CLIPascene: Scene Sketching with Different Types and Levels of Abstraction Supplementary Materials

Yael Vinker
Tel Aviv University
yaelvi116@gmail.com

Yuval Alaluf
Tel Aviv University
yuvalalaluf@gmail.com

Daniel Cohen-Or
Tel Aviv University
cohenor@gmail.com

Ariel Shamir
Reichman University
arik@runi.ac.il

Contents

1. Implementation Details **1**
1.1. Image Preprocessing 1
1.2. MLP Training 2
1.3. Matrix Composition 2

2. Additional Quantitative Analysis **2**
2.1. Sketch Recognizability 2
2.2. Number of Strokes by Simplicity Level . . . 2

3. Ablation Study: General Design Choices **4**
3.1. ViT vs. ResNet 4
3.2. Foreground-Background Separation 4

4. Ablation Study: Simplicity Axis **6**
4.1. Explicitly Defining the Number of Strokes . 6
4.2. Replace the Ratio Loss With a Target Number of Strokes 8
4.3. Fine-tuning MLP-loc During Simplification . 8
4.4. Defining the Function f-k as an Exponential . 9
4.5. Defining a Different Set of Factors for Each Layer 11
4.6. Defining a Different Sampling Step for Each f-k 11

5. Ablation Study: Fidelity Axis **11**
5.1. Using l-4 for Object Sketching 11
5.2. Using Other ViT Layers for Training 11

6. Additional Results **12**

7. Additional Comparisons **19**
7.1. Diffusion Models 19
7.2. Scene Sketching Approaches 19

1. Implementation Details

In this section, we provide specific details about the implementation of our method. We will further release all code and image sets used for evaluations to facilitate further research and comparisons.

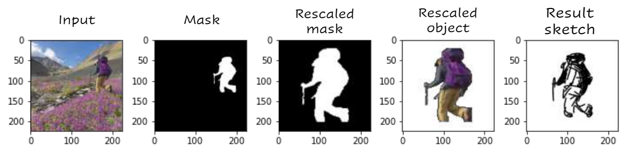

Figure 1. Object re-scaling procedure.

1.1. Image Preprocessing

As stated in the paper, we use a pre-trained U^2-Net salient object detector [9] to extract the scene's salient object(s). To receive a binary map, we threshold the resulting map from U^2-Net such that pixels with a value smaller than 0.5 are classified as background, and the remaining pixels are classified as salient objects. We then use this mask as an input to a pre-trained LaMa [15] inpainting model to recover the missing regions in the background image.

Object Scaling In the case where the saliency detection process detected only one object, and this object fills less than 70% of the image size, we perform an additional preprocessing step to increase the size of the object before sketching. This assists in sketching key features of the object when using a large number of strokes. Specifically, we first take the masked object and compute its bounding box. We then shift the masked object to the center of the image and resize the object such that it covers $\approx 70\%$ of the image. We apply the sketching procedure on the scaled object and then resize and shift the resulting sketch back to the original location in the input image. Note that since our sketches are given in vector representation, it is possible to re-scale and shift them without changing their resolution. This process is illustrated in Figure 1.

1.2. MLP Training

Hyper-parameters In all experiments, we set the number of strokes to $n = 64$ in the first phase of sketching and train MLP_{loc} for $2,000$ iterations. For generating the series of simplified sketches (Section 3.4 in the main paper), we perform 8 iterative steps. As discussed in Section 3.4, for each fidelity level k, we define a separate function f_k for defining the set of ratios used in $\mathcal{L}_{ratio}$. Along with this function, we define a separate step size for sampling the function f_k. For simplifying the background sketches, we set this step size to be $\{0.35, 0.45, 0.5, 0.9\}$ for layers $\{2, 7, 8, 11\}$, respectively. For simplifying the object sketches, we set the step sizes to be $\{0.45, 0.4, 0.5, 0.9\}$. Each simplification step is obtained by training MLP_{simp} and MLP_{loc} for 500 iterations. We employ the Adam optimizer with a constant learning rate of $1\text{e}{-4}$ for training both MLP networks. The input to MLP_{simp} is set to be a random-valued vector of dimension n.

Augmentations As also done in Vinker *et al.* [16], we apply random affine augmentations (*i.e.* random perspective and random cropping transformations) to both the input image and generated sketch before passing them as inputs to the CLIP model for computing the loss.

GradNorm We train MLP_{simp} and MLP_{loc} with three different losses simultaneously in order to achieve our visual simplifications. As these losses compete with each other, training has the potential to be highly unstable. For example, when training with multiple losses, the gradients of one loss may be stronger than the other, resulting in the need to weigh the losses accordingly. To help achieve a more stable training process and to ensure that each loss contributes equally to the optimization process, we use GradNorm [3], which automatically balances the training process by dynamically adjusting the gradient magnitudes. This balancing is achieved by weighing the losses inversely proportional to their contribution to the overall gradient.

1.3. Matrix Composition

As stated in the main paper, we separate the scene into two regions (based on their saliency map) and apply the sketching scheme to both independently, we then combine the resulting sketches to form the final matrix. To combine the foreground and background we simply aggregate the corresponding strokes at a given level of fidelity and simplicity. Note that we also export the mask used to separate them, if the user wish to locate it behind the object to avoid the collision of strokes. We also export the separate matrices, to allow users to combine sketches from different levels of abstraction as a post process.

2. Additional Quantitative Analysis

In this section, we provide additional details, examples, and results regarding the quantitative evaluations presented in the paper. First, in Figure 2 we present example inputs and representative generated sketches for each of our five scene categories used for evaluations.

2.1. Sketch Recognizability

To compute our recognizability metrics, we perform zero-shot classification using a pre-trained ViT-B/16 [4] CLIP model. Observe this model is different than the ViT-B/32 model used to generate sketches, ensuring a more fair evaluation of our sketches. When performing the zero-shot classification, we follow the evaluation setup used in CLIP [10] and apply 80 prompt templates when defining our 200 classes to CLIP's text encoder. This includes prompts of the form: "a rendering of a $\{\}$", "a drawing of a $\{\}$', and "a sketch of a $\{\}$". We then compute the cosine similarity between all text embeddings and the embedding corresponding to either our input image or generated sketches.

In Figure 4, we present example zero-shot classification results obtained on various input images and sketches across our five scene categories.

2.2. Number of Strokes by Simplicity Level

We examine our method's ability to generate sketches at varying levels of simplicity. For that purpose, we measure the final number of strokes used to generate the sketches. We extract this information from the generated sketch SVGs across all 560 sketches. We present the results in Figure 3, split between the different fidelity levels (indicated by different colors) and simplicity levels (shown along the x-axis). As can be seen, the number of strokes decreases as we move along the simplicity axis, across all fidelity levels.

In Figure 5, we present the same results but split between the different scene categories and split between composing the foreground and background sketches. As can be seen, the resulting functions for the different fidelity levels follow an exponential relation as we strengthen the simplification level.

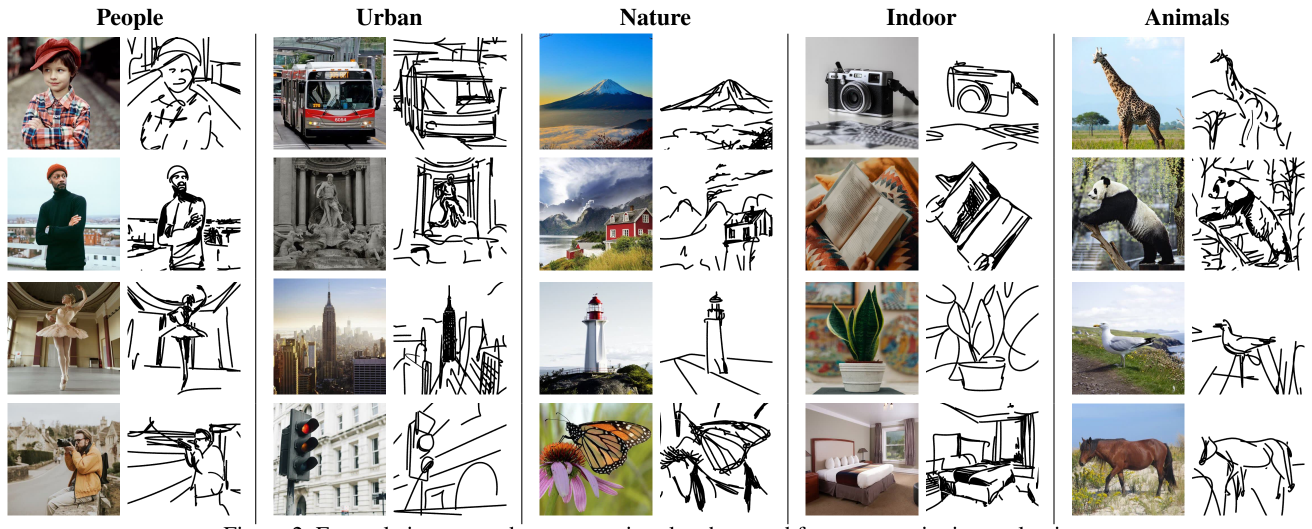

Figure 2. Example images and representative sketches used for our quantitative evaluations.

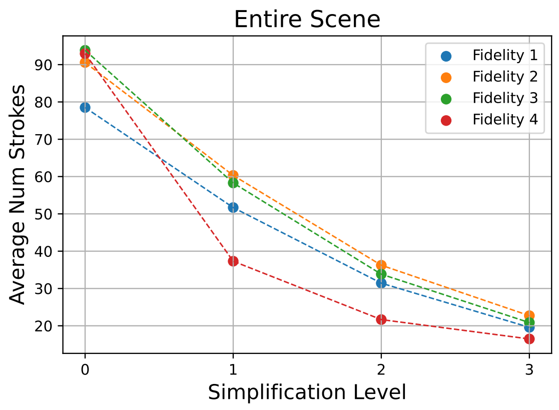

Figure 3. Examining the number of strokes used to compose the sketch across each fidelity and simplification level. Results are averaged across all images across all scene categories. In the supplementary materials, we additionally illustrate the number of strokes split between foreground and background and between the five scene categories.

Image Top 5: person, plain, skyscraper, bridge, tower
Sketch 1 Top 5: person, laptop, skyscraper, pencil, glasses
Sketch 2 Top 5: skyscraper, person, tower, statue, hair drier

Image Top 5: tower, statue, plain, castle, toilet
Sketch 1 Top 5: tower, house, statue, toilet, oven
Sketch 2 Top 5: house, tower, castle, bridge, lighthouse

Image Top 5: house, lighthouse, mountain, sea, plain
Sketch 1 Top 5: house, lighthouse, mountain, plain, kite
Sketch 2 Top 5: mountain, house, plain, snowboard, castle

Image Top 5: bed, couch, table, vase, chair
Sketch 1 Top 5: laptop, couch, table, remote, chair
Sketch 2 Top 5: plant, couch, vase, kettle, bowl

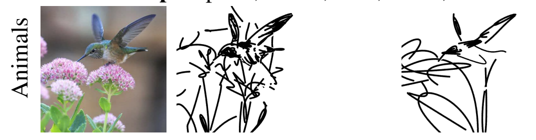

Image Top 5: bird, bee, flower, butterfly, camera
Sketch 1 Top 5: bird, bee, mouse, rabbit, butterfly
Sketch 2 Top 5: bee, butterfly, flower, bird, plant

Figure 4. Examples of CLIP zero-shot class predictions on various input images and representative sketches of varying abstractions. These predictions are then used to compute a recognizability metric for each scene category across different levels of abstractions (see Section 4.3 in the main paper).

3. Ablation Study: General Design Choices

3.1. ViT vs. ResNet

In Figure 6, we demonstrate the scene sketching results obtained with a ResNet101-based CLIP compared to those obtained with the ViT-based CLIP model employed in this work. When computing $\mathcal{L}_{CLIP}$ using a single layer of ResNet (*i.e.* layer 2, 3, or 4), we are unable to capture the input scene, indicating that a combination of the layers must be used for capturing the more global details of a complete scene. However, even when computing $\mathcal{L}_{CLIP}$ using multiple ResNet layers (as was done in CLIPasso), the network still struggles in capturing the details of the scene. For example, in row 3, although we are able to roughly capture the outline of the bull's head and horns, the network is unable to capture the bull's body and scene background. In contrast, when replacing the ResNet model with the more powerful ViT model, we are able to capture both the scene foreground and background, even when using a single layer for computing the loss. This naturally allows us to control the level of fidelity of the generated sketch by simply altering the single ViT layer that is used for computing $\mathcal{L}_{CLIP}$.

3.2. Foreground-Background Separation

In Section 3.4 of the main paper, we introduced our scene decomposition technique where the foreground and background components of the input are sketched separately and then merged. In Figure 7 we provide additional sketching results obtained with and without the scene decomposition for both abstraction axes. At the top, we present the resulting sketches along the fidelity axis. Observe how the house in the leftmost sketch in the first row appears to disappear within the entire scene. Furthermore, note the artifacts that appear in the face of the dog as the abstraction increases. In contrast, by explicitly separating the foreground and background, we can apply additional constraints over the foreground sketches to help mitigate unwanted artifacts. As a result, we are able to better maintain the correct structure of both the house and dog in the provided examples.

At the bottom of Figure 7 we show the resulting sketches along the simplification axis. Note how the house in the first row almost disappears completely, and that there are not enough strokes to depict the mountains in the background. By considering the entire scene as a whole, the model has no explicit control over how to balance the level of details placed between the object and the background. As a result, more strokes are typically used to sketch the background, which consumes a larger portion of the entire image (and therefore leads to a larger reduction in $\mathcal{L}_{CLIP}$).

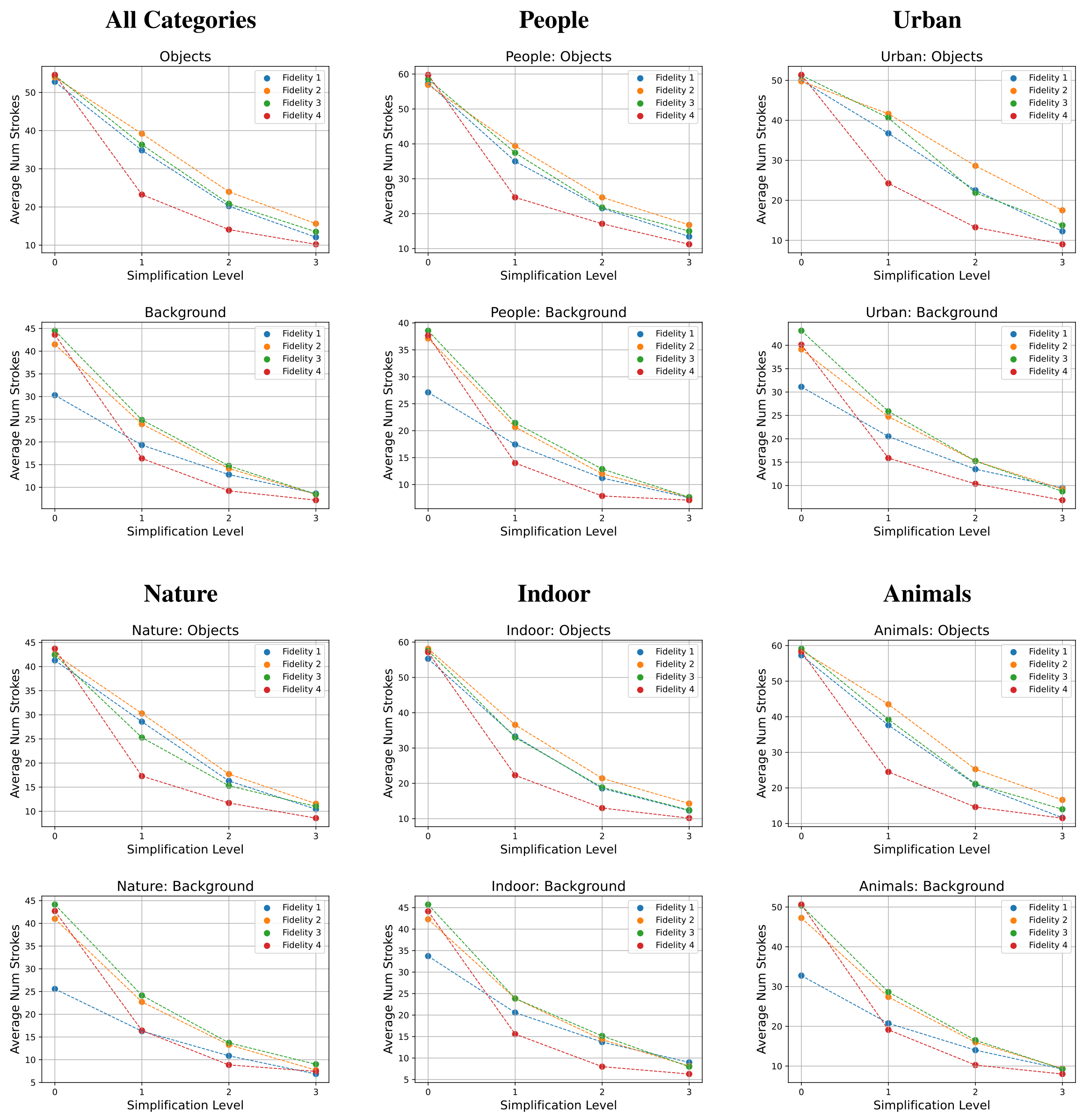

Figure 5. Examining the number of strokes used to compose the sketch across each fidelity and simplification level, split between the different scene categories.

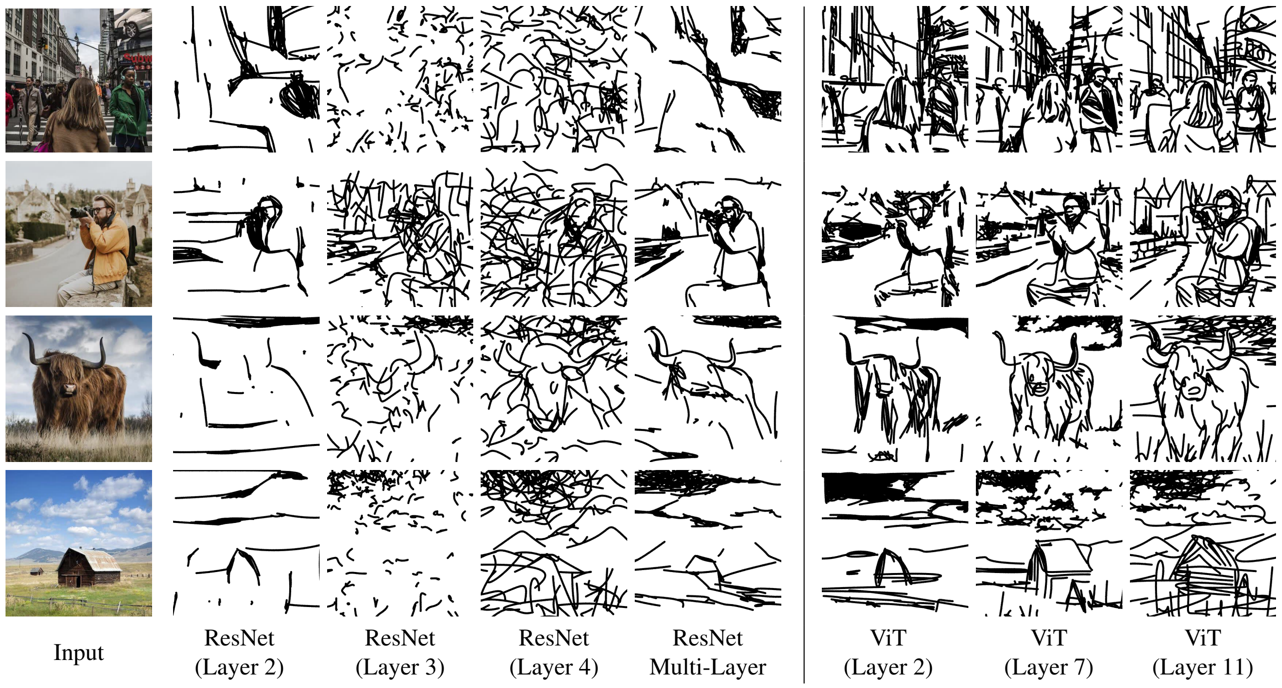

Figure 6. Ablation study on using ResNet-CLIP and ViT-CLIP for guiding the training process. For both variants, we generate the sketches for the entire scene together (*i.e.* no scene decomposition is performed) and set the number of strokes to 128. In addition to evaluating ResNet using a single layer for computing $\mathcal{L}_{CLIP}$, we additionally show results obtained when multiple ResNet layers are used to compute $\mathcal{L}_{CLIP}$ (marked as "ResNet Multi-Layer"). For this variant, we follow CLIPasso [16] as set the layer weights to $0, 0, 1, 1, 0$ for layers ℓ_1 to ℓ_5, respectively. In addition, we use the output of the fully connected layer and set its weight to 0.1 in the loss computation.

4. Ablation Study: Simplicity Axis

In this section, we analyze the design choices for the simplification training scheme and corresponding loss objectives.

4.1. Explicitly Defining the Number of Strokes

We begin by analyzing our simplification scheme as a whole. That is, are we able to achieve a smooth simplification of an input scene by simply varying the number of strokes used to sketch the scene? In Figure 8 we provide results comparing our implicit simplification scheme (on the right) with results obtained by sketching the scene using a varying number of strokes defined in advance (on the left). For the latter results, we use $64, 32, 16$, and 8 strokes for sketching. For each input, we present the simplifications achieved at the last level of fidelity (*i.e.* using layer 11 of CLIP-ViT for training).

Knowing in advance the number of strokes needed to achieve a specific level of abstraction is often challenging and varies between different inputs. For example, consider the image in the first row, containing a simpler scene of a house. When using only 16 strokes for sketching this image, we are able to capture the components of the scene such as the existence of the house in the center, its roof, and its door. However, when sketching the more complex urban scene in the third row, using 16 strokes may struggle to capture the general structure of the buildings or may not converge at all. By *learning* how to simplify each sketch, our simplification scheme is able to adjust the number of strokes needed to more faithfully sketch a given image at various levels of simplification, while adapting to the complexity of the input scene in the learning process.

Moreover, we observe that when defining the number of strokes explicitly, we may fail to get a smooth simplification of the initial sketch since each sketch is generated independently and may converge to a different local minimum. For example, in the second row on the left, the sketches do not appear to be simplified versions of each previous step (*e.g.* between the second and third steps), but rather new sketches of the input scene with an increased level of visual simplification. In contrast, each of our simplification results is initialized with the previous result, resulting in a smoother transition between each image.

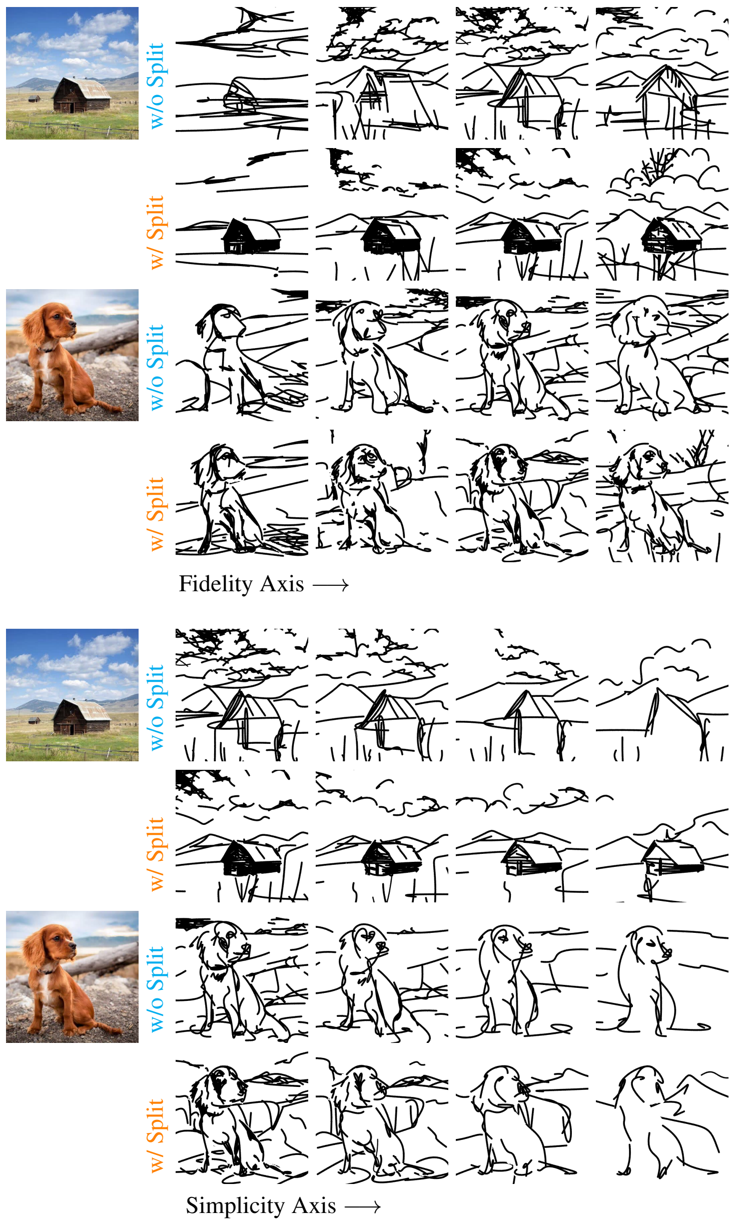

Figure 7. Scene sketching results obtained with and without decomposing the scene. We show sketches generated across both the fidelity axis (first four rows) and the simplicity axis (bottom four rows).

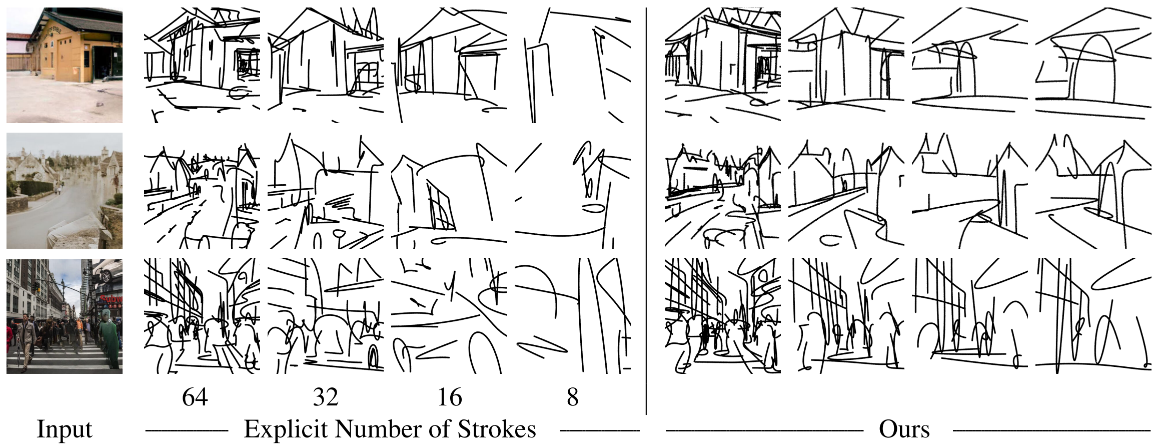

Figure 8. Ablation results of explicitly defining the number of strokes (left) compared to using our implicit simplification scheme (right). The presented sketches of all three input images are of the last fidelity level (obtained using layer 11 of CLIP-ViT).

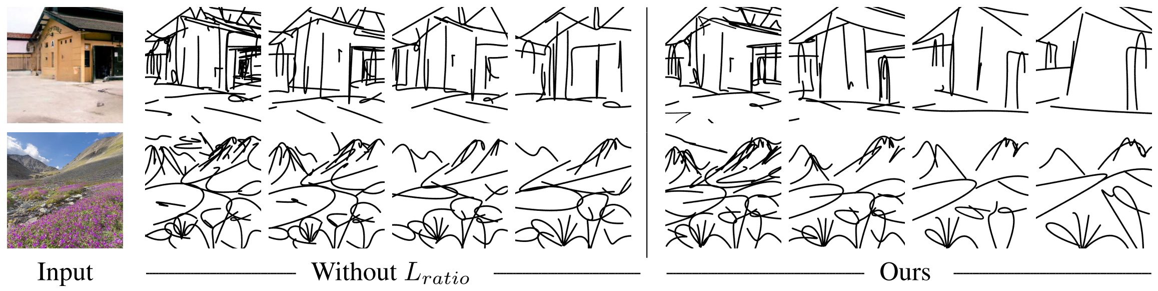

Figure 9. Ablation results of replacing our $\mathcal{L}_{ratio}$ loss with an alternative loss $||\mathcal{L}_{sparse} - n_{target}||$ guided by a desired number of strokes n_{target}.

4.2. Replace the Ratio Loss With a Target Number of Strokes

We note in the main paper that achieving gradual visual simplification requires balancing between $\mathcal{L}_{sparse}$ and $\mathcal{L}_{CLIP}$. In our method, we do so by defining a set of factors used to define a balance between the relative strengths of the losses. This section examines another possible approach: encouraging the training process to achieve a certain number of strokes during training and reducing this number at each level. As opposed to Section 4.1 where we restrict the number of strokes completely, here we include the target number of strokes as another objective in the training process, thus allowing deviance from this number.

To implement this approach we simply redefine L_{ratio} as follows:

$$\mathcal{L}_{ratio} = ||\mathcal{L}_{sparse} - n_{target}||, \tag{1}$$

where n_{target} is the desired number of strokes. Specifically, we define four levels of abstraction using 64, 32, 16, and 8 strokes. We then normalize the number of strokes to be between 0 and 1 and define n_{target} as $\{1, 0.5, 0.25, 0.125\}$ for each level, respectively.

In Figure 9 we show the result of this experiment. As can be seen on the left, such an approach does not achieve the desired simplification. As can be seen, on the left, the levels of abstraction are less gradual than on the right and do not reach full abstraction. This approach also relies on an arbitrary fixed number of strokes per abstraction level for all images, as opposed to allowing the ratio itself to implicitly define it in a content-dependent manner.

4.3. Fine-tuning MLP-loc During Simplification

As described in Section 3.3 of the main paper, when performing the simplification of a given sketch by training MLP_{simp}, we continue fine-tuning MLP_{loc}. Doing so is important since by training MLP_{loc}, we allow to slightly adjust the locations of strokes in the canvas, which helps encourage the simplified sketch to resemble the original input image. In Figure 10, we show sketch simplifications across various inputs obtained with and without the fine-tuning of MLP_{loc}.

When MLP_{loc} is held fixed, the simplification process is equivalent to simply selecting a subset of strokes to remove at each step. This approach will result in the appearance of visual simplification, but may not be sufficient to maintain the semantics of the scene. For example, in the last simplification step of the first example, the mountains in the background have disappeared, as have the buildings in the third example. In addition, the house in the second image can no longer be identified. On the other hand, our results, obtained with the fine-tuning of MLP_{loc}, produce the desired visual simplification, while still preserving the same semantics of the input scenes.

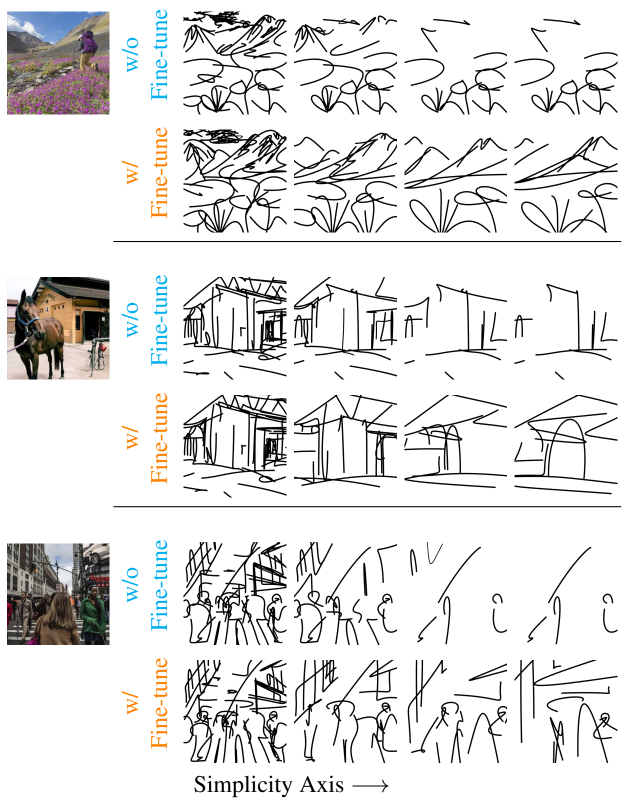

Figure 10. Ablation of fine-tuning MLP_{loc} when training MLP_{simp} during the simplification process. The sketches presented here are obtained using layer 11 of the CLIP-ViT model.

4.4. Defining the Function f-k as an Exponential

In Section 3.3 of the main paper, we describe the process of selecting the set of factors used to achieve a gradual simplification of a given sketch. To do so, we defined a function f_k for each fidelity level k defining the balance between $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{sparse}$. We find that an f_k that models in an exponential relationship between L_{CLIP} and L_{sparse} achieves a simplification that is perceived smooth.

In this section, we demonstrate this effect by visually demonstrating the gradual simplification we achieve when choosing an f_k that gives a linear relation between $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{sparse}$. To do so, we define the linear f_k such that the sampled set of factors $\{r_k^1, ..r_k^m\}$ represent a constant step size by encouraging the removal of 8 strokes in each step.

In Figure 11 we present the results of this alternative setup. For each set of generated sketches, we additionally present two graphs: (1) the resulting L_{sparse} as a function of L_{CLIP} (left), and (2) the final number of strokes as a function of the simplification step (right). Each point in the graphs corresponds to a single sketch with the color of the points indicating the location of the corresponding sketch along the simplification axis. That is, 0 (or dark blue) indicates the leftmost, non-simplified sketch while 7 (or yellow) indicates the rightmost sketch with the highest level of simplification. Recall, as discussed in the main paper, the left graph should ideally depict an exponential relation between the two loss objectives in order for the simplification to appear smooth.

The results presented on the left side of Figure 11 show the sketches and corresponding graphs produced when using a linear f_k as defined above. The results on the right-hand side of the Figure show sketches obtained with our method when using the exponential f_k as described in the main paper. As can be seen, the sketches in the linear alternative (left) remain too detailed at the initial abstraction levels and do not convey the smooth and gradual change perceptually as is present with the exponential function (right).

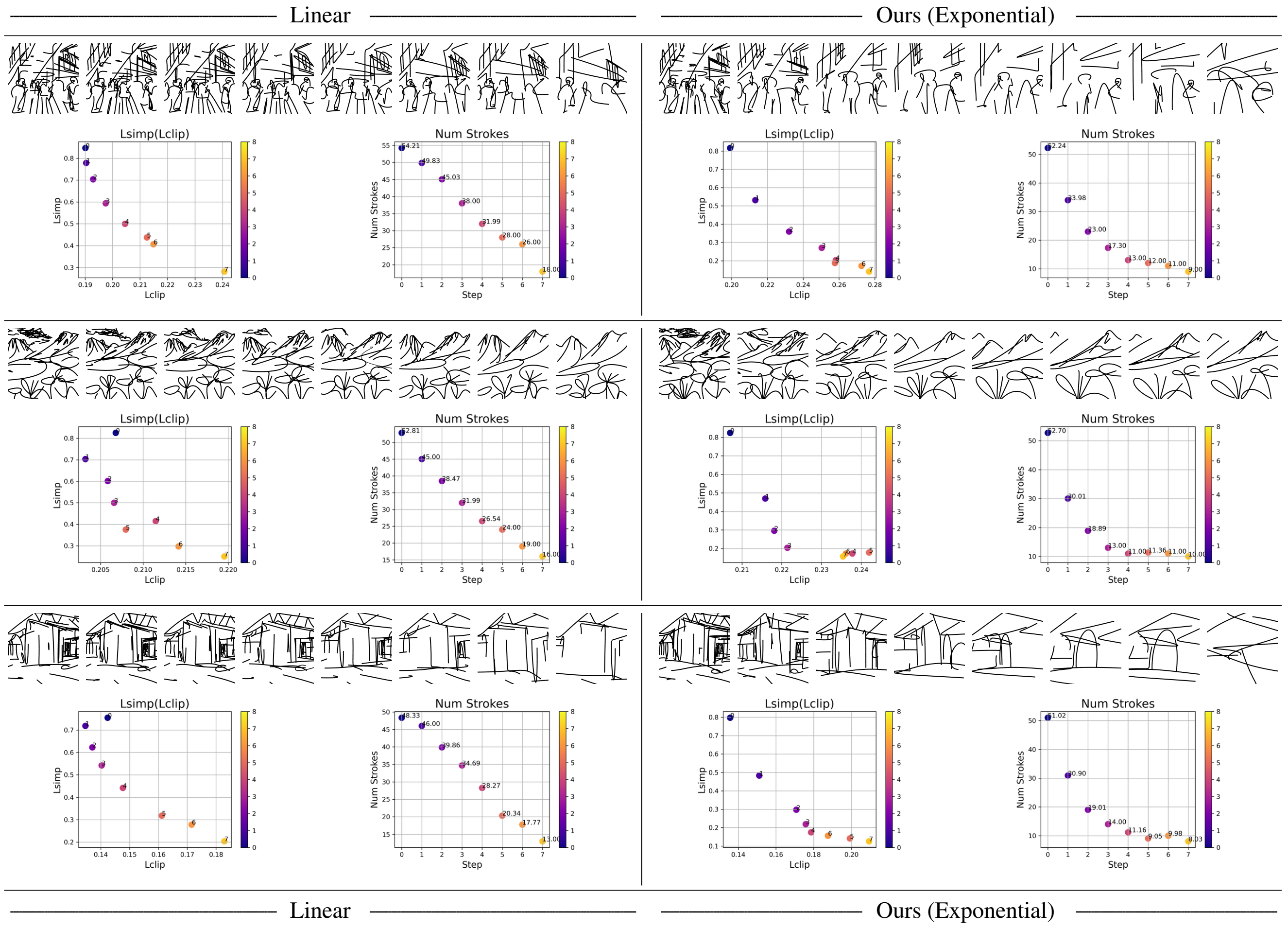

Figure 11. Ablation of the defining an exponential f_k when performing an iterative simplification of the sketch. On the right-hand side, we define a linear relation between the two loss objectives. On the left side, we show sketch results obtained when defining an exponential relation between $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{sparse}$. For each set of simplified sketches, we additionally present two graphs depicting (1) the relation between the two loss objectives at each simplification step (left graph) and (2) the number of strokes used to compose the sketch (right graph).

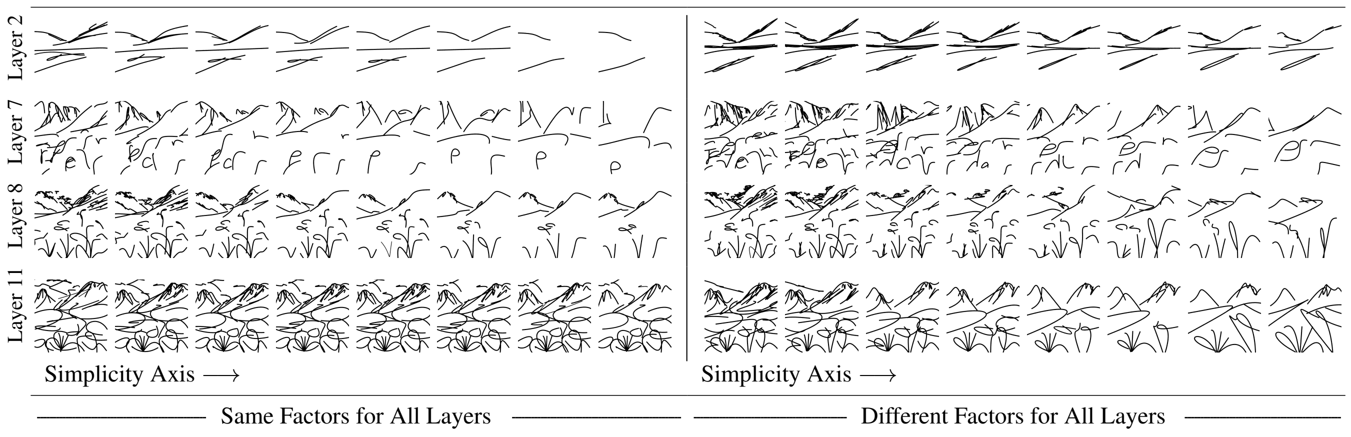

Figure 12. Ablation study on using a different set of factors for each fidelity level k when defining our $\mathcal{L}_{ratio}$ loss. On the left, we should simplification results obtained when applying the same set of factors across all levels. On the right side, we should our simplification results obtained by adjusting the factors for each fidelity level.

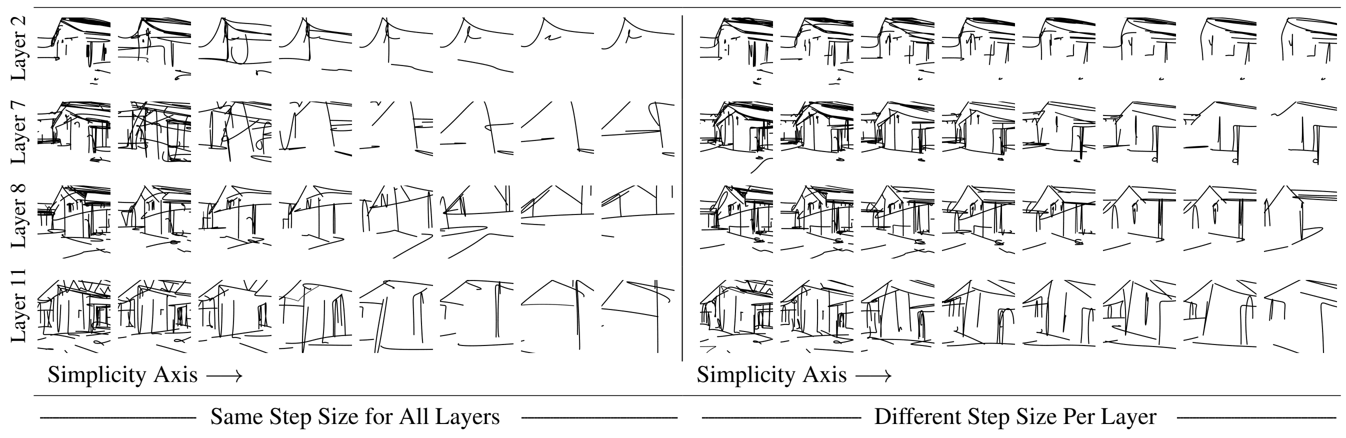

Figure 13. Ablation study on applying the same step size for sampling each function f_k during simplification. On the left, we apply the same step size across all fidelity levels, resulting in non-smooth and non-uniform simplifications between the different fidelity levels. On the right, we show our simplification results obtained by adjusting the step size for each level. As shown, we achieve smoother simplifications that are more consistent between the four different levels.

4.5. Defining a Different Set of Factors for Each Layer

In Section 3.3 of the main paper, we describe the process of selecting the set of factors $\{r_k^1, ..., r_k^m\}$ used to achieve a gradual simplification of a given sketch. In this section, we validate the use of different sets of factors for each fidelity level k. Note that, as stated in the main paper, the set of factors r_k^j determine the balance between $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{sparse}$, which directly determines the level of visual simplification.

We show on the left-hand side of Figure 12 the simplified sketches obtained for different levels k of fidelity when using the same set of factors. Specifically, we apply the set of factors used for layer ℓ_8 to the remaining ViT layers. As can be seen, the perceived level of visual simplification is not uniform between different layers: the simplification achieved for layer ℓ_{11} is too weak with very little perceived change realized across all steps. In contrast, for layer ℓ_2 the simplification is too strong with the background quickly "disappearing" as we move to the right. On the right-hand side of Figure 12, we present the results obtained with our method, where we fit a dedicated set of factors for each fidelity level k. As can be seen, the perceived level of abstraction among the different fidelity levels is more uniform and smooth.

4.6. Defining a Different Sampling Step for Each f-k

After defining the function f_k for each fidelity level k, we use a different step size for sampling this function for each k. We wish to achieve a similar appearance of simplification across different fidelity levels, and we find that in order to do so, using a different step size for sampling f_k is crucial.

In Figure 13, on the left, we demonstrate results obtained when using the same step size for our four fidelity levels (*i.e*. ViT layers $\ell_2, \ell_7, \ell_8, \ell_{11}$). As can be seen, for layers ℓ_2 and ℓ_7 the gradual simplification is not smooth and a noticeable jump in the strength of the abstraction can be seen between the second and third sketches. Furthermore, for layer ℓ_7, the change in step size caused the networks to converge to a noisy solution. Observe how the second sketch does not resemble a simplification of the previous one.

In contrast, the results on the right are obtained using the proposed approach of selecting different step sizes for each fidelity level k. As shown, the sketches do not suffer from the perceived artifacts present on the left. Moreover, the simplification results are also smooth and uniform in appearance between the different layers.

5. Ablation Study: Fidelity Axis

Finally, in this section, we perform various ablation studies to validate the design choices made with respect to our fidelity abstraction axis.

5.1. Using l-4 for Object Sketching

When decomposing the scene and sketching for the foreground image, we additionally compute L_{CLIP} over layer ℓ_4 of ViT. We found that doing so may help in preserving the geometry of more complex subjects, as illustrated in Figure 15. This is most noticeable in finer details such as in the facial details of the old man and the dog or the body shape of the panda, for example.

5.2. Using Other ViT Layers for Training

In order to obtain different levels of fidelity, we train MLP_{loc} guided by different layers of the CLIP-ViT model for computing $\mathcal{L}_{CLIP}$. Our model is based on the ViT-B/32

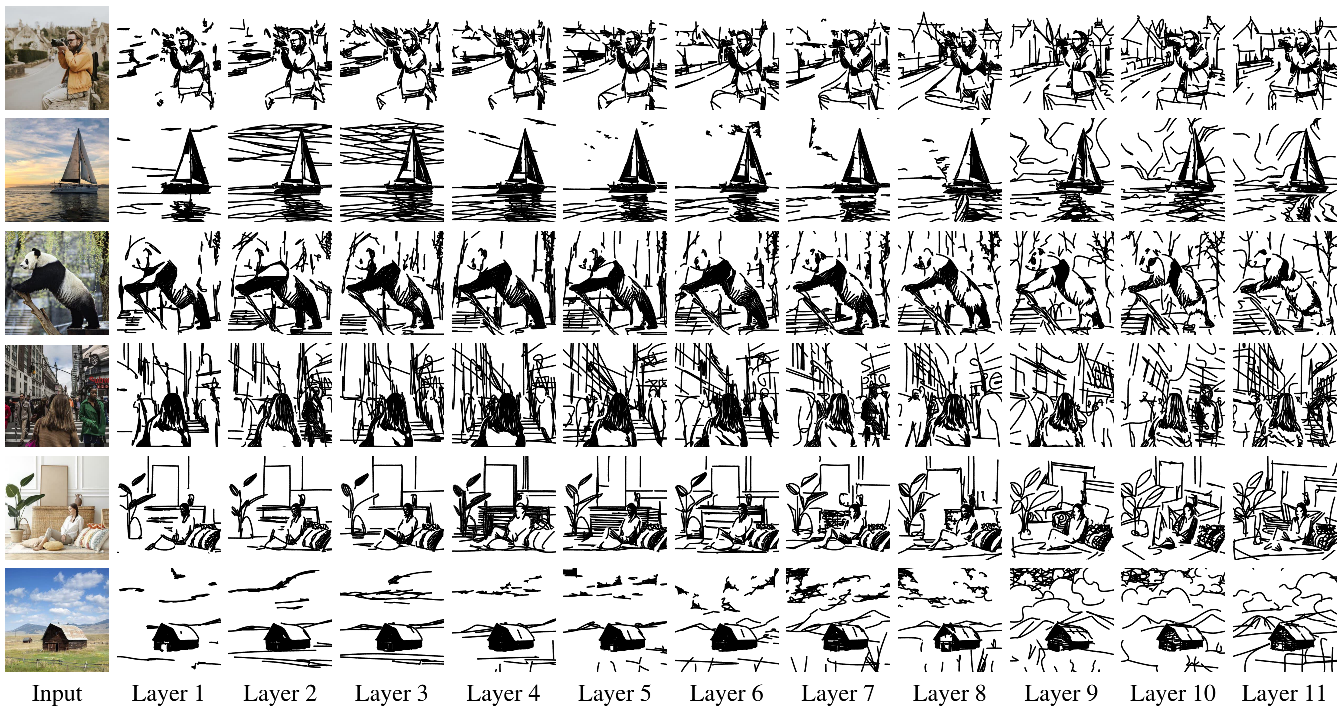

Figure 14. Ablation study on using different ViT layers for computing $\mathcal{L}_{CLIP}$ for generating sketches at different levels of fidelity.

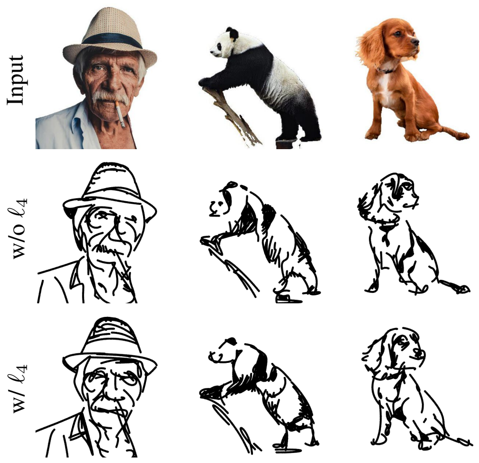

Figure 15. Ablation results for object sketching when additionally using layer ℓ_4 when computing $\mathcal{L}_{CLIP}$ for object sketching.

architecture that includes 11 intermediate layers. Our main paper presents the results of applying our training scheme to a subset of four layers: 2, 7, 8, and 11. This subset of layers represents a range of possible fidelity levels that can be achieved by our method. While we focus on presenting results using only these four layers, our method can naturally generate additional levels of fidelity by using the remaining intermediate layers. We present the results of using additional layers in Figure 14.

6. Additional Results

We begin with additional results generated by our method. In Figures 16 to 20 we provide 4×4 abstraction matrices for various scene images. In addition, we present additional examples of the added control provided by the separation technique in Figure 21. This includes: (1) editing the style of strokes using Adobe Illustrator and (2) combining the foreground and background sketches and varying levels of abstractions to achieve various artistic effects.

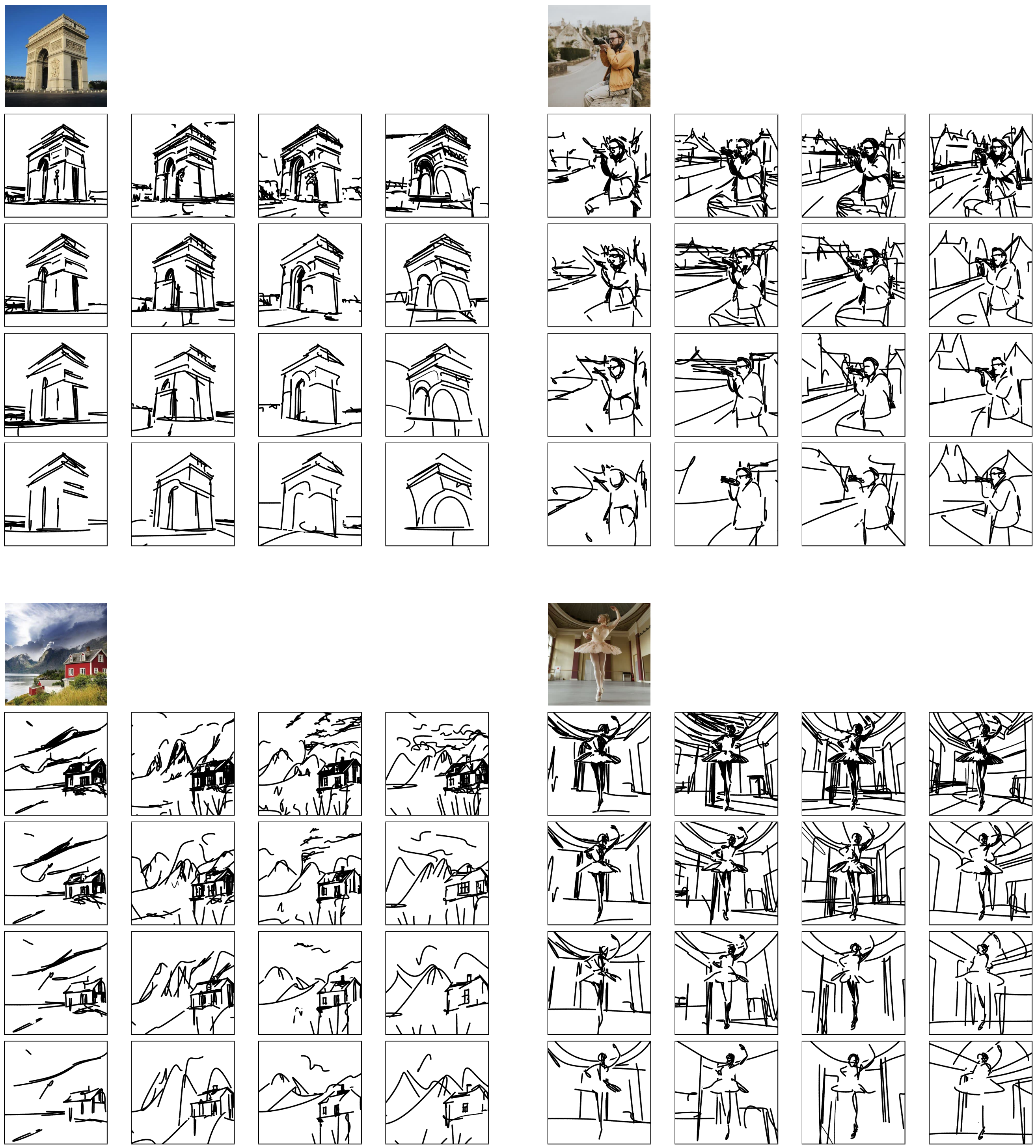
Figure 16. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.

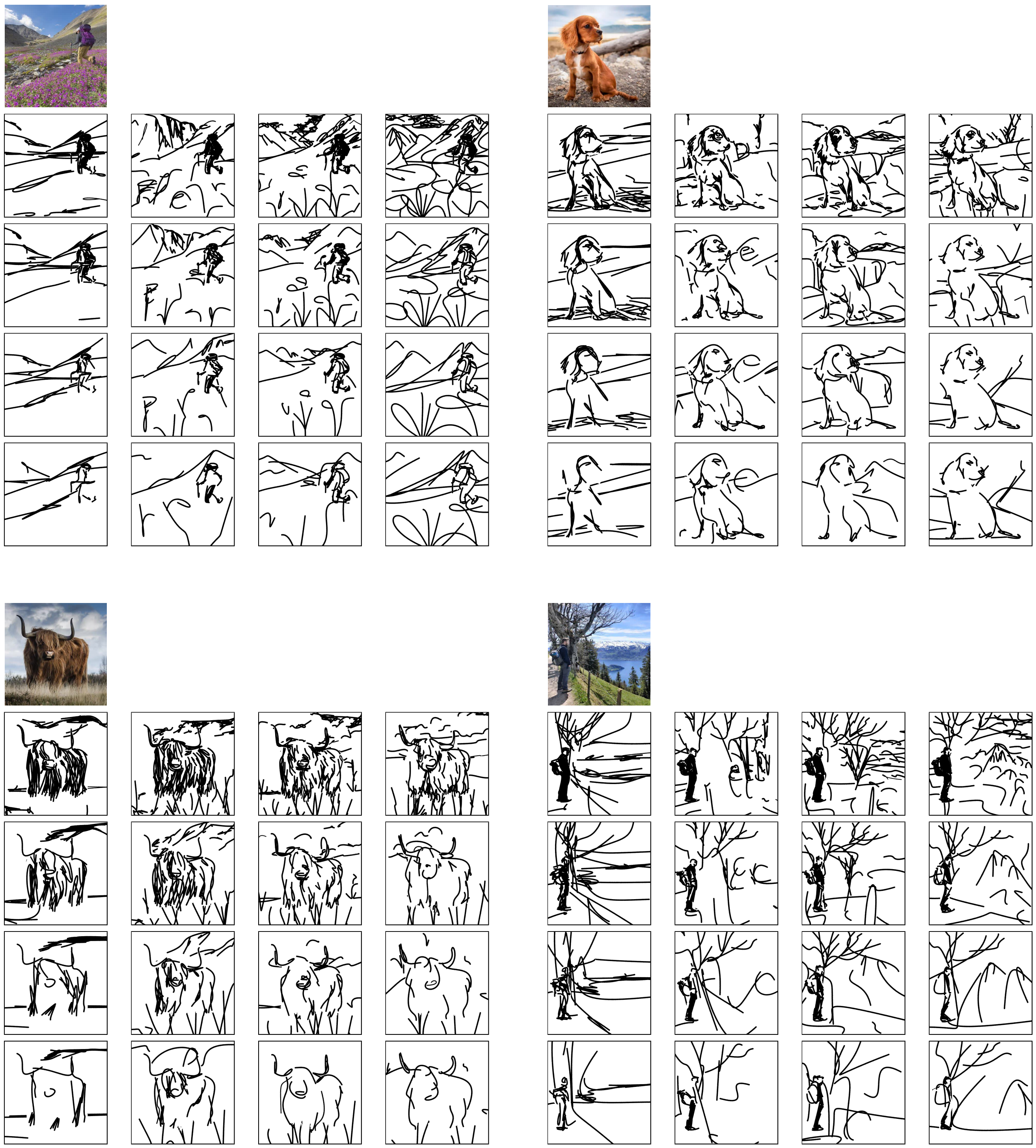
Figure 17. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.

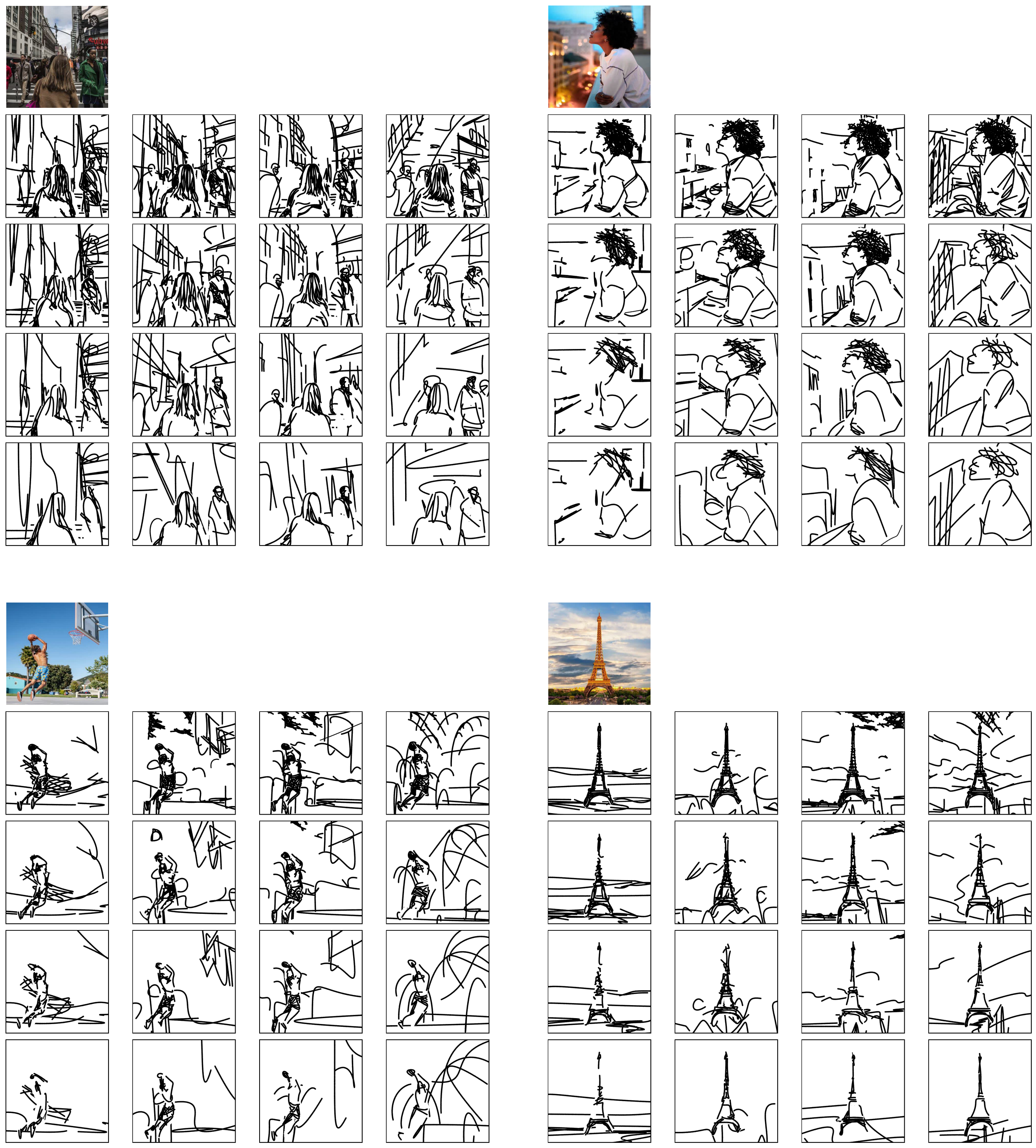
Figure 18. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.

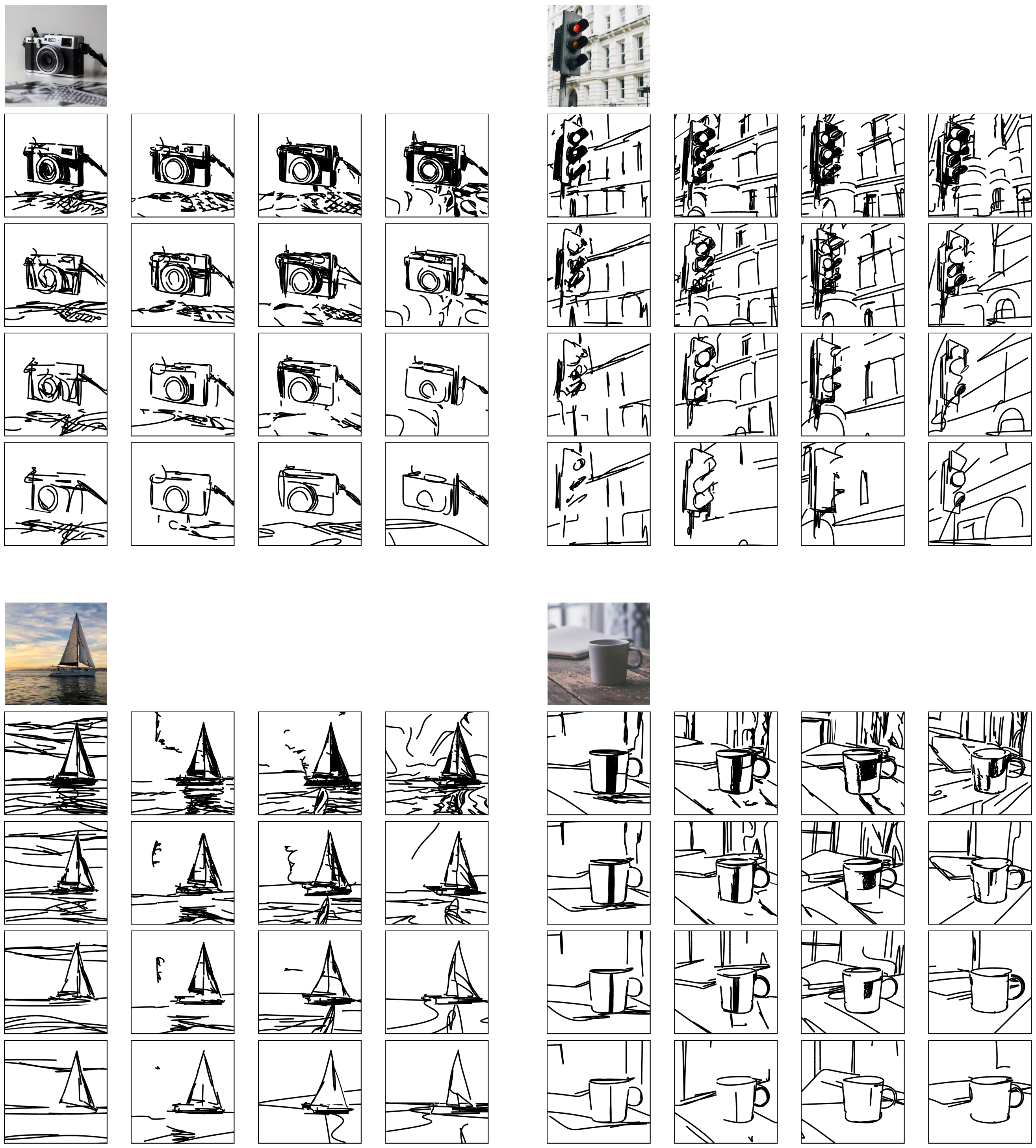

Figure 19. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.

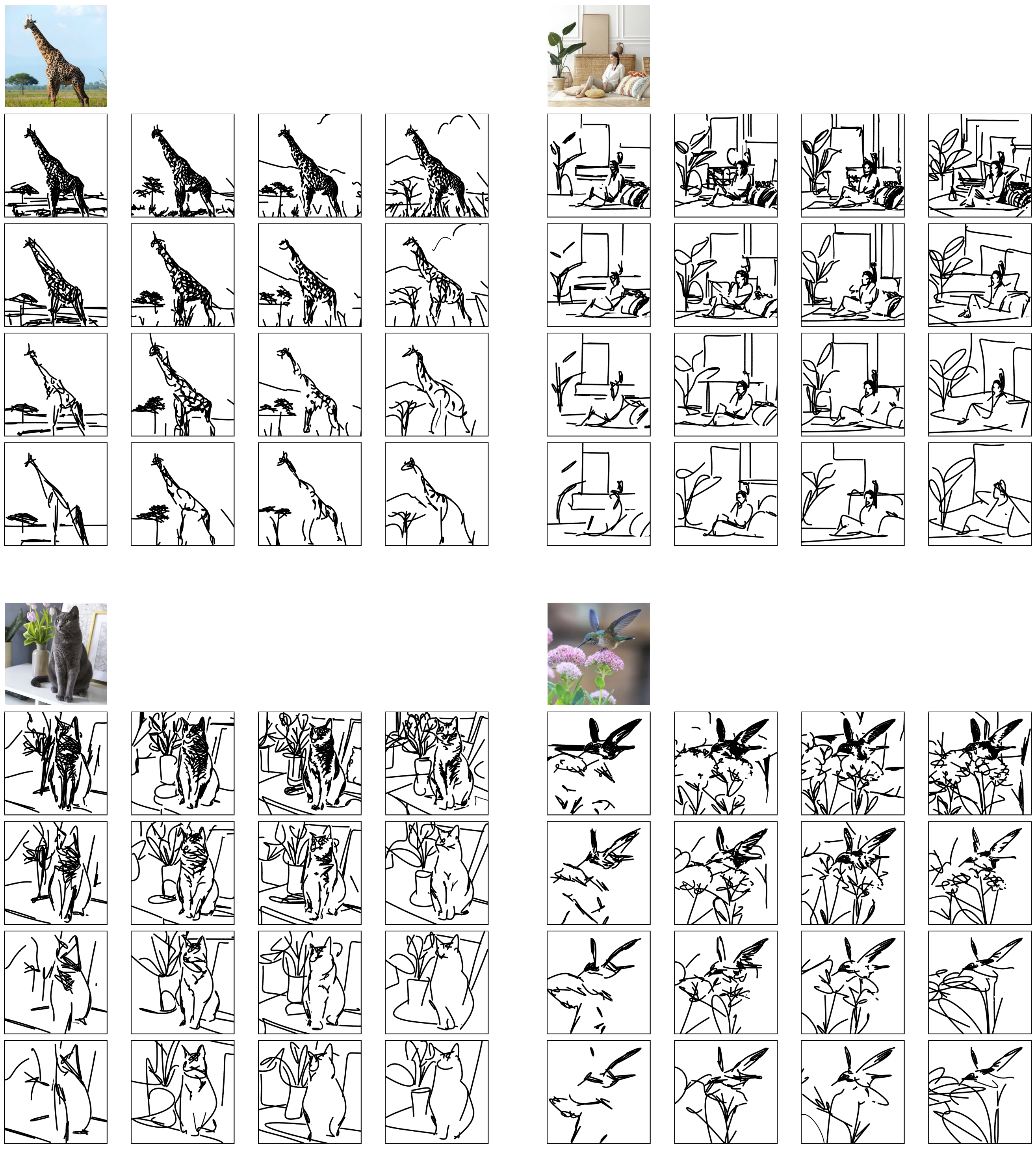
Figure 20. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.

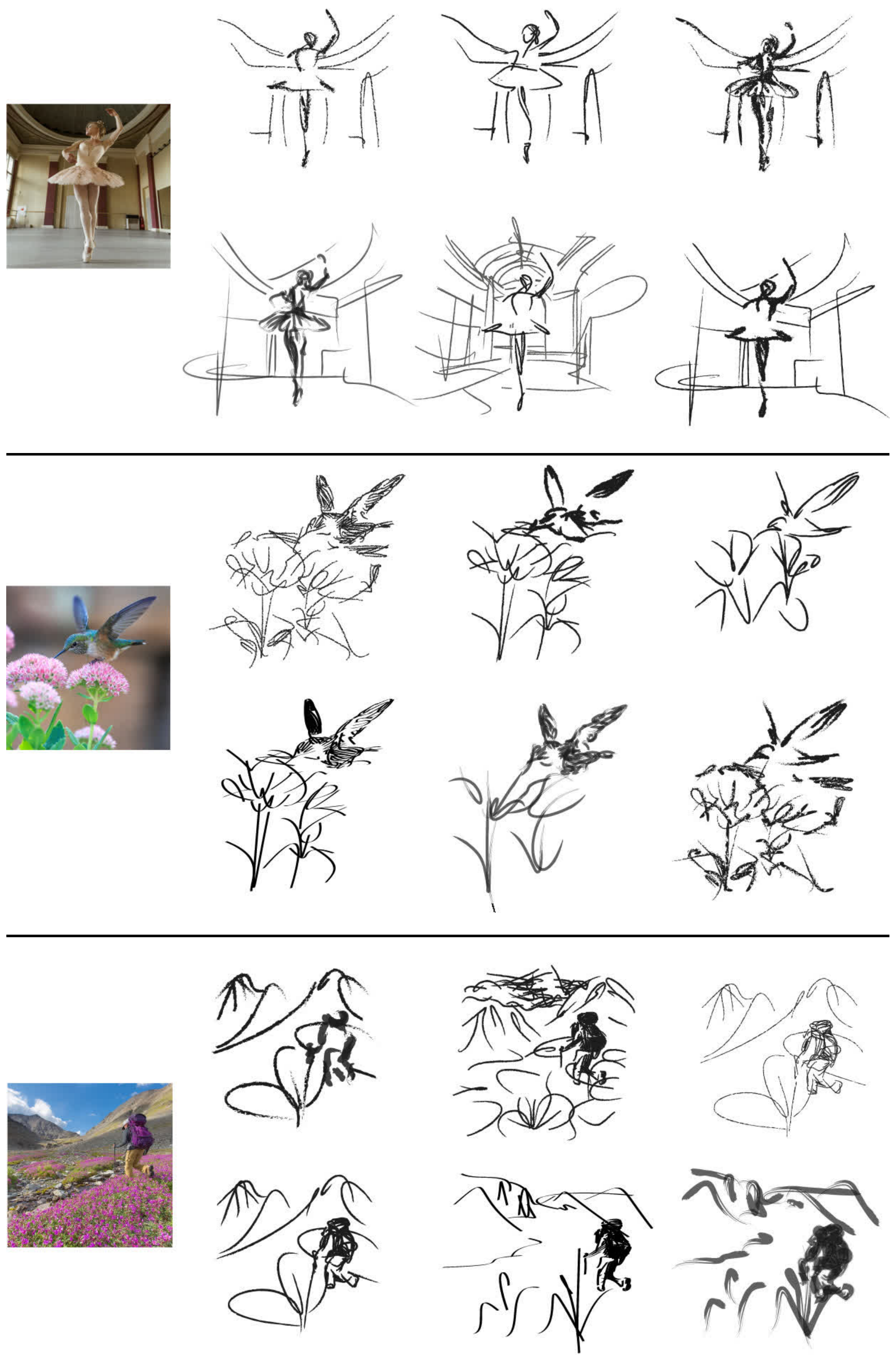

Figure 21. Additional control. For each image, we combine foreground and background sketches from different levels of abstraction, and edit the style of strokes using Adobe Illustrator. This illustrates the power of our method in providing various options for the user to edit the resulted sketches.

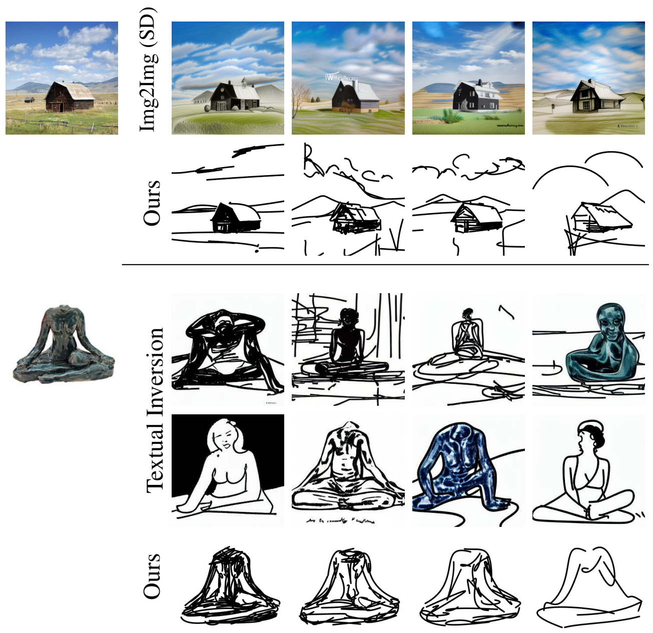

Figure 22. Comparison to various diffusion model-based techniques. At the top, we compare our sketch results to those obtained using Stable Diffusion [13] image-to-image technique guided by the text prompt "A black and white sketch image". At the bottom, we compare with Textual Inversion [5] by learning new tokens representing a detailed sketch (top row) and abstract sketch (bottom) row. As shown, both approaches struggle in either capturing the desired sketch style or input subject.

7. Additional Comparisons

7.1. Diffusion Models

Recent advancements in diffusion models [6] have demonstrated an unprecedented ability to generate amazing imagery guided by a target text prompt or image [13, 12, 11, 8, 18, 14]. In this section, we explore whether such models can be leveraged to generate abstract sketches of a given scene. We begin by exploring the recent Stable Diffusion model [13]. Given an input image, we perform a text-guided image-to-image translation of the input using text prompts such as: "A black and white sketch image" and "A black and white single line abstract sketch." Results are illustrated in Figure 22. As can be seen in the top row, Stable Diffusion struggles in capturing the sketch style even when guiding the denoising process with keywords such as "A black and white sketch". We do note that better results may be achieved with heavy prompt engineering or tricks such as prompt re-weighting methods [1]. However, doing so would require heavy manual overhead for each input image.

Another approach to assist in better capturing the desired sketch style would be to fine-tune the entire diffusion model on a collection of sketch images. However, this would require collecting a few hundred or thousands of images with matching captions and training a separate model for each desired style and level of abstraction.

As another diffusion-based approach, we consider the recent Textual Inversion (TI) technique from Gal *et al.* [5]. Given a few images (*e.g.* 5) of the desired style (*e.g.* sketch), TI can be used to learn a new "word" representing the style. Users can then use a pre-trained text-to-image model such as the recent Latent Diffusion Model [13] to generate images of the learned style. For example, users can generate a sketch image of a house using the prompt "A photo of a house in the style of S_*" where S_* represents our learned sketch style.

To evaluate TI's ability to generate sketch images supported by our method, we collect 10 sketches generated by our method — 5 detailed and 5 abstract — and learn a new token representing each of the sketch styles. In a similar fashion, we can learn a new word representing a unique object of interest (*e.g.* the headless statue shown in Figure 22). We can then generate images of the learned object in our learned style using prompts of the form "A drawing of a S_{statue} in the style of $S_{detailed}$" or "A drawing of a S_{statue} in the style of $S_{abstract}$". Example results are presented in the bottom half of Figure 22. As can be seen, TI struggles in composing both the learned style and subject in a single image. Specifically, TI either struggles in capturing the unique shape of the statue (*e.g.* its missing head) or struggles in capturing the learned sketch style (*e.g.* TI may generate images in color). In contrast, our method is able to generate a range of possible sketch abstractions that successfully capture the input subject.

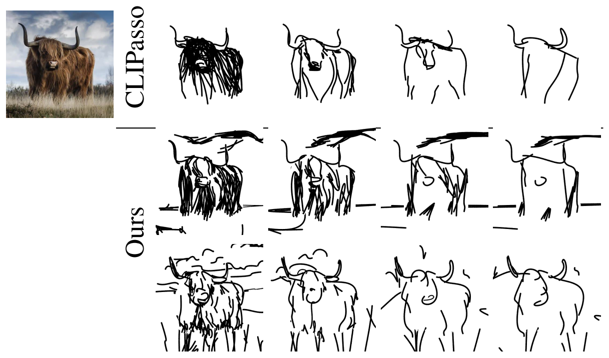

Figure 23. Comparisons with CLIPasso [16]. We show CLIPasso results obtained when applied over an entire scene image using 128 and 64 strokes respectively. We show our sketch results obtained using approximately the same number of strokes.

7.2. Scene Sketching Approaches

In Figure 23, we provide a comparison to CLIPasso. We applied CLIPasso on the masked object and obtained the abstraction by explicitly specifying the number of strokes (*i.e.* 64, 32, 16, and 8 strokes). In the second and third rows we show the simplification results obtained by our method, at two different fidelity levels. Since CLIPasso only offers a single axis of abstraction (mostly governed by simplification), the fidelity level of the sketch can not be explicitly controlled. Additionally, unlike CLIPasso, where the user must manually determine the number of strokes required to achieve different levels of abstraction, our approach *learns* the desired number of strokes.

Lastly, observe that since each sketch of CLIPasso is generated independently, the resulting sketches may not portray a gradual, smooth simplification of the sketch since each optimization process may converge to a different local minimum. By training an MLP network to *learn* this gradual simplification, our resulting sketches depict a smoother simplification, where each sketch is a simplified version of the previous one.

In Figure 24 we provide additional scene sketching comparisons to alternative scene sketching methods. In Figure 25 we provide additional sketch comparisons to all styles supported by UPDG [17] and Chan *et al*. [2]. In Figure 26 we provide additional comparisons to CLIPasso. In Figures 27 to 29 we show the 35 sketches produced by the different sketch approaches used for the quantitative experiment. Note that in Figure 28 we show the results obtained by CLIPasso when using our scene decomposition technique, specifically, we separate the input images into foreground and background and use CLIPasso to sketch each image separately, and then combine the results.

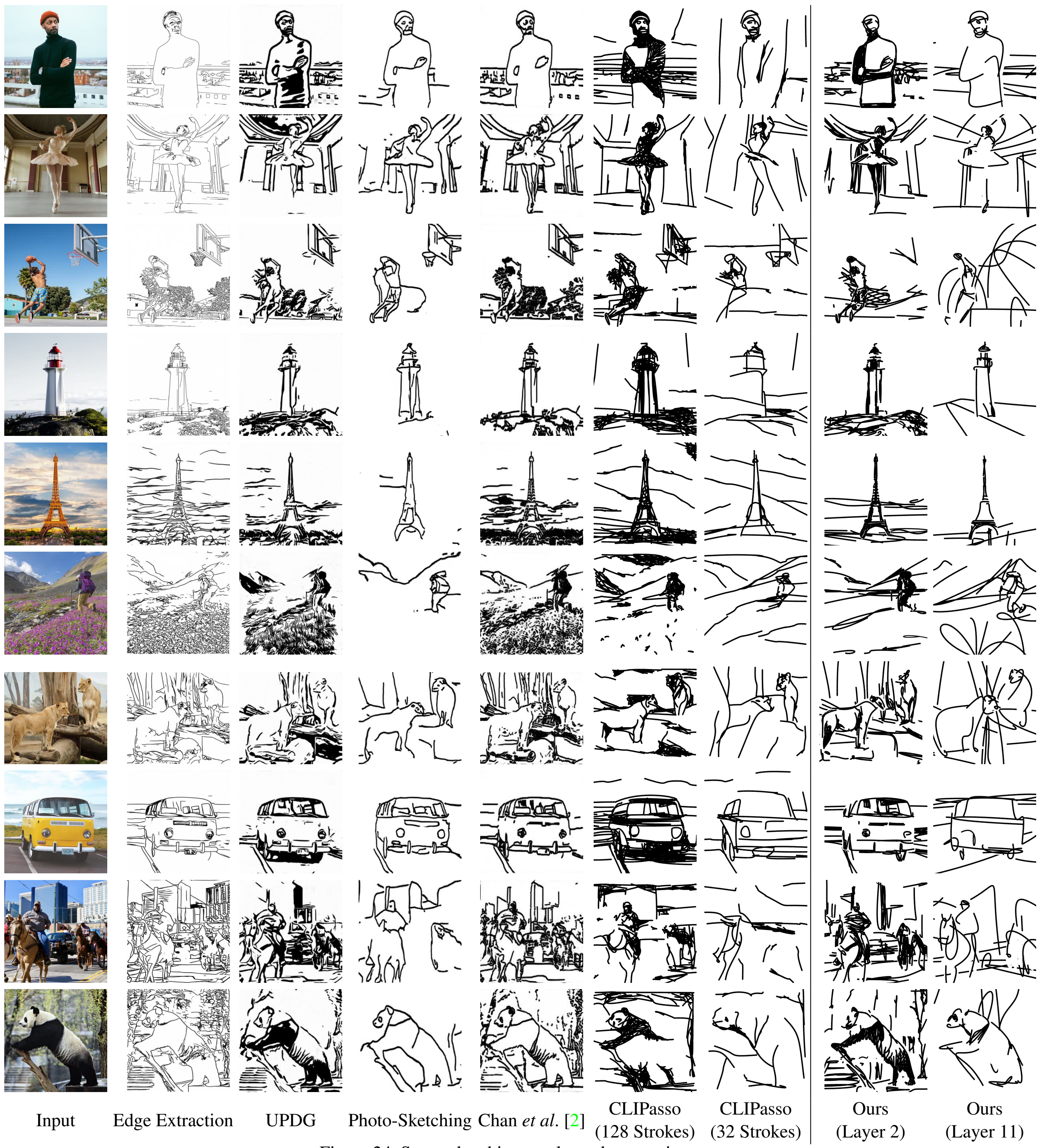

Figure 24. Scene sketching results and comparisons.

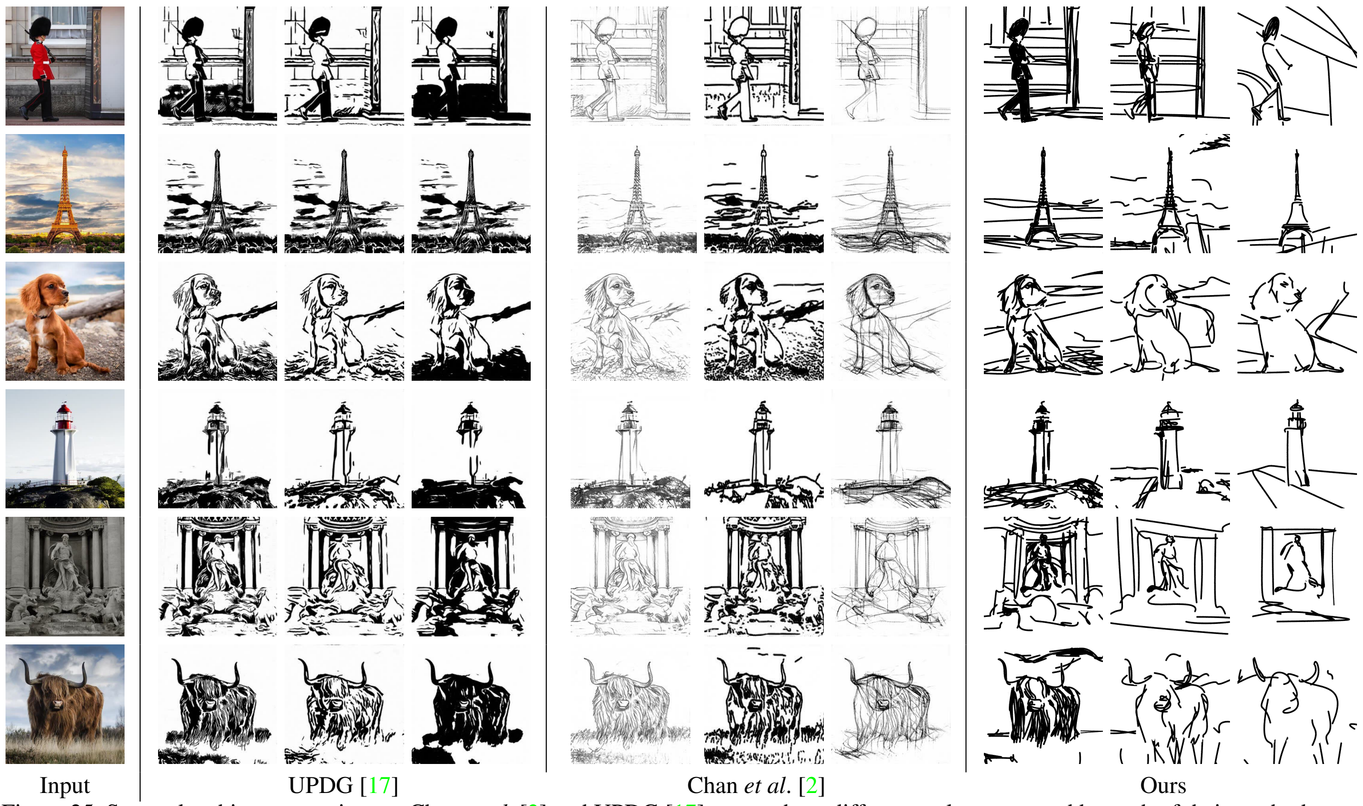

Figure 25. Scene sketching comparison to Chan *et al*. [2] and UPDG [17] across three different styles supported by each of their methods. For our results, we show three sketches illustrating the various levels of abstraction that our method is capable of achieving.

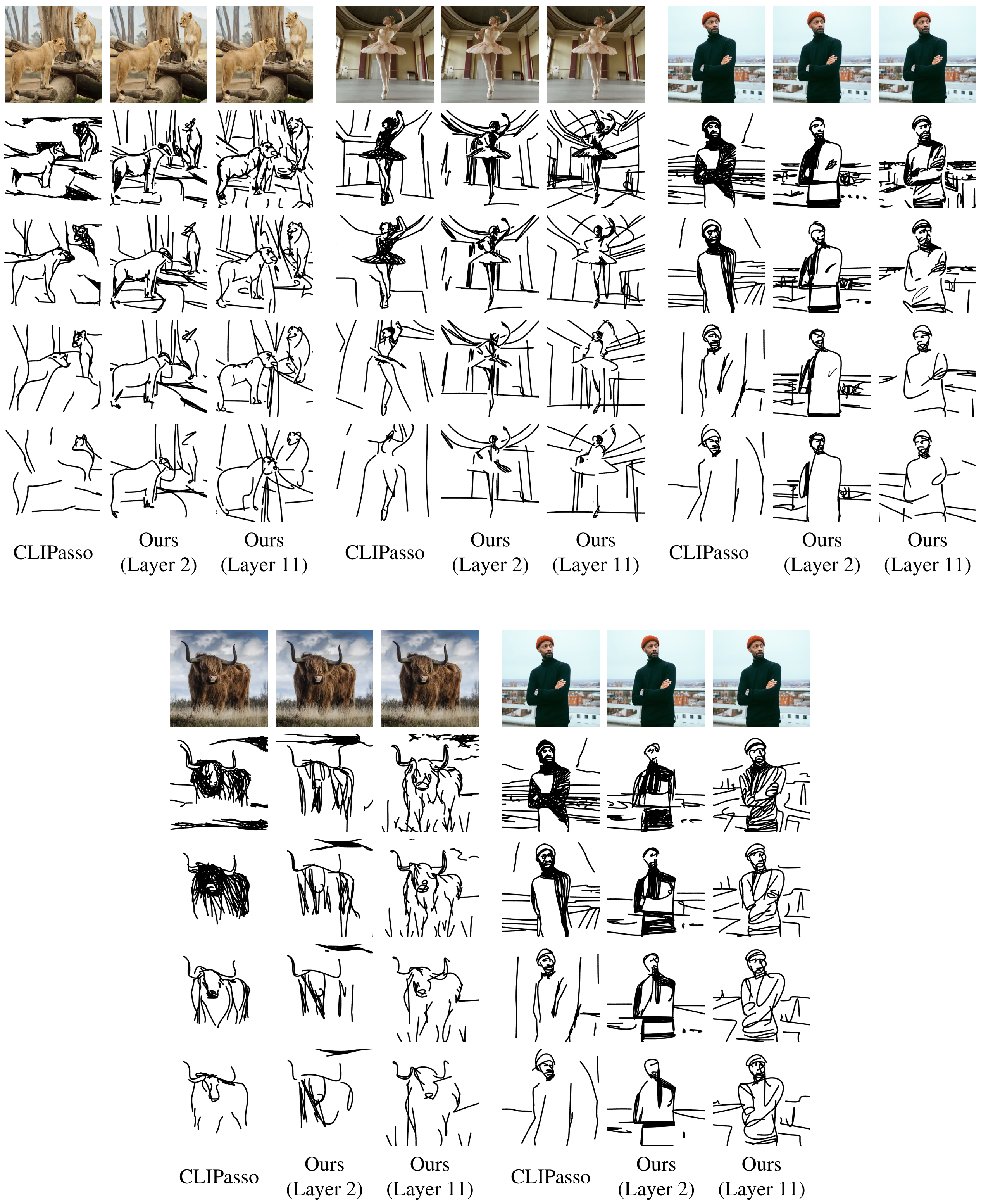

Figure 26. Comparison to CLIPasso. For CLIPasso, we generate the sketches using 128, 64, 32, and 16 strokes. In the top set of results, we use our scene decomposition technique and apply our implicit simplification starting with 64 strokes for the foreground and background sketches. For the bottom set of results, we do not use the scene decomposition approach and start with simplification using 128 strokes. We show our results for layers 2 and 11.

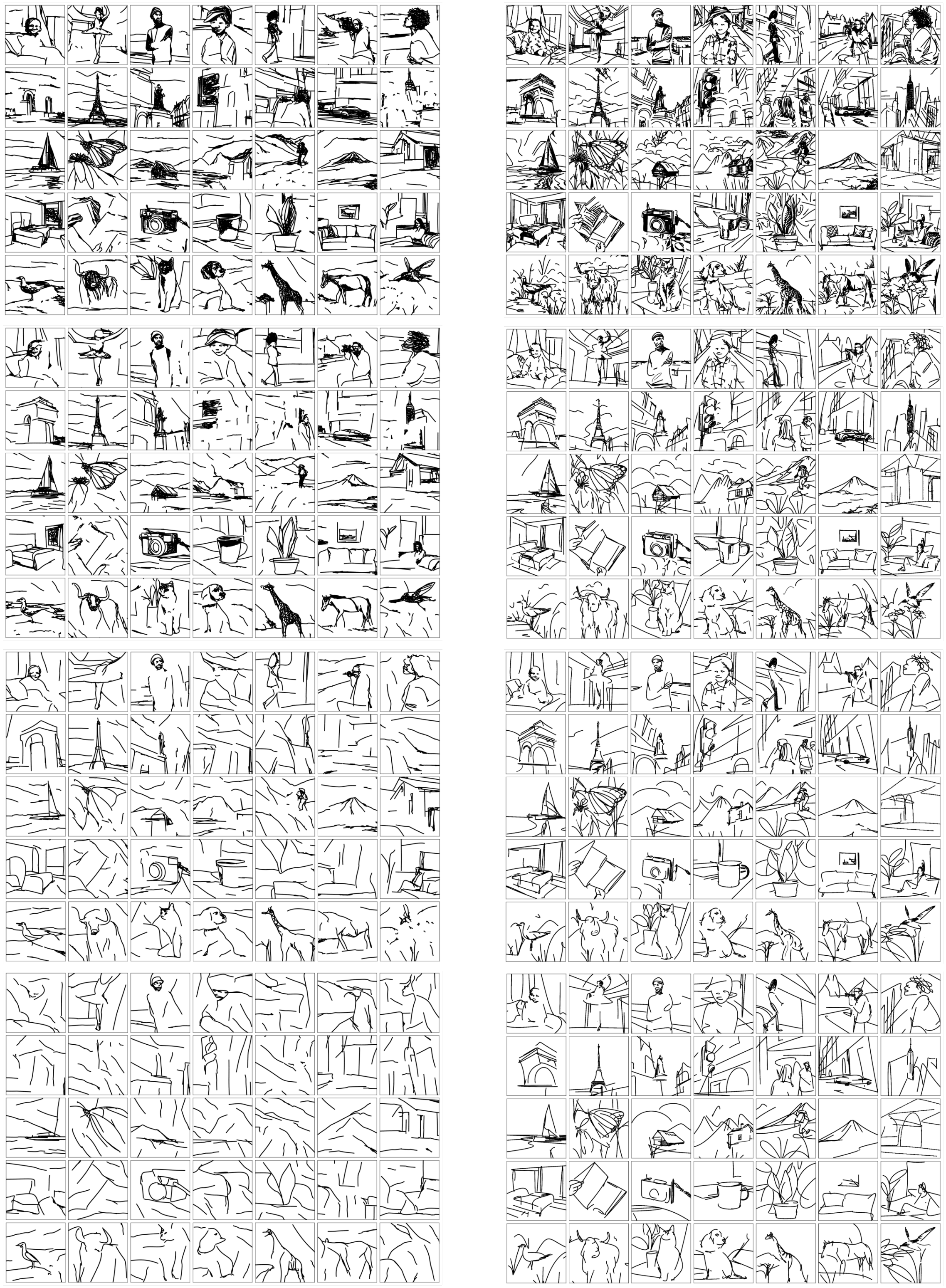
Figure 27. The 35 sketches produced for the quantitative experiment. On the left are the results of CLIPasso with four levels of abstraction, and on the right are our results with four levels of abstraction obtained using layer 11 of CLIP-ViT.

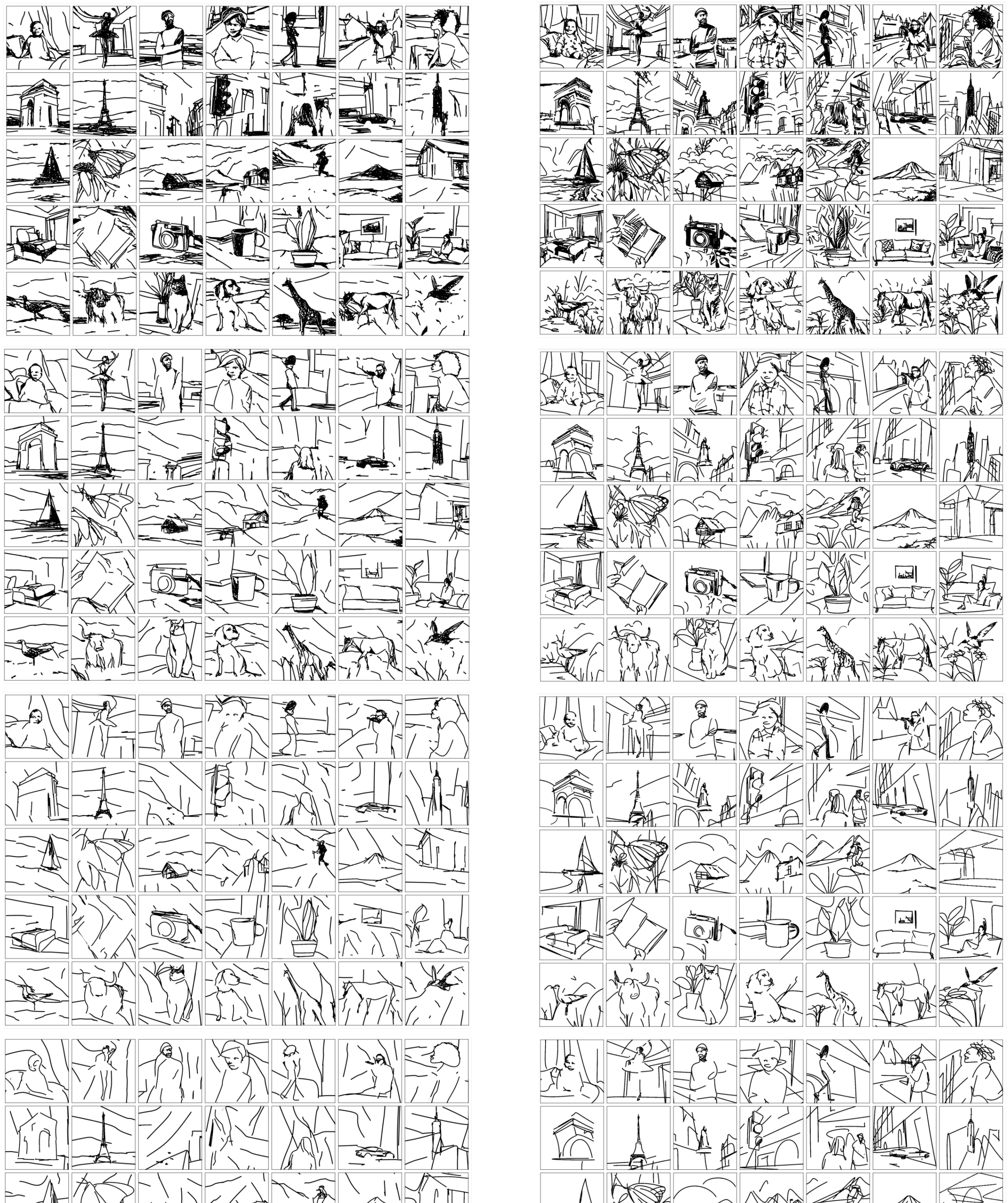

Figure 28. A comparison to CLIPasso with the scene separation technique on the 35 images used for the quantitative experiment. On the left are the results of CLIPasso with four levels of abstraction, when we separate the image into foreground and background, and sketch each of the separately. On the right are our results with four levels of abstraction obtained using layer 11 of CLIP-ViT.

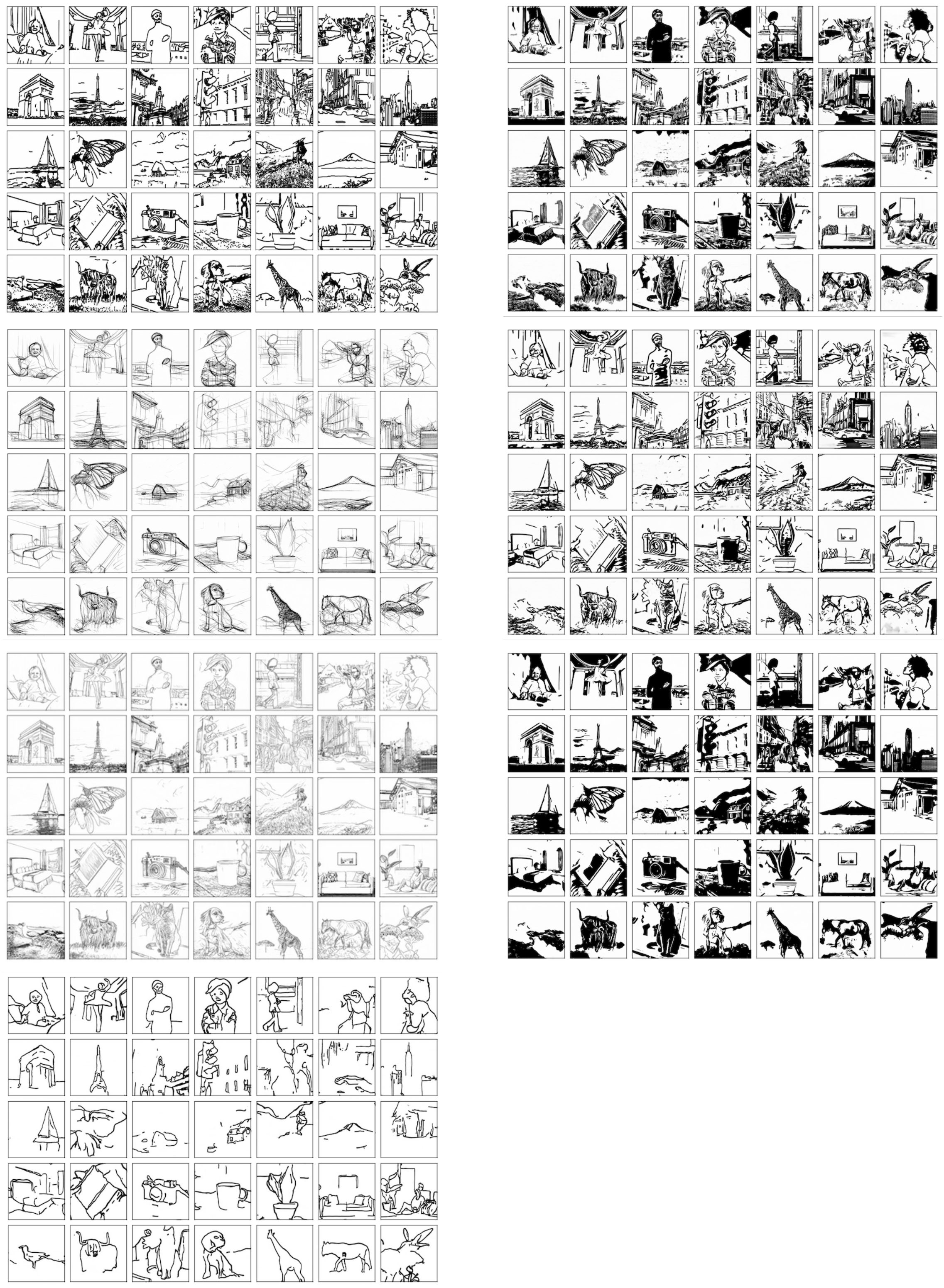

Figure 29. The 35 sketches produced for the quantitative experiment. On the left are the results by Chan *et al*. [2] with the three provided styles, the last row on the left is by Photo-Sketch [7]. On the right are the results by UPDG [17] with the three provided styles.

References

[1] AUTOMATIC1111. Stable diffusion webui. https://github.com/AUTOMATIC1111/stable-diffusion-webui, 2022. 19

[2] Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7915–7925, 2022. 20, 21, 22, 26

[3] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *CoRR*, abs/1711.02257, 2017. 2

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 2

[5] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022. 19

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 19

[7] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1403–1412. IEEE, 2019. 26

[8] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 19

[9] Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R Zaiane, and Martin Jagersand. U2-net: Going deeper with nested u-structure for salient object detection. *Pattern recognition*, 106:107404, 2020. 1

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 2

[11] 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. 19

[12] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 19

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 19

[14] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022. 19

[15] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2149–2159, 2022. 1

[16] Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. Clipasso: Semantically-aware object sketching. *ACM Trans. Graph.*, 41(4), jul 2022. 2, 6, 19

[17] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Unpaired portrait drawing generation via asymmetric cycle mapping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8217–8225, 2020. 20, 22, 26

[18] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022. 19