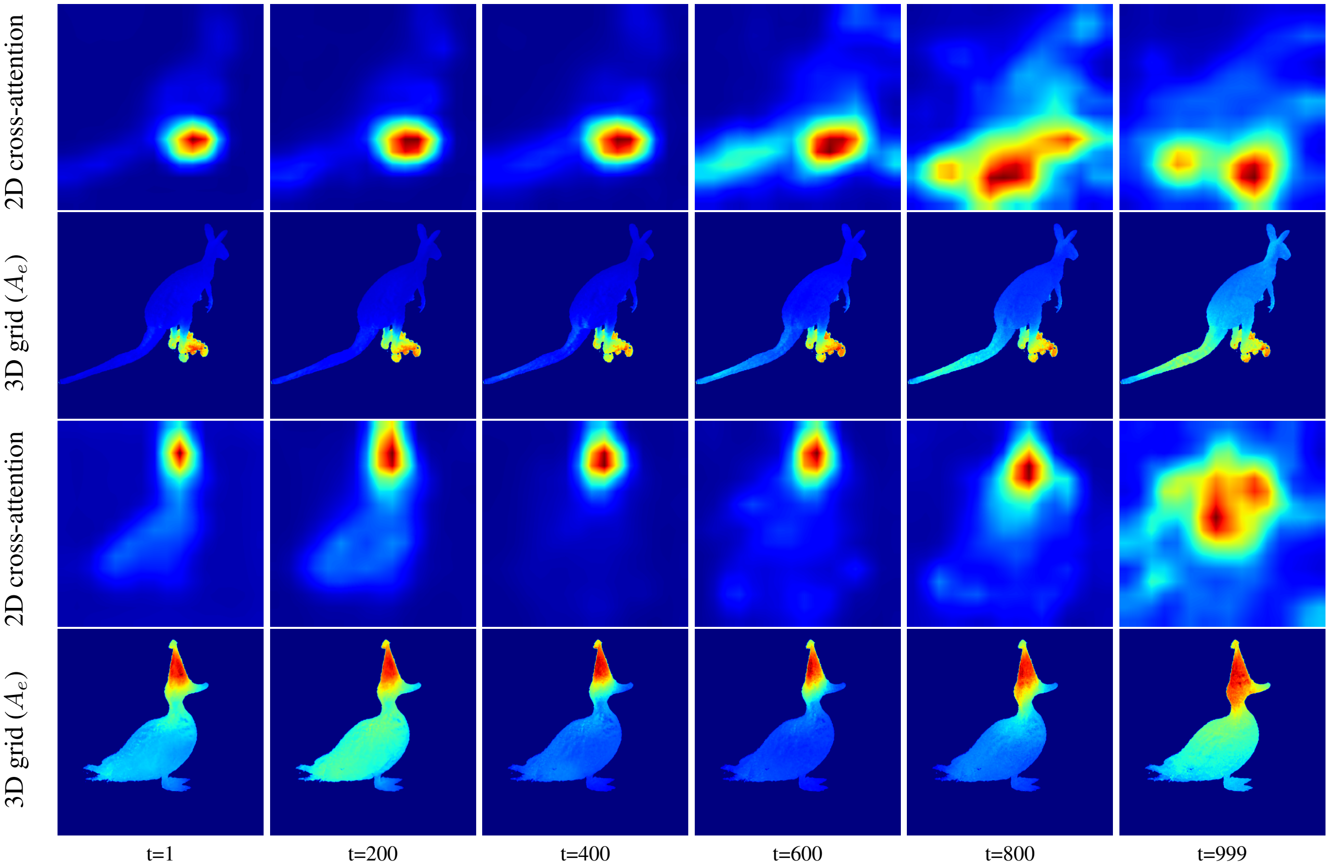


Figure 2. **Visualizing 2D cross-attention maps and 3d cross-attention grids over different diffusion timestamps**. We visualize the trained 3d cross-attention grids and the corresponding 2D cross-attention maps used as supervision across different diffusion timestamps. We show them for the edit region corresponding to the token associated with the word “rollerskates" (top two rows) and “hat" (bottom two rows).

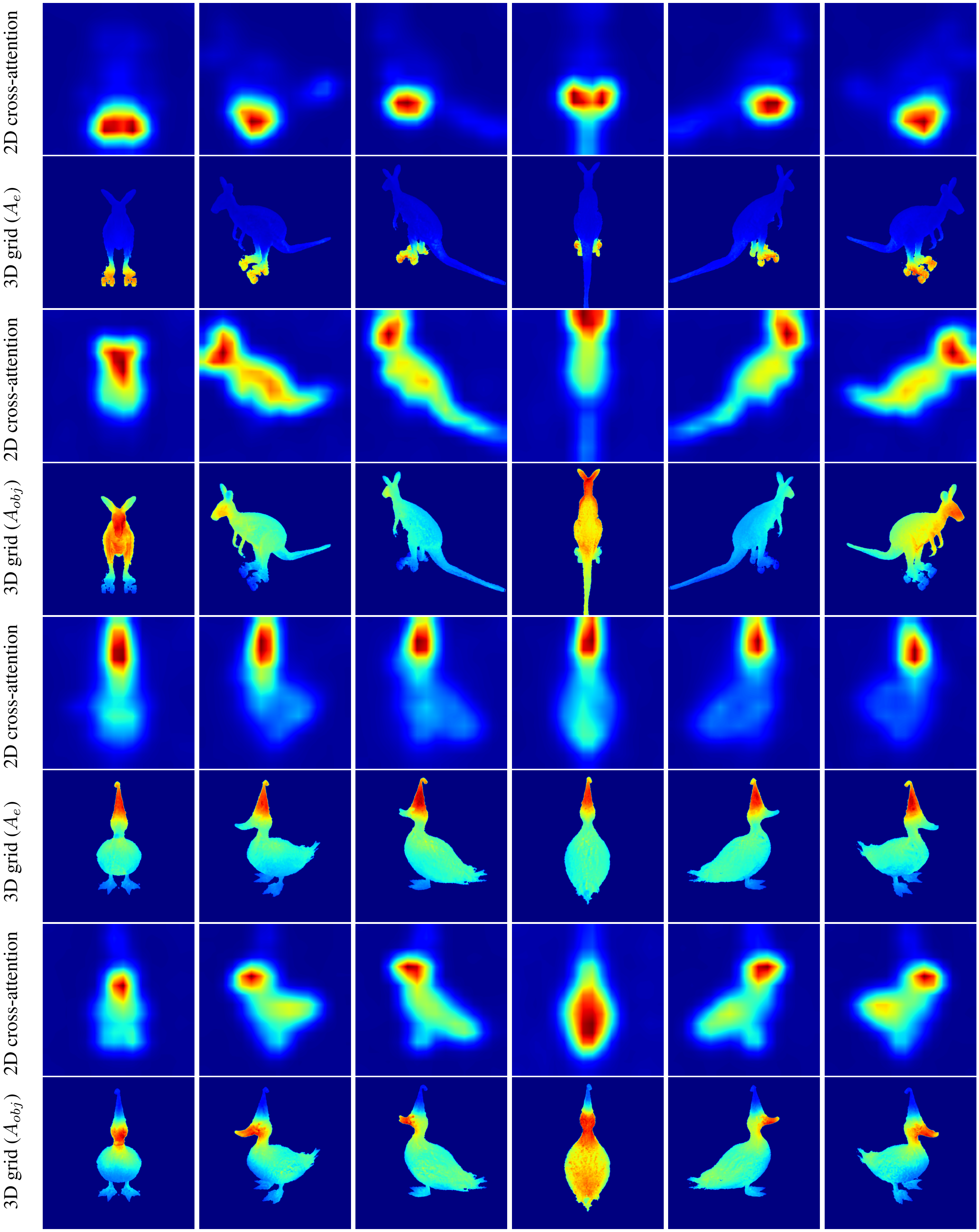


Figure 3. **Visualizing 2D cross-attention maps and 3d cross-attention grids over multiple viewpoints**. We visualize the optimized 3d cross-attention grids and the corresponding 2D cross-attention maps used as supervision. We show them for the edit region corresponding to the token associated with the word “rollerskates" (top two rows) and “hat" (fifth and sixth rows) and the object region (third and fourth rows for the kangaroo and bottom two rows for the duck).