

CLIPascene: Scene Sketching with Different Types and Levels of Abstraction

Yael Vinker
Tel Aviv University

Yuval Alaluf
Tel Aviv University

Daniel Cohen-Or
Tel Aviv University

Ariel Shamir
Reichman University

<https://clipascene.github.io/CLIPascene/>

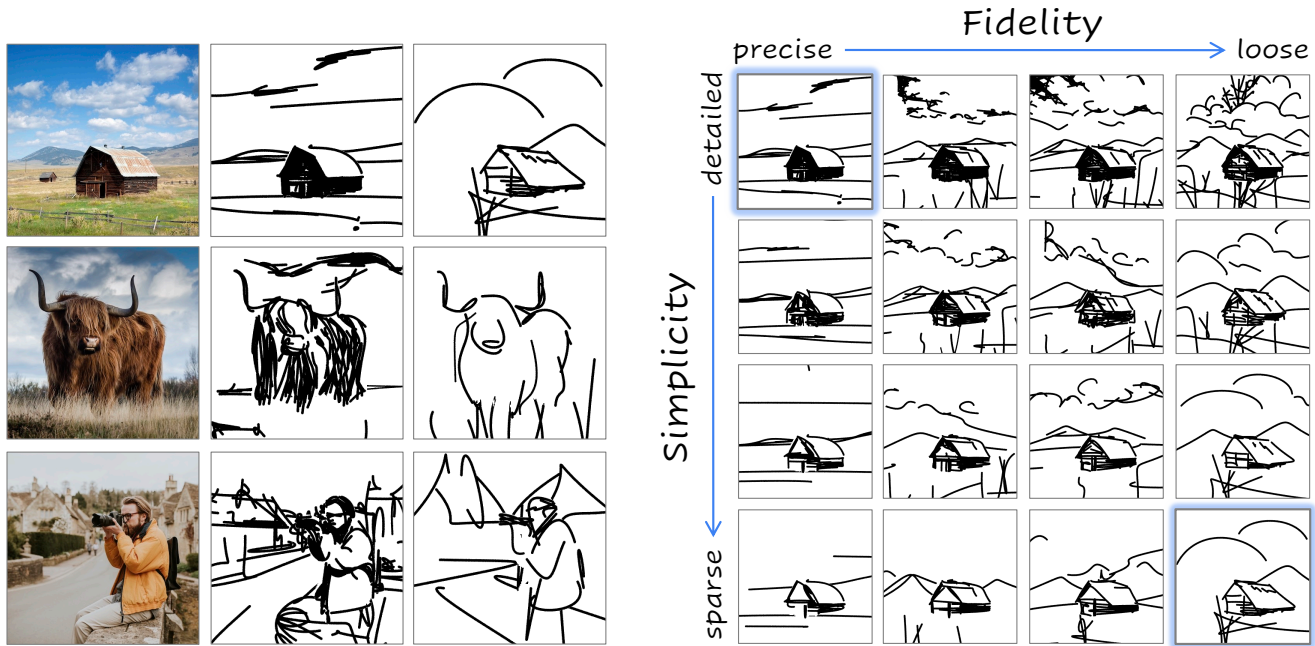


Figure 1. Our method converts a *scene* image into a sketch with different types and levels of abstraction by disentangling abstraction into two axes of control: *fidelity* and *simplicity*. For each image on the left, we illustrate two example sketches generated by our method. The first sketch is more precise and detailed, accurately depicting the scene’s structure and geometry. The second is more abstract, using fewer strokes and relying more on the semantics of the image instead of on the scene’s exact geometry. These sketches were selected from a complete *matrix* that was generated by our method (shown on the right), encompassing a broad range of possible sketch abstractions for a given image. Users can select the desired sketch based on their goals and personal taste.

Abstract

In this paper, we present a method for converting a given scene image into a sketch using different types and multiple levels of abstraction. We distinguish between two types of abstraction. The first considers the fidelity of the sketch, varying its representation from a more precise portrayal of the input to a looser depiction. The second is defined by the visual simplicity of the sketch, moving from a detailed depiction to a sparse sketch. Using an explicit disentanglement into two abstraction axes — and multiple levels for each one — provides users additional control over selecting the desired sketch based on their personal goals and preferences. To form a sketch at a given level of fidelity and simplification, we train two MLP networks. The first network learns the desired placement of strokes, while the second network learns to gradually remove strokes from the sketch

without harming its recognizability and semantics. Our approach is able to generate sketches of complex scenes including those with complex backgrounds (e.g. natural and urban settings) and subjects (e.g. animals and people) while depicting gradual abstractions of the input scene in terms of fidelity and simplicity.

1. Introduction

Several studies have demonstrated that abstract, minimal representations are not only visually pleasing but also helpful in conveying an idea more effectively by emphasizing the essence of the subject [4, 16]. In this paper, we concentrate on converting photographs of natural scenes to sketches as a prominent minimal representation.

Converting a photograph to a sketch involves abstraction,



Figure 2. Levels of scene complexity: (A) contains a single, central object with a simple background, (B) contains multiple objects (the cat and vase) with a slightly more complicated background (the table and photos), and (C) contains both foreground and background that include many details. Our work tackles all types of scenes.



Figure 3. Drawings of different scenes created by different artists. Notice the significant differences in style and levels of abstraction between each drawing — moving from more detailed and precise to more abstract from left to right. In the second row, we demonstrate how the level of abstraction not only varies *between* drawings, but also *within* the *same* drawing. In each pair, we highlight less abstract (in red) and more abstract (in blue) regions of the drawing. For example, in the middle drawing the person highlighted in red is drawn with a high level of detail as this is the object that the artist chose to emphasize.

which requires the ability to understand, analyze, and interpret the complexity of the visual scene. A scene is composed of multiple objects, relations between the foreground and background, and an inherent hierarchy between them (see Figure 2). Therefore, when sketching a scene, the artist has many options regarding how to express the various components and the relations between them (see Figure 3).

In a similar manner, computational sketching methods must deal with scene complexity and consider a variety of abstraction levels. Our work focuses on the challenging task of *scene* sketching while doing so using multiple *types* and multiple *levels* of abstraction.

Only a few previous works attempted to produce sketches with multiple levels of abstraction. However, these works focus specifically on the task of *object* sketching [29, 44] or *portrait* sketching [2], and often simply use the number of strokes to define the level of abstraction. We are not aware of any previous work that attempts to separate different *types* of abstractions.

We define two axes that represent two different types of abstractions, and produce gradual levels of abstraction by

moving along these axes. The first axis governs the *fidelity* of the sketch. This axis moves from more precise sketches, where the sketch composition follows the geometry and structure of the photograph to more loose sketches, where the composition relies more on the semantics of the scene. We illustrate this axis in Figure 4. The leftmost sketch is most faithful to the geometry of the input image, and as we move right, the sketch becomes less precise, deviating from the contours of the input and relying more on key semantic aspects of the input image. Observe the flowers in the rightmost sketch are added to the front, yet these flowers deviate significantly from the *specific* flowers present in the input.



Figure 4. The *fidelity* axis. From left to right, using the same number of strokes the sketches gradually depart from the geometry of the input image, but still convey the semantic meaning of the scene. The first sketch follows the exact contours of the mountains on the horizon while the mountains and flowers in the rightmost sketch are a more abstract representation of the scene’s semantics.

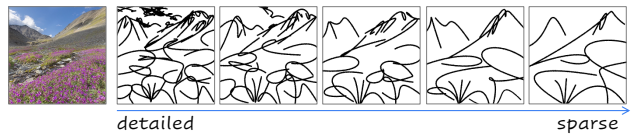


Figure 5. The *simplification* axis. On the left, we start with a more detailed sketch and as we move to the right the sketch is gradually simplified while still preserving recognizability.

The second axis governs the level of detail the sketch contains and moves from more detailed depictions to sparser depictions, which appear more abstract. Hence, we refer to this axis as the *simplicity* axis. An example can be seen in Figure 5, where the same general characteristics of the scene (*e.g.* the mountains and flowers) are captured in all sketches, but with gradually fewer details.

To deal with scene complexity, we separate the foreground and background elements in the scene and sketch each of them separately. This explicit separation and the disentanglement into two abstraction axes provide a more flexible framework for computational sketching. Users can choose the desired sketch along the two abstraction axes, and apply different abstractions to the foreground and background elements according to their goals and personal taste from a range of possibilities, as illustrated in Figure 1.

We define a sketch as a set of Bézier curves, and train a simple multi-layer perceptron (MLP) network to map an initial set of stroke parameters to their final locations forming the final sketch. The training process is done per im-

age (e.g. without an external dataset) and is guided by a pre-trained CLIP-ViT [9, 34] model, leveraging its powerful ability to capture the semantics and global context of the entire scene.

To realize the *fidelity* axis, we utilize different intermediate layers of CLIP-ViT to guide the training process, where shallow layers preserve the geometry of the image and deeper layers encourage the creation of looser sketches that rely more on the semantics of the scene. To realize the *simplicity* axis, we jointly train an additional MLP network that learns how to best discard strokes without harming the recognizability of the sketch. As shall be discussed, the use of the networks over a direct optimization-based approach is crucial for achieving a learned simplification of the sketch. To alter the level of simplicity, we introduce two loss functions that are used to balance the faithfulness and sparsity of the sketch with respect to the input. We gradually alter the relative strength of these two functions in an exponential fashion to define a perceptually smooth sequence of simplifications. This choice of an exponential relation also aligns well with the Weber-Fechner law [10, 46] which notes that relations between a perceived change and an actual change in a physical stimulus in many cases are non-linear.

The resulting sketches demonstrate our ability to cope with various scenes and to capture their core characteristics, while providing gradual abstraction along both the fidelity and simplicity axes. We compare our results with existing methods for semantically-aware scene sketching. We additionally evaluate our results quantitatively and demonstrate that the generated sketches, although abstract, successfully preserve the geometry and semantics of the input scene.

2. Related Work

Free-hand sketch generation differs from edge-map extraction methods [5, 47] in that it attempts to produce sketches that are representative of the style of human drawing to some extent. Yet, there are significant differences in drawing styles among individuals depending on their goals, skill levels, and more, see Figure 3. As such, computation sketching methods, which aim to mimic human drawing, must consider a wide range of sketch representations.

This ranges from methods aiming to produce sketches that are grounded in the edge map of the input image [21, 24, 43, 48], to those that aim to produce sketches that are more abstract [3, 11, 13, 14, 31, 44]. Several works have attempted to develop a unified algorithm that can output sketches with a variety of styles [6, 26, 51]. There are, however, only a few works that attempt to provide various levels of abstraction [2, 29, 44]. In the following we elaborate on the relevant previous methods, focusing on scene-sketching approaches. We refer the reader to [49] for a comprehensive survey on computation sketching techniques.

2.1. Photo-Sketch Synthesis

Various works formulate the sketch generation task as an image-to-image translation task using paired data of images and the corresponding sketches [21, 23, 27, 50]. Others approach the translation task via unpaired data, often relying on a cycle consistency constraint [6, 41, 51]. Li et al. [21] introduce a GAN-based contour generation algorithm and utilize multiple ground truth sketches to guide training to account for diversity present in human sketches. Yi et al. [51] generate portrait drawings with unpaired data via an adversarial training scheme by employing a cycle-consistency objective and a discriminator trained to learn a specific style. As these works rely on curated datasets, they require training a new model for each desired style while supporting a single level of sketch abstraction. A different approach for image-sketch synthesis formulates the sketching task as a multi-agent referential game in which two reinforcement learning agents must communicate visual concepts to each other through sketches [28, 33, 53].

Recently, Chan et al. [6] propose an unpaired GAN-based approach. They train a generator to map a given image into a sketch with multiple styles defined explicitly from four existing sketch datasets with a dedicated model trained for each desired style. They utilize a CLIP-based loss between the sketch and input image to achieve semantically-aware sketches. However, their sketches tend to be more geometric in nature, compared to our method which can generate sketches at multiple levels of abstraction, including those with both a precise and loose depiction of the input.

Importantly, compared to existing work, our approach does not rely on any explicit dataset and is not limited to a pre-defined set of styles. Instead, we leverage the powerful semantics captured by a pre-trained CLIP model [34]. Additionally, our work is the only one among the alternative scene sketching approaches that generates sketches in vector form rather than pixels. This representation is much more natural for sketches and allows the user to modify the style of strokes as a post-process.

2.2. Sketch Abstraction

The process of sketching naturally requires the artist to perform abstraction due to the simple tools used for sketching. Existing methods for computational sketching implicitly perform abstraction as well, fitting the sketches to a single pre-defined style. While abstractions are fundamental to sketches, only a few works have attempted to create sketches at multiple levels of abstraction, while no previous works have attempted to do so over an entire scene. Berger et al. [2] utilize sketches collected from various artists to learn a mapping from a face photograph to a portrait sketch at various levels of abstraction. Their method, however, requires a new dataset for each desired level of abstraction or artistic style and is only suited for faces. Muhammad

et al. [29] approach sketch abstraction with a reinforcement learning agent trained to remove strokes from an edge map without harming the sketch’s recognizability. To train the agent, an external sketch classifier is trained on nine classes from the QuickDraw dataset [14].

2.3. CLIPasso

Finally, most similar to our work is CLIPasso [44] which was designed for *object* sketching at multiple levels of abstraction. They define a sketch as a set of Bézier curves and optimize the stroke parameters with respect to a CLIP-based [34] similarity loss between the input image and generated sketch. Multiple levels of abstraction are realized by reducing the number of strokes used to compose the sketch.

Similar to CLIPasso, we use Bézier curves to represent our sketches and employ a pre-trained CLIP model to guide the sketching process. In contrast to CLIPasso, however, our method is not restricted to objects and can handle the challenging task of *scene* sketching. Additionally, while Vinker et al. examine only a single form of abstraction, we disentangle abstraction into two distinct axes controlling both the simplicity and the faithfulness of the sketch.

Moreover, CLIPasso defines abstraction *explicitly* via the number of strokes used to draw the sketch, and uses optimization-based approach to learn the parameters of the strokes. However, sketching different inputs requires a different number of strokes, and knowing the number of strokes in advance is often challenging. In contrast, we learn the desired number of strokes *implicitly* by training two MLP networks to achieve a desired trade-off between simplicity and fidelity with respect to the input image.

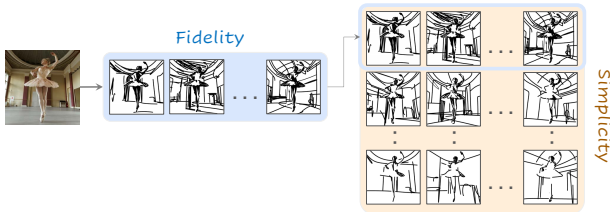


Figure 6. High-level overview of our method. We begin by forming the top row of the fidelity axis. Then, we gradually simplify each sketch in this row to create the columns of the simplicity axis.

3. Method

Given an input image \mathcal{I} of a scene, our goal is to produce a set of corresponding sketches at different levels of *fidelity* and *simplicity*. Conceptually this can be thought of as forming a 2-dimensional space of sketch abstractions, where each axis represents one form of abstraction. Sampling this space along the two abstraction axes produces a 2-dimensional abstraction matrix of size $m \times n$.

An outline of our scene sketching process is presented in Figure 6. We begin by producing a set of sketches at multiple levels of abstraction along the *fidelity* axis (Sections 3.1 and 3.2), with no simplification, thus forming the top row (highlighted in blue) in the abstractions matrix. Next, for each sketch at a given level of fidelity, we perform an iterative visual simplification (highlighted in orange) by learning how to best remove select strokes and adjust the locations of the remaining strokes (Section 3.3). This process results in an abstraction matrix representing various abstractions of the complete scene.

In the following, our method is described in detail, taking into account the entire scene as a whole. However, to allow for greater control over the appearance of the output sketches, our final scheme splits the image into two regions (the salient foreground object and the background). We apply our 2-axes abstraction method to each region separately, allowing even more flexibility in combining them to form the matrix of sketches (Section 3.4).

3.1. Training Scheme

We define a sketch as a set of n black strokes placed over a white background, where each stroke is a two-dimensional Bézier curve with four control points. We mark the i -th stroke by its set of control points $z_i = \{(x_i, y_i)^j\}_{j=1}^4$, and denote the set of the n strokes by $Z = \{z_i\}_{i=1}^n$. Our goal is to find the set of stroke parameters that produces a sketch adequately depicting the input scene image.

An overview of our training scheme used to produce a single sketch image is presented in the gray area of Figure 7. We train an MLP network, denoted by MLP_{loc} , that receives an initial set of control points $Z_{init} \in \mathbb{R}^{n \times 4 \times 2}$ (marked in blue) and returns a vector of offsets $MLP_{loc}(Z_{init}) = \Delta Z \in \mathbb{R}^{n \times 4 \times 2}$ with respect to the initial stroke locations. The final set of control points are then given by $Z = Z_{init} + \Delta Z$, which are then passed to a differentiable rasterizer \mathcal{R} [22] that outputs the rasterized sketch,

$$\mathcal{S} = \mathcal{R}(Z_{init} + \Delta Z). \quad (1)$$

To guide the training process, we leverage a pre-trained CLIP model [34] due to its capabilities of encoding shared information from both sketches and natural images. As opposed to Vinker et al. [44] that use the ResNet-based [15] CLIP model for the sketching process (and struggles with depicting a scene image), we find that the ViT-based [9] CLIP model is more suitable for capturing the global context required for generating a coherent sketch of a whole scene. This also follows the observation of Raghu et al. [35] that ViT models better capture more global information at lower layers compared to ResNet-based models. We further analyze this design choice in the appendix.

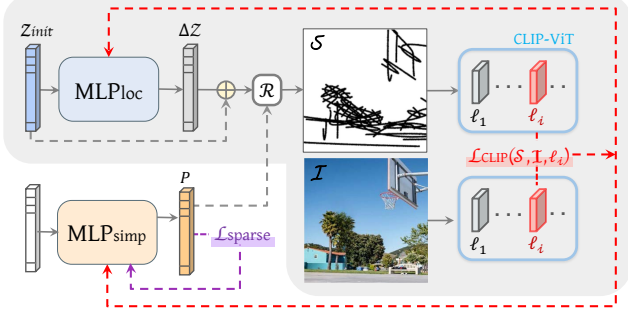


Figure 7. Single sketch generation scheme. In gray, we show our training scheme used to produce a single sketch image at a single level of our fidelity abstraction. In the bottom left we show our sketch simplification scheme used to gradually simplify the generated sketches.

The sketch \mathcal{S} and the input image \mathcal{I} are fed into a pre-trained ViT-CLIP model. The loss function is then defined as the L2 distance between the corresponding activations of CLIP at a chosen layer ℓ_k :

$$\mathcal{L}_{CLIP}(\mathcal{S}, \mathcal{I}, \ell_k) = \|\text{CLIP}_{\ell_k}(\mathcal{S}) - \text{CLIP}_{\ell_k}(\mathcal{I})\|_2^2, \quad (2)$$

At each step during training, we back-propagate the loss through the CLIP model and the differentiable rasterizer \mathcal{R} whose weights are frozen, and only update the weights of MLP_{loc} . This process is repeated iteratively until convergence. Observe that no external dataset is needed for guiding the training process. Instead, we rely solely on the expressiveness and semantics captured by the pre-trained CLIP model. This training scheme produces a *single* sketch image at a *single* level of fidelity and simplicity. Below, we describe how to control these two axes of abstraction.

3.2. Fidelity Axis

To achieve different levels of fidelity, as illustrated by a single row in our abstraction matrix, we select different activation layers of the ViT-CLIP model for computing the loss defined in Equation (2). Specifically, in all our examples we train a separate MLP_{loc} using layers $\{\ell_2, \ell_7, \ell_8, \ell_{11}\}$ of ViT-CLIP. Optimizing via deeper layers leads to sketches that are more semantic in nature and do not necessarily confine to the precise geometry of the input. Note that it is also possible to use the remaining layers to achieve additional fidelity levels (see the appendix).

3.3. Simplification Axis

The procedure described above generates a single row in the abstraction matrix. The second axis allows altering the level of *simplicity* of the sketch to convey the same concept with varying amount of explicit information.

Given a sketch \mathcal{S}_k at fidelity level k , our goal is to find a set of sketches $\{\mathcal{S}_k^1, \dots, \mathcal{S}_k^m\}$ that are visually and concep-

tually similar to \mathcal{S}_k but have a gradually simplified appearance. In practice, we would like to learn how to best remove select strokes from a given sketch and refine the locations of the remaining strokes without harming the overall recognizability of the sketch.

We illustrate our sketch simplification scheme for generating a single simplified sketch \mathcal{S}_k^j in the bottom left region of Figure 7. We train an additional MLP network, denoted as MLP_{simp} , that receives a constant valued vector and is tasked with learning an n -dimensional vector $P = \{p_i\}_{i=1}^n$, $p_i \in [0, 1]$ (marked in orange). Here, p_i represents the probability of the i -th stroke appearing in the rendered simplified sketch. P is passed as an additional input to \mathcal{R} which outputs the simplified sketch \mathcal{S}_k^j in accordance. One may view p_i as the importance of stroke i to the sketch where a value of 0 indicates that this stroke can be discarded and a value of 1 indicates that this stroke should be kept.

In practice, to implement the probabilistic-based removal or addition of strokes into the rendering process, we multiply the width parameter of each stroke z_i with p_i . When rendering the sketch, strokes with a very low probability will be “hidden” due to their very small width. Note that using the MLP network rather than performing a direct optimization over the stroke parameters (as is done in Vinker *et al.*) is crucial as it allows us to restore strokes that have been previously removed. If we were to use direct optimization, the gradients of deleted strokes remain as such since they were multiplied by a probability of 0.

To encourage a sparse representation of the sketch (*i.e.* one with fewer strokes) we minimize the normalized L1 norm of P :

$$\mathcal{L}_{sparse}(P) = \frac{\|P\|_1}{n}. \quad (3)$$

Observe that a naïve solution minimizing \mathcal{L}_{sparse} would simply discard all strokes. Therefore, when training MLP_{simp} we additionally compute the \mathcal{L}_{CLIP} loss presented in Equation (2) to ensure that the resulting sketch still resembles the original input image. During the training of MLP_{simp} , we also continue to fine-tune MLP_{loc} .

Formally, we minimize the sum:

$$\mathcal{L}_{CLIP}(\mathcal{S}_k^j, \mathcal{I}, \ell_k) + \mathcal{L}_{sparse}(P). \quad (4)$$

We back-propagate the gradients from \mathcal{L}_{CLIP} to both MLP_{loc} and MLP_{simp} while \mathcal{L}_{sparse} is used only for training MLP_{simp} (as indicated by the red and purple dashed arrows in Figure 7).

Balancing the Losses. Naturally, the balance between the two losses \mathcal{L}_{CLIP} and \mathcal{L}_{sparse} affects the appearance of the simplified sketch. Specifically, if \mathcal{L}_{sparse} incurs a low value, we attain a sketch with a lower number of strokes (*i.e.* more abstract). This in turn results in a high value



Figure 8. Trade-off between \mathcal{L}_{sparse} and \mathcal{L}_{CLIP} . As the sketch becomes more abstract, the number of strokes decreases, resulting in a lower score for \mathcal{L}_{sparse} . However, the sketch also becomes less recognizable with respect to the input image, resulting in a higher penalty for \mathcal{L}_{CLIP} .

for \mathcal{L}_{CLIP} since the sketch is less recognizable (see Figure 8). Therefore, the balance between \mathcal{L}_{sparse} and \mathcal{L}_{CLIP} is essential for achieving recognizable sketches with varying degrees of abstraction. Thus, we define the following loss function:

$$\mathcal{L}_{ratio} = \left\| \frac{\mathcal{L}_{sparse}}{\mathcal{L}_{CLIP}} - r \right\|_2^2, \quad (5)$$

where the scalar factor r (denoting the *ratio* of the two losses) controls the strength of simplification. As we decrease r , we encourage the network to output a sparser sketch and vice-versa.

The final objective for performing a visual simplification of a given sketch \mathcal{S}_k , at a single simplification abstraction level, is then given by:

$$\mathcal{L}_{simp} = \mathcal{L}_{CLIP} + \mathcal{L}_{sparse} + \mathcal{L}_{ratio}. \quad (6)$$

Defining the Visual Simplification Factor r To achieve the set of gradually simplified sketches $\{\mathcal{S}_k^1, \dots, \mathcal{S}_k^m\}$, we define a set of corresponding factors $\{r_k^1, \dots, r_k^m\}$ to be applied in Equation (5).

We begin by determining the value of the first ratio r_k^1 for each k . As we already generated the first row in the abstraction matrix (Section 3.2), a natural choice for r_k^1 is one that would reproduce the strength of simplification present in \mathcal{S}_k . As such, we can consider the factor that will lead to a value of \mathcal{L}_{sparse} equal to 1 (*i.e.* no visual simplification is performed). Following Equation (5), we can then define the first factor as:

$$r_k^1 = \frac{1}{\mathcal{L}_{CLIP}(\mathcal{S}_k, \mathcal{I}_k)}. \quad (7)$$

The derivation of the remaining factors r_k^j is described next.

Perceptually Smooth Simplification As introduced above, the set of factors determines the strength of the visual simplification. When defining this set, we aim to achieve a *smooth* simplification. By *smooth* we mean that there is no large jump perceptually between two consecutive steps. We find that this is achieved when \mathcal{L}_{sparse} is exponential with respect to \mathcal{L}_{CLIP} . This is illustrated in Figure 9, where the first row provides an example of a

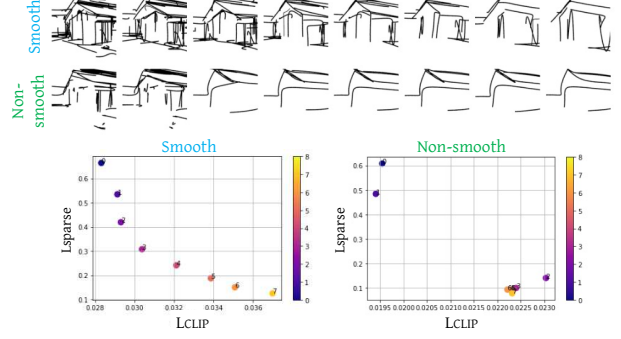


Figure 9. Simplification results for layers 8 (top) and 2 (bottom) with corresponding graphs of \mathcal{L}_{sparse} as a function of \mathcal{L}_{CLIP} . In the first row, the simplification appears perceptually smooth, with each step depicting a simplified version of the previous sketch using a consistent “jump” in the degree of abstraction. The bottom row demonstrates a “non-smooth” simplification, as there is a visible “jump” between the second and third sketches with the abstraction plateauing thereafter.

smooth transition from \mathcal{S}_8^1 to \mathcal{S}_8^m , while the second row demonstrates a non-smooth transition, where there is a large perceptual “jump” in the abstraction level between the second and third sketches, and almost no perceptual change in the following levels.

The two graphs at the bottom of Figure 9 describe this observation quantitatively, illustrating the trade-off between \mathcal{L}_{sparse} and \mathcal{L}_{CLIP} . We can see how the smooth transition between the sketches in the first row forms an exponential relation between \mathcal{L}_{sparse} and \mathcal{L}_{CLIP} , while the large “jump” in the second row is clearly shown in the gap in the right graph. This observation aligns well with the Weber-Fechner law [10, 46] which states that human perception is linear with respect to an exponentially-changing signal. In our case, perception is “measured” by the CLIP similarity between the sketch and the input image, and the abstraction signal is attained by \mathcal{L}_{sparse} which is simply a function of the number of strokes.

Given this, we define an exponential function recursively by $f(j) = f(j-1)/2$. The initial value of the function is defined differently for each fidelity level k as $f_k(1) = r_k^1$. To create the following $m-1$ simplification levels we sample the function f_k to define the set of factors $\{r_k^2, \dots, r_k^m\}$. For each k , the sampling step size is set proportional to the strength of the \mathcal{L}_{CLIP} loss incurred at level k . Hence, layers that incur a large \mathcal{L}_{CLIP} value are sampled with a larger step size. We found this procedure achieves simplifications that are perceptually smooth. An analysis of the ratios and our design choices are provided in the appendix.

Generating the Simplified Sketches To generate the set of simplified sketches $\{\mathcal{S}_k^1, \dots, \mathcal{S}_k^m\}$, we use the set of m corresponding ratios $\{r_k^1, \dots, r_k^m\}$ and apply the training procedure

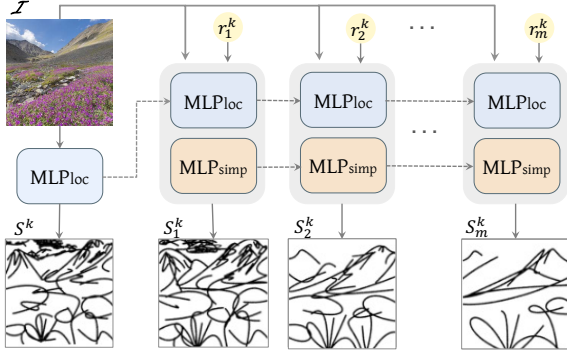


Figure 10. Iterative simplification of the sketch S_k . Given the image \mathcal{I} and the trained MLP_{loc} used to generate S_k (shown on the left), we start with training MLP_{simp} (marked in orange) and fine-tune MLP_{loc} to output the first sketch S_k^1 with a similar simplification level as observed in S_k . We then iteratively fine-tune these networks simultaneously to fit the new \mathcal{L}_{simp} loss defined by each r_k^j and generate the next simplified sketch S_k^j .



Figure 11. Preprocessing input and output examples.

iteratively. This process is illustrated in Figure 10: to generate S_k^1 we start with the trained MLP_{loc} used to generate S_k (marked in blue) and a randomly-initialized MLP_{simp} network (marked in orange). We then use r_k^1 to define the first \mathcal{L}_{ratio} loss and generate the first simplified sketch S_k^1 . After generating S_k^1 , we sequentially generate each S_k^j for $2 \leq j \leq m$ by continuing training both networks for a fixed number of steps and applying \mathcal{L}_{ratio} with the corresponding target ratio r_k^j .

3.4. Decomposing the Scene

While sketching the entire scene together works well across different scenes, we found that doing so may lead to degraded performance in various cases. First, weakening the faithfulness of the sketch with respect to the input image may over-exaggerate features of the subject (see Figure 12, first row). In other cases, when moving along the simplification abstraction axis, the scene’s subject may “disappear” into the background (see Figure 12, third row).

To avoid such artifacts and provide additional control over the appearance of the final sketches, we separate the scene’s foreground subject from background, and sketch each of them independently. We do so by extracting the scene’s salient object(s) using a pretrained U²-Net salient object detector [32]. We then apply a pretrained LaMa [42] inpainting model to recover the missing regions. This process is illustrated in Figure 11.

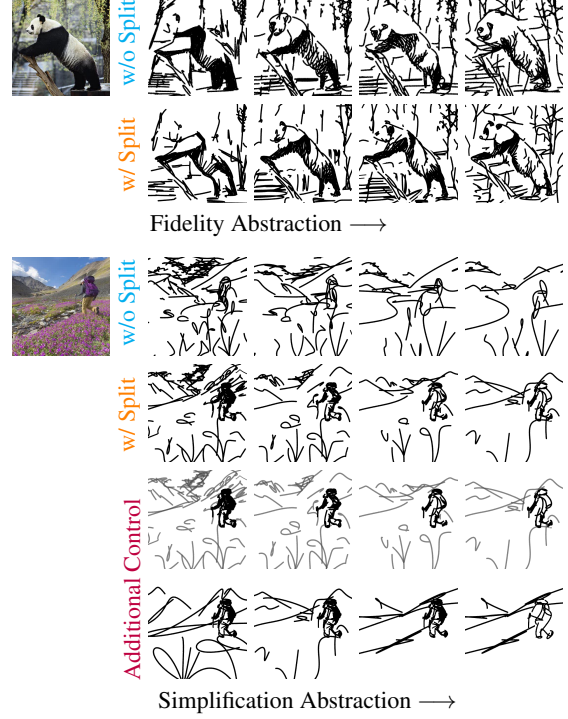


Figure 12. Scene sketching results obtained with and without decomposing the scene. In the first two rows of each segment we show abstractions obtained with and without the scene-splitting technique. The last two rows show example of varying background opacity, and combination of foreground and background sketches at different abstraction levels.

Having split the scene into two independent components, we compose the abstraction matrix for each using the same technique as described above. When performing object sketching, we additionally compute \mathcal{L}_{CLIP} over layer l_4 . We find doing so assists in preserving the input geometry and finer details, and helps in avoiding artefacts as seen in the first row of Figure 12.

In addition, independently sketching the scene’s foreground and background provides users with more control. First, users can choose how to balance the emphasis placed on the scene subject by altering the opacity of the background sketches. Second, users may combine the foreground and background sketches at different levels of abstraction. Examples are provided in Figure 12.

4. Results

In the following sections, we demonstrate the performance of our scene sketching technique qualitatively and quantitatively. We additionally provide visual comparisons to state-of-the-art sketching methods. All results presented in this section were generated using the object-background separation approach. Further analysis, comparisons, and results are provided in the appendix.



Figure 13. Examples results created by our method. For each input image, we present two representative sketches. The first sketch illustrates a more precise depiction of the input with a high level of detail. The second sketch demonstrates a more abstract and semantic depiction realized with fewer strokes that deviate from the precise structure of the input. Additional results and full abstraction matrices are presented in the appendix.

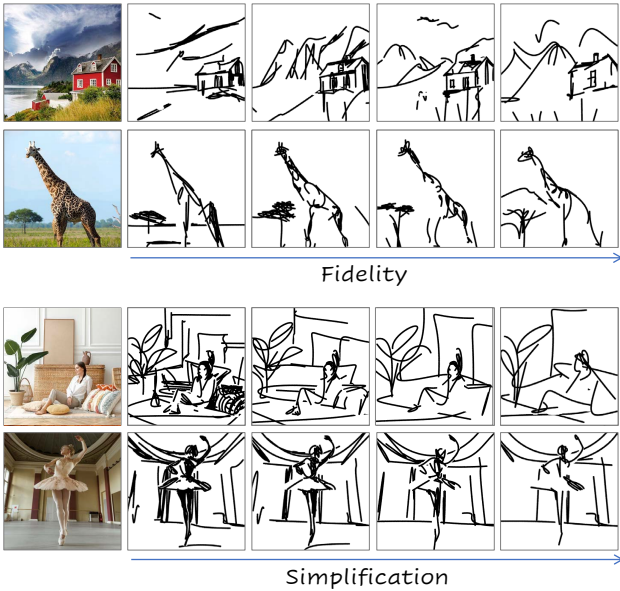


Figure 14. Examples of smooth levels of abstraction along our abstraction axes.

4.1. Qualitative Evaluation

In Figures 1 and 13 we show sketches at different levels of abstraction on various scenes generated by our method. For each image, we choose two representative sketches (one detailed and one more abstract) from the abstraction matrix to demonstrate the ability of our method to produce diverse levels of abstraction. Notice how the sketches are still easy to recognize as depicting the same scene even though they vary significantly in their abstraction level.

In the top of Figure 14 we show sketch abstractions along the *fidelity* axis. Sketches become less precise as we move from left to right, while still conveying the semantics of the images. For example, in the leftmost sketch of the house and the giraffe, the strokes emphasize areas with strong edges, such as the dark sky above the house and the shadow

on the giraffe’s legs. As we move to the right, the mountains behind the house and the shape of the tree behind the giraffe and the giraffe’s body become more loose. Importantly, even though the composition of the sketches becomes less precise, the sketches still capture the essence of the scene.

At the bottom of Figure 14 we show sketch abstractions along the *simplicity* axis. Our method successfully simplifies the sketches in a smooth fashion, while still capturing the core characteristics of the scene. For example, notice how the unique pose of the ballerina is preserved across the images, even when sketched using a small number of strokes. Observe that this simplification is achieved *implicitly* through our iterative sketch simplification technique.

4.2. Comparison with Existing Methods

In Figure 15 we present a comparison with three state-of-the-art methods for scene sketching [6, 21, 51]. As a simple baseline approach, we additionally apply an edge detector filter over the input image using the Sobel operator [18] followed by the sketch simplification model from Simo-Serra *et al.* [40] (referred as “Edge Extraction”). On the right, we present three sketches produced by our method depicting three representative levels of abstraction.

The sketches produced by UPDG [51] and Chan *et al.* [6] are detailed, closely following the contours of the input images (e.g., such as the buildings in row 2). The sketches produced by these methods are most similar to the sketches shown in the leftmost column of our set of results, which also align well with the input scene structure. The sketches produced by Li *et al.* [21] (marked “Photo-Sketching”) are less detailed, and may lack the semantic meaning of the input scene. For example, in the first row it is difficult to identify the sketch as being that of a person. In contrast, our approach can produce sparse sketches while faithfully maintaining the semantics of the scene. Importantly, none of the alternative methods can produce sketches with varying abstraction levels. For example, observe the different levels of detail in the hair of the woman in the first row,

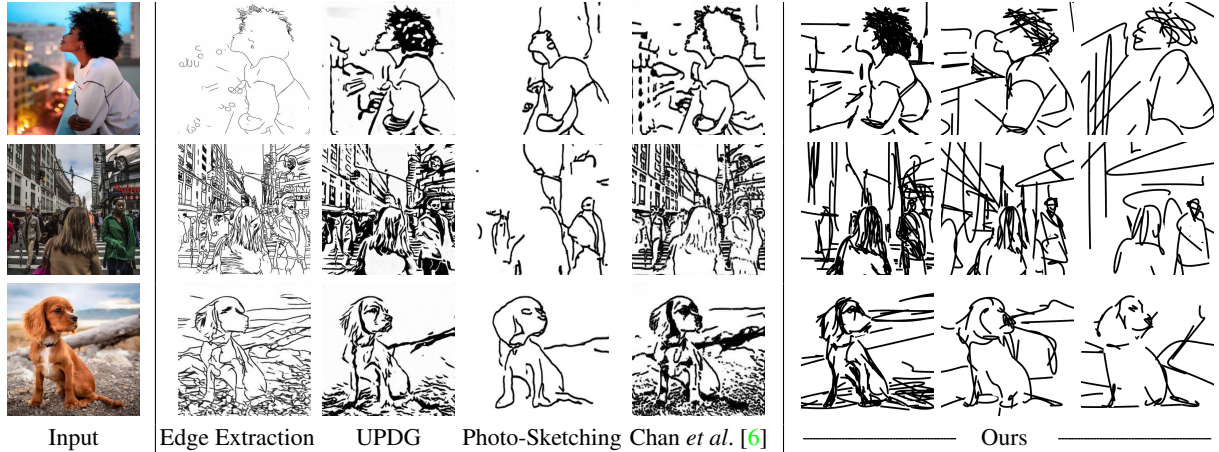


Figure 15. Scene sketching results and comparisons. On the left are the sketches obtained by UPDG [51], Photo-Sketching [21], and Chan et al [6]. On the right, are three representative sketches produced by our method depicting three levels of abstraction. Note that the method by Chan et al. [6] can produce sketches with three different styles, however all the sketches represent a similar level of abstraction. We therefore choose one representative style and provide more comparisons to different styles in the supplementary material.

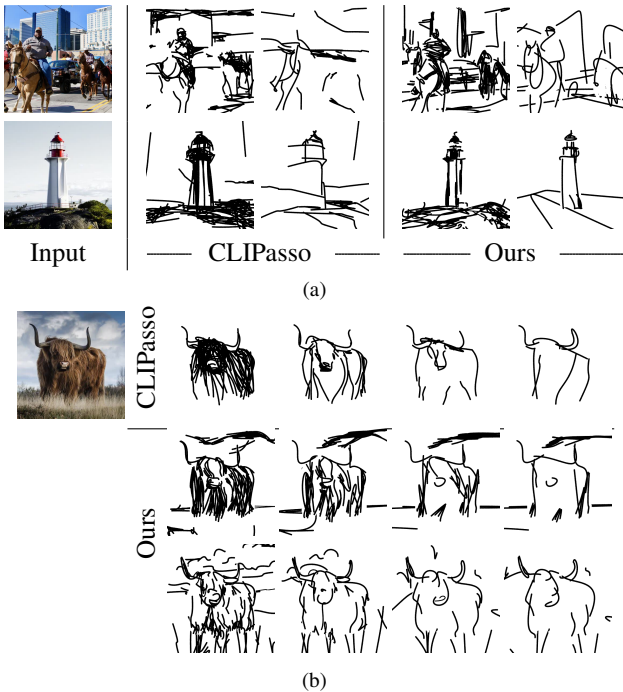


Figure 16. Comparisons with CLIPasso [44]. In (a) we show CLIPasso results obtained when applied over an entire scene image using 128 and 64 strokes respectively. We show our sketch results obtained using approximately the same number of strokes. In (b) we show CLIPasso results applied only over the object in the scene (*i.e.* the bull). For our method, we show simplification results obtained for fidelity levels 1 and 4.

ranging from a highly-detailed depiction to a scribble-like sketch with a few strokes. We do note that in some cases, our sketches may lack some details such as the legs of the

dog in the rightmost image and the windows of the buildings in the second row. Finally, in contrast to the methods above, our sketches are provided in vector format, allowing the user to easily change the stroke style, as well as apply various post-processing operations.

In Figure 16a we provide a comparison to CLIPasso [44]. For each scene, we present sketches obtained by CLIPasso using 128 and 64 strokes, respectively, which is approximately equal to the number of strokes returned by our implicit simplification. As can be seen, when operating over a complete scene image, CLIPasso struggles in balancing the emphasis placed on the subject and the details provided in the background. On the right, we present the sketches produced by our method at two levels of abstraction. As can be seen, our method is able to capture the scene semantics and structure even when composing a sparse sketch. In the appendix, we provide an additional comparison to CLIPasso without our scene decomposition technique.

In Figure 16b, 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.

4.3. Quantitative Evaluation

In this section, we provide a quantitative evaluation of our method’s ability to produce sketch abstractions along both the simplicity and fidelity abstraction axes. To compute our quantitative metrics, we collected a variety of images spanning five classes of scene imagery: people, urban, nature, indoor, and animals. We collected eight images for each class, and for each image we obtained the 4×4 sketch abstraction matrix using our sketching technique – resulting in a total number of 640 sketches.

Table 1. Geometry preservation of our generated sketches by fidelity level, across different fidelity levels using the MS-SSIM [45] metric.

| Category | 11 | 12 | 13 | 14 |
|----------|-------|-------|-------|-------|
| People | 0.295 | 0.18 | 0.163 | 0.129 |
| Urban | 0.274 | 0.169 | 0.153 | 0.126 |
| Nature | 0.292 | 0.156 | 0.134 | 0.126 |
| Indoor | 0.372 | 0.183 | 0.166 | 0.132 |
| Animals | 0.344 | 0.236 | 0.184 | 0.166 |

Fidelity Changes We begin by analyzing the ability to produce sketches at different levels of fidelity. As we move right along the fidelity axis, we should expect the geometry of the sketches to increasingly deviate from that of the original input. To assess this aspect of the generated sketches, we compute the MS-SSIM [45] score between the edge map (extracted using XDoG [47]) and each generated sketch. In Table 1 we show the average MS-SSIM score across all images in the same scene category, split by the four fidelity levels. As we move to the right along the fidelity axis, the MS-SSIM score gradually decreases, indicating that the sketches gradually become “looser” with respect to the input geometry.

Sketch Recognizability It is essential to ensure that the sketches remain recognizable as depicting the same input scene across different levels of abstraction. To evaluate this, we define a set of 150 object class names taken from commonly used image classification and object detection datasets [20, 25]. Using a pre-trained ViT-B/16 CLIP model (different than the one used for training), we performed zero-shot image classification over each input image and the corresponding 16 generated sketches. We then computed the percent of sketches where one of the top 3 image classes was present in the sketch’s top 3 classes. We consider the top 3 predicted classes since a scene image naturally contains multiple objects to be captured by the classifier and

Table 2. Recognizability rates by simplicity and fidelity levels. To evaluate the recognizability of our generated sketches, we utilize a pre-trained CLIP model and perform zero-shot classification on the input image and each generated sketch. We then compute the percent of sketches where at least one of the top 3 predicted image classes appears in the top 3 predicted sketch classes. We present the results split across different scene classes and fidelity/simplicity levels.

| Category | By Simplicity Level | | | | By Fidelity Level | | | |
|----------|---------------------|------|------|------|-------------------|------|------|------|
| | 11 | 12 | 13 | 14 | 11 | 12 | 13 | 14 |
| People | 0.90 | 0.88 | 0.82 | 0.64 | 0.61 | 0.84 | 0.91 | 0.89 |
| Urban | 0.91 | 0.84 | 0.74 | 0.70 | 0.71 | 0.84 | 0.86 | 0.87 |
| Nature | 1.00 | 0.97 | 0.90 | 0.90 | 0.78 | 1.00 | 1.00 | 0.97 |
| Indoor | 1.00 | 0.94 | 0.97 | 0.92 | 0.84 | 0.94 | 1.00 | 1.00 |
| Animals | 0.92 | 0.90 | 0.80 | 0.78 | 0.62 | 0.94 | 0.88 | 0.94 |

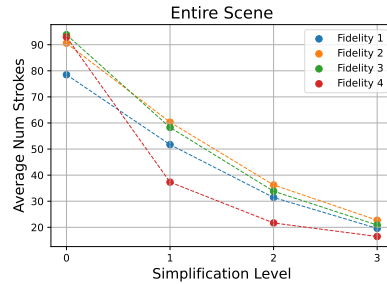


Figure 17. Examining the number of strokes used to compose the sketch across each fidelity and simplification level. Results are averaged across all images across all scene categories. In the supplementary materials, we additionally illustrate the number of strokes split between foreground and background and between the five scene categories.

their order may slightly vary between the original input image and the generated sketches.

Table 2 show the average score across all images belonging to the same scene class, splitted between fidelity and simplicity levels. Observe that the recognizability of the sketches remains relatively consistent across different simplicity levels, with a naturally slight decrease as we increase the abstraction level. This indicating that our method is able to capture the core characteristics of the scene image even at high simplicity levels, as desired.

Across different fidelity levels, we observe a rather large decrease in the recognition score in the first level for several classes (animals and people). This discrepancy can be attributed to the first fidelity level capturing image structure. For the remaining fidelity levels, we achieve much higher score across all scene categories, indicating that the sketches along the fidelity axis indeed capture the image semantics. We provide some example classification predictions in the appendix.

Simplicity Levels 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 640 sketches. We present the results in Figure 17, 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.

5. Summary and Future Work

We presented a method for performing scene sketching using different types and multiple levels of abstraction. We disentangle the concept of sketch abstraction into two axes: *fidelity* and *simplicity*. We demonstrated the ability to cover a wide range of abstractions across various challenging scene images of varying complexities.

There are several limitations to our method. First, since we separate foreground and background, and generate many sketches per one scene, generating a 4×4 matrix of abstractions requires three hours on a single commercial GPU. Although further optimization is possible, sketches along the simplicity axis depends on the previous step. Second, there are cases where some sketches of the resulting matrix contain unwanted artifacts, see the additional results in the appendix. However, in most cases, there are sufficient high-quality alternative sketches at various levels of fidelity and simplicity. Third, we defined two axes for sketch abstraction, however, other axes of abstraction could be defined. Lastly, our resulting sketches cover only a portion of each of the axes. A future extension could focus on further extending these axes.

This work has focused on generating sketches of an input static scene image, an interesting future direction could extend the work to support video. Finally, future potential uses of our proposed method could focus on generating image-sketch paired datasets containing a range of styles and abstractions. Such data could potentially be used to train multi-modal machine learning algorithms.

References

- [1] AUTOMATIC1111. Stable diffusion webui. <https://github.com/AUTOMATIC1111/stable-diffusion-webui>, 2022. 32
- [2] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Trans. Graph.*, 32(4), jul 2013. 2, 3
- [3] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Fahad Shahbaz Khan, Jorma Laaksonen, and Michael Felsberg. Doodleformer: Creative sketch drawing with transformers. *ECCV*, 2022. 3
- [4] Irving Biederman and Ginny Ju. Surface versus edge-based determinants of visual recognition. *Cognitive psychology*, 20(1):38–64, 1988. 1
- [5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, pages 679–698, 1986. 3
- [6] 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. 3, 8, 9, 32, 33, 34
- [7] Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 387–396, 2021. 14
- [8] 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. 15
- [9] 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. 3, 4, 22
- [10] Gustav Theodor Fechner. *Elemente der psychophysik*. 1860. 3, 6, 22
- [11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *CoRR*, abs/2106.14843, 2021. 3
- [12] 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. 32
- [13] Songwei Ge, Vedanuj Goswami, Larry Zitnick, and Devi Parikh. Creative sketch generation. In *International Conference on Learning Representations*, 2021. 3
- [14] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018. 3, 4
- [15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4
- [16] Aaron Hertzmann. Why do line drawings work? a realism hypothesis. *Perception*, 49(4):439–451, mar 2020. 1
- [17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 32
- [18] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988. 8
- [19] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Ad-*

- vances in neural information processing systems*, 30, 2017. 14
- [20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 10
 - [21] 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. 3, 8, 9
 - [22] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 4
 - [23] Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2pencil: Controllable pencil illustration from photographs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1525–1534, 2019. 3
 - [24] Yi Li, Yi-Zhe Song, Timothy M. Hospedales, and Shaogang Gong. Free-hand sketch synthesis with deformable stroke models. *CoRR*, abs/1510.02644, 2015. 3
 - [25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 10
 - [26] Difan Liu, Matthew Fisher, Aaron Hertzmann, and Evangelos Kalogerakis. Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14204–14213, 2021. 3
 - [27] Vijish Madhavan. Artline. <https://github.com/vijishmadhavan/ArtLineTechnical-Details>, 2020. 3
 - [28] Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. *Advances in Neural Information Processing Systems*, 34:7153–7166, 2021. 3
 - [29] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning deep sketch abstraction. *CoRR*, abs/1804.04804, 2018. 2, 3, 4
 - [30] 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. 32
 - [31] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkan Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 953–961, 2021. 3
 - [32] 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. 7, 14
 - [33] Shuwen Qiu, Sirui Xie, Lifeng Fan, Tao Gao, Song-Chun Zhu, and Yixin Zhu. Emergent graphical conventions in a visual communication game. *arXiv preprint arXiv:2111.14210*, 2021. 3
 - [34] 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. 3, 4, 22
 - [35] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021. 4
 - [36] 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. 32
 - [37] 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. 32
 - [38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2021. 32
 - [39] 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. 32
 - [40] Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Transactions on Graphics (TOG)*, 35(4):1–11, 2016. 8
 - [41] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Learning to sketch with shortcut cycle consistency, 2018. 3
 - [42] 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. 7, 14
 - [43] Zhengyan Tong, Xuanhong Chen, Bingbing Ni, and Xiaohang Wang. Sketch generation with drawing process guided by vector flow and grayscale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 609–616, 2021. 3
 - [44] 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, 3, 4, 9, 14, 15, 25
 - [45] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multi-scale structural similarity for image quality assessment. In *The Thirty-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 10

- [46] E H Weber. De pulsu, resorptione, auditu et tactu. 1834. [3](#), [6](#), [22](#)
- [47] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C. Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Comput. Graph.*, 36:740–753, 2012. [3](#), [10](#)
- [48] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. [3](#)
- [49] Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. [3](#)
- [50] Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L Rosin. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2019. [3](#)
- [51] 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. [3](#), [8](#), [9](#), [32](#), [34](#)
- [52] 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. [32](#)
- [53] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to sketch with deep q networks and demonstrated strokes. *ArXiv*, abs/1810.05977, 2018. [3](#)

Appendix

Table of Contents

| | |
|---|-----------|
| A Implementation Details | 14 |
| A.1. Image Preprocessing | 14 |
| A.2. Sketch Initialization | 14 |
| A.3. MLP Architecture | 14 |
| A.4. MLP Training | 14 |
| A.5. Matrix Composition | 15 |
| B Additional Results | 15 |
| C Additional Quantitative Analysis | 22 |
| C.1. Sketch Recognizability | 22 |
| C.2. Number of Strokes by Simplicity Level | 22 |
| D Ablation Study: General Design Choices | 25 |
| D.1. ViT vs. ResNet | 25 |
| D.2. Foreground-Background Separation . | 25 |
| E Ablation Study: Simplicity Axis | 27 |
| E.1. Explicitly Defining the Number of Strokes | 27 |
| E.2. Replace the Ratio Loss With a Target Number of Strokes | 28 |
| E.3. Fine-tuning MLP-loc During Simplification | 28 |
| E.4. Defining the Function $f-k$ as an Exponential | 28 |
| E.5. Defining a Different Set of Factors for Each Layer | 29 |
| E.6. Defining a Different Sampling Step for Each $f-k$ | 30 |
| F Ablation Study: Fidelity Axis | 31 |
| F.1. Using l-4 for Object Sketching | 31 |
| F.2. Using Other ViT Layers for Training . | 31 |
| G Additional Comparisons | 32 |
| G.1. Diffusion Models | 32 |
| G.2. Scene Sketching Approaches | 32 |

A. 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.

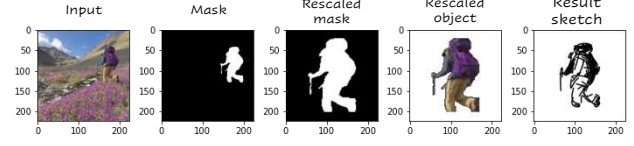


Figure 18. Object re-scaling procedure.

A.1. Image Preprocessing

As stated in the paper, we use a pre-trained U²-Net salient object detector [32] to extract the scene’s salient object(s). To receive a binary map, we threshold the resulting map from U²-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 [42] 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 pre-processing 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 18.

A.2. Sketch Initialization

For initializing the locations of the n strokes, we follow the saliency-based initialization introduced in Vinker *et al.* [44]. Specifically, we employ the vision transformer interpretability method from Chefer *et al.* [7] to extract a relevancy map from a pre-trained ViT-B/32 CLIP model. We use the resulting relevancy map to sample the location of each Bézier curve such that pixels in salient regions are assigned a higher probability of being selected.

A.3. MLP Architecture

Our MLP networks are simple 3-layer networks with SeLU [19] activations. For MLP_{simp} we append a Sigmoid activation to convert the final outputs to probabilities.

A.4. 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 $\{2, 7, 10, 11\}$, 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 $1e-4$ for training both MLP networks. The input to MLP_{simp} is set to be a constant-valued vector of dimension n . We set this constant equal to 1.5, although we found that other scalar values can be also used.

Augmentations As also done in Vinker *et al.* [44], 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 [8], 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.

A.5. 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.

B. Additional Results

We begin with additional results generated by our method. In Figures 19 to 23 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 24. 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 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.

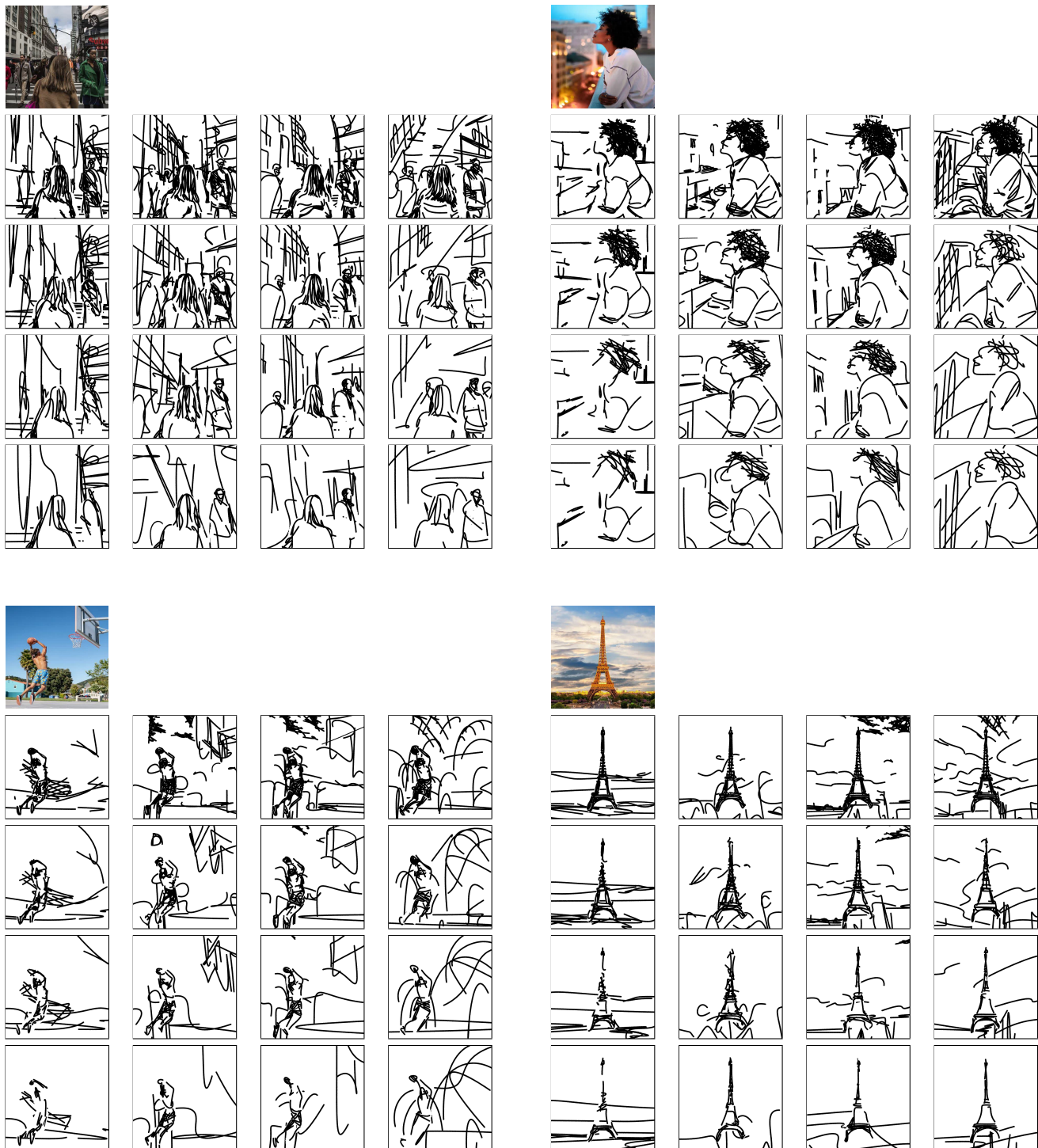


Figure 21. The 4×4 matrix of sketches produced by our method. Columns from left to right illustrate the change in fidelity, from precise to loose, and rows from top to bottom illustrate the visual simplification.



Figure 22. 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 23. 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.

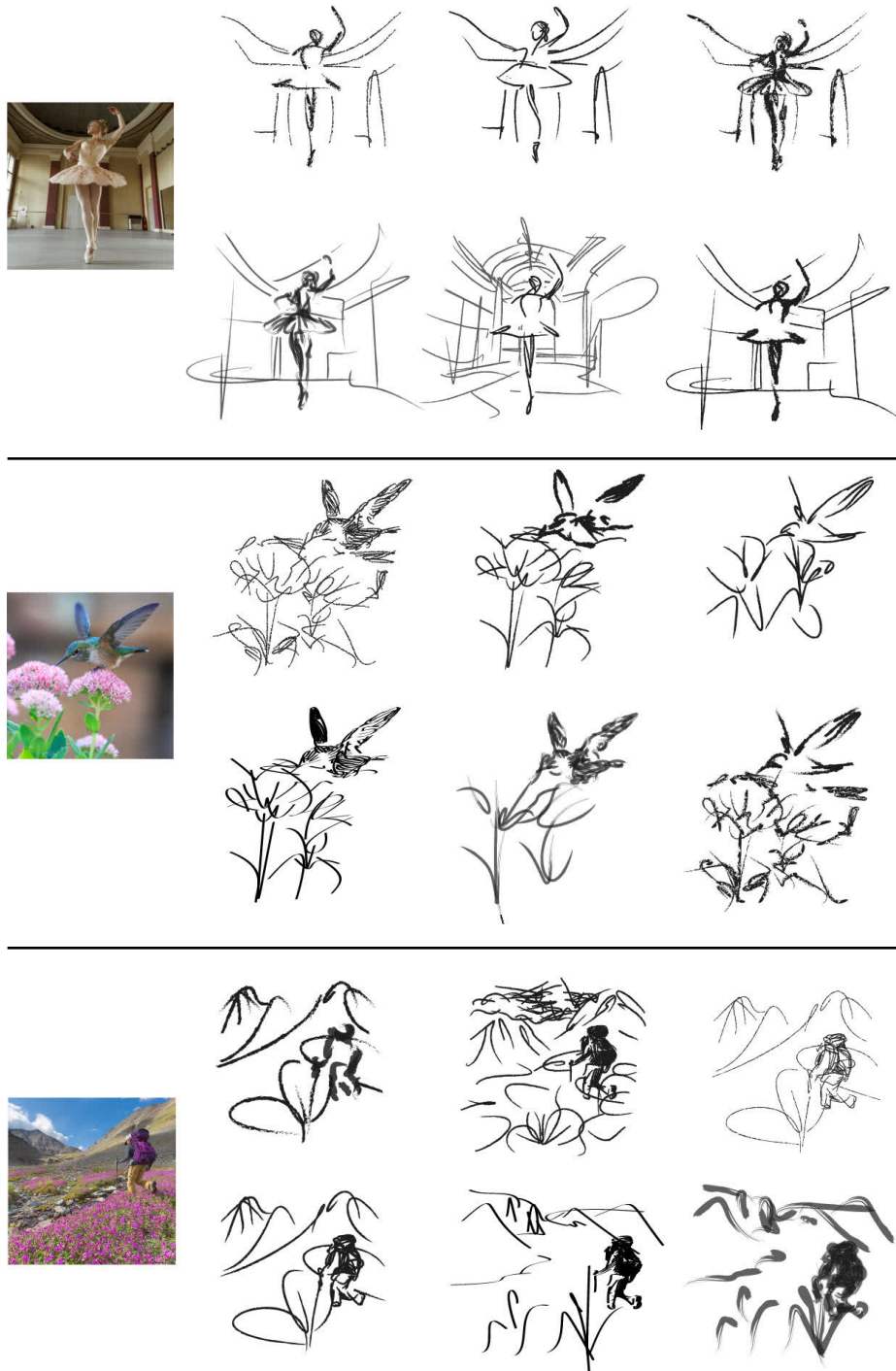


Figure 24. Additional control. For each image, we combine foreground and background sketches from different levels of abstraction, and edit the style of strokes using Adobe Illustrator. This illustrates the power of our method in providing various options for the user to edit the resulted sketches.

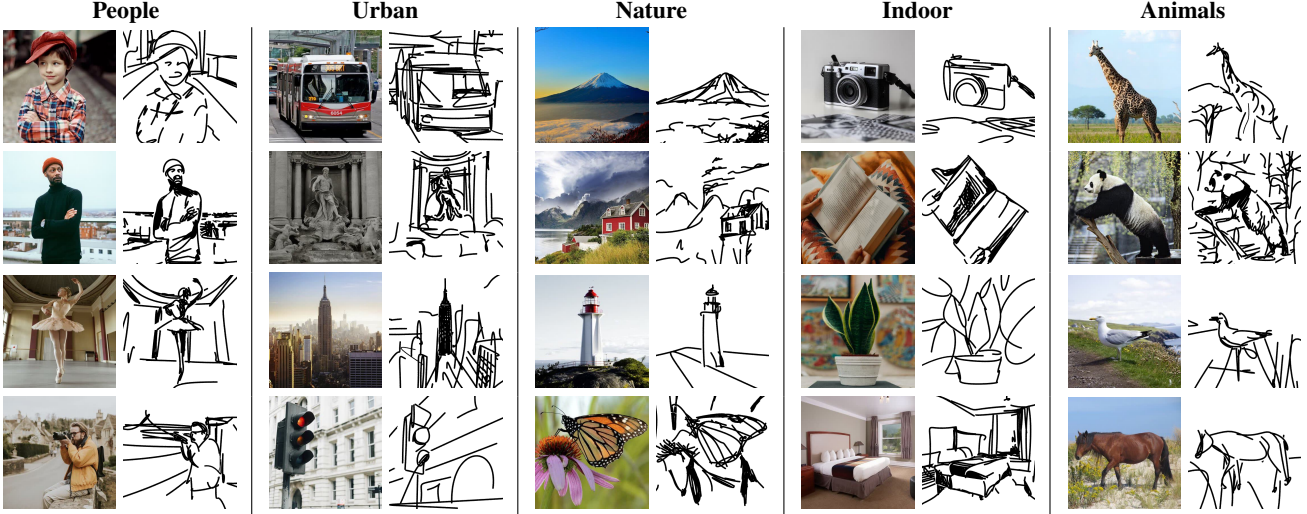


Figure 25. Example images and representative sketches used for our quantitative evaluations.

C. Additional Quantitative Analysis

In this section, we provide additional details, examples, and results regarding the quantitative evaluations presented in the paper. First, in Figure 25 we present example inputs and representative generated sketches for each of our five scene categories used for evaluations.

C.1. Sketch Recognizability

To compute our recognizability metrics, we perform zero-shot classification using a pre-trained ViT-B/16 [9] 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 [34] and apply 80 prompt templates when defining our 150 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 26, we present example zero-shot classification results obtained on various input images and sketches across our five scene categories.

C.2. Number of Strokes by Simplicity Level

In the main paper, we examined the number of strokes used to compose our sketches across the different fidelity and simplification levels. For conciseness, we presented the average number of strokes used across all images and all scene categories. In Figure 27, 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. This behavior aligns well with the Weber-Fechner law [10, 46] which states that an exponentially-changing signal corresponds to a change that is perceived to be linear.

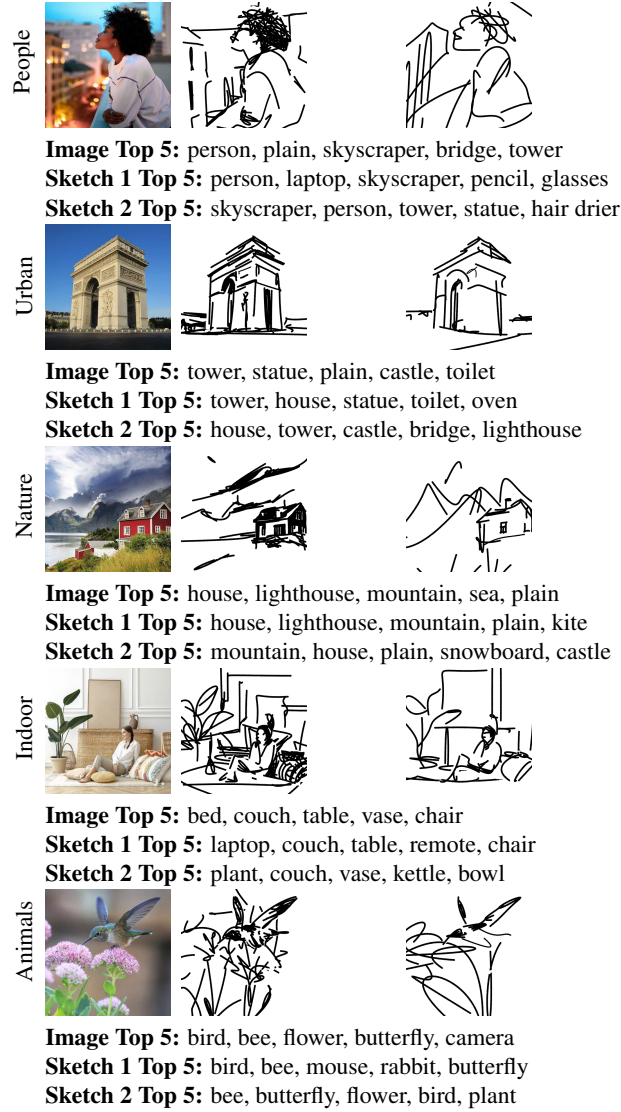


Figure 26. Examples of CLIP zero-shot class predictions on various input images and representative sketches of varying abstractions. These predictions are then used to compute a recognizability metric for each scene category across different levels of abstractions (see Section 4.3 in the main paper).

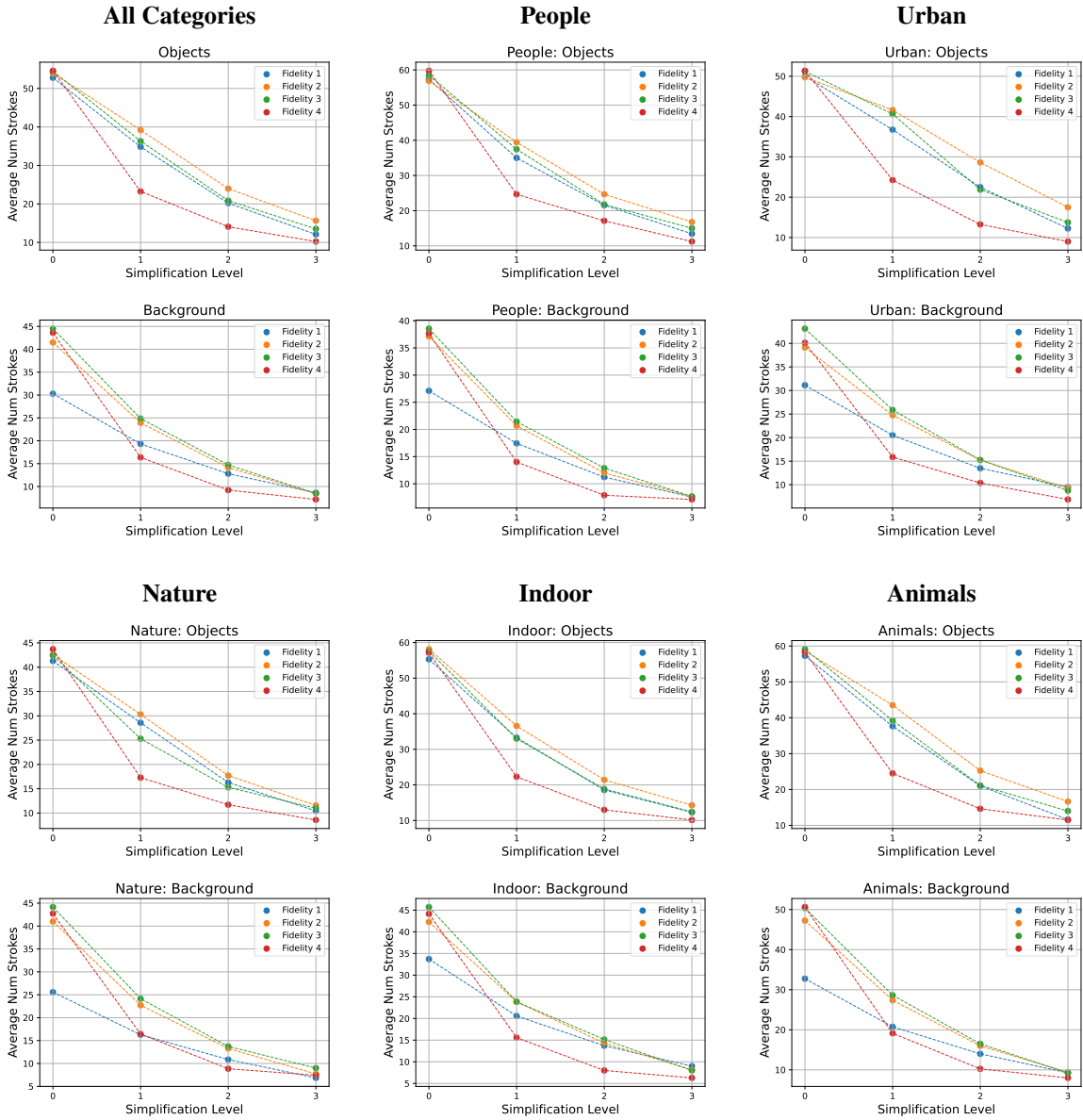


Figure 27. Examining the number of strokes used to compose the sketch across each fidelity and simplification level, split between the different scene categories.

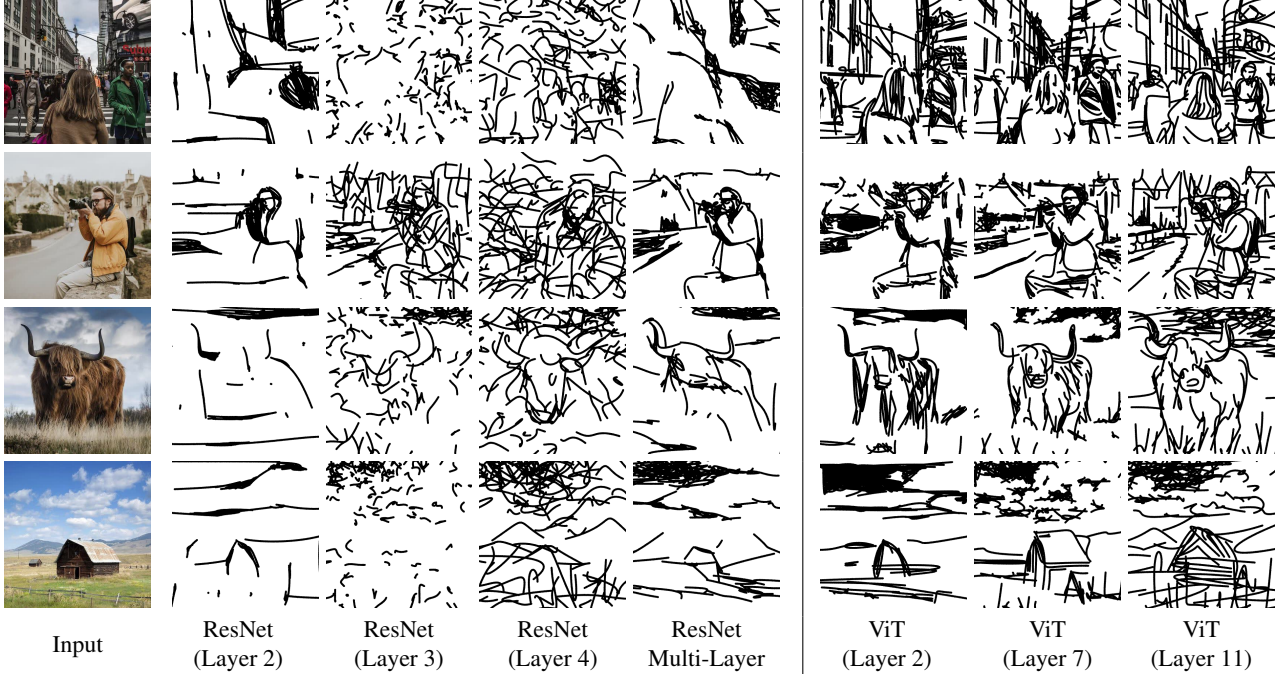


Figure 28. Ablation study on using ResNet-CLIP and ViT-CLIP for guiding the training process. For both variants, we generate the sketches for the entire scene together (*i.e.* no scene decomposition is performed) and set the number of strokes to 128. In addition to evaluating ResNet using a single layer 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 [44] 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.

D. Ablation Study: General Design Choices

D.1. ViT vs. ResNet

In Figure 28, 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} .

D.2. Foreground-Background Separation

In Section 3.4 of the main paper, we introduced our scene decomposition technique where the foreground and back-

ground components of the input are sketched separately and then merged. In Figure 29 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 29 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}).

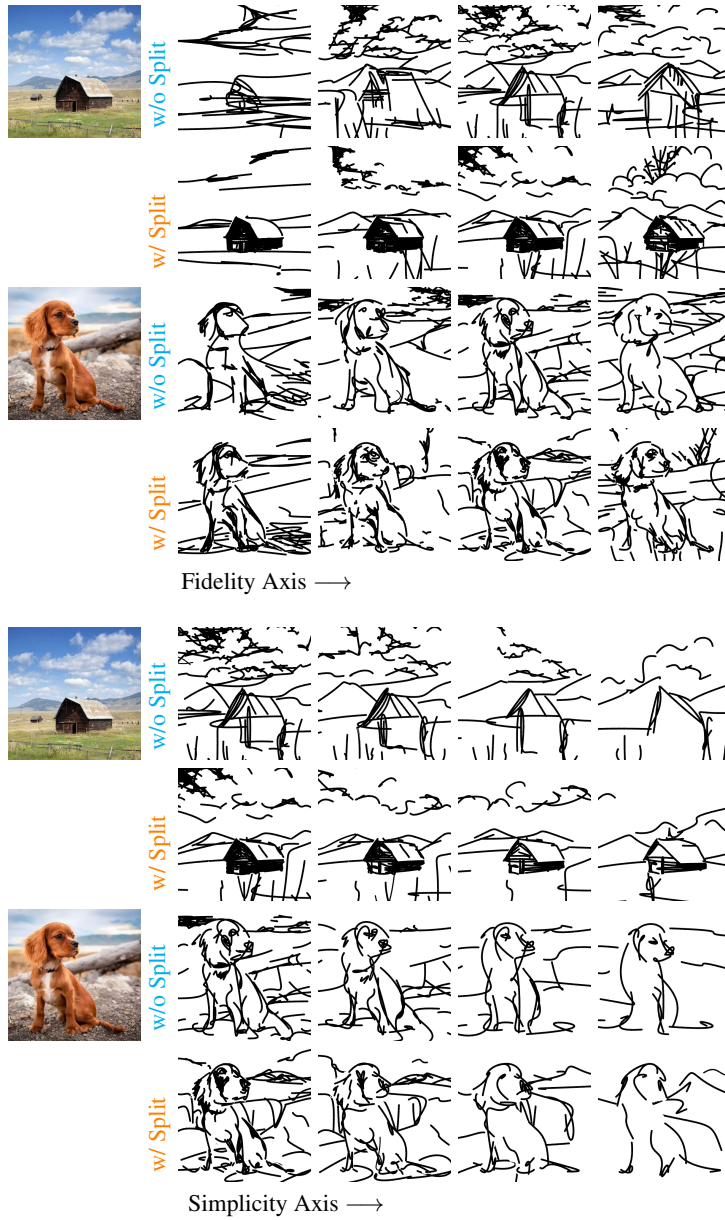


Figure 29. Scene sketching results obtained with and without decomposing the scene. We show sketches generated across both the fidelity axis (first four rows) and the simplicity axis (bottom four rows).

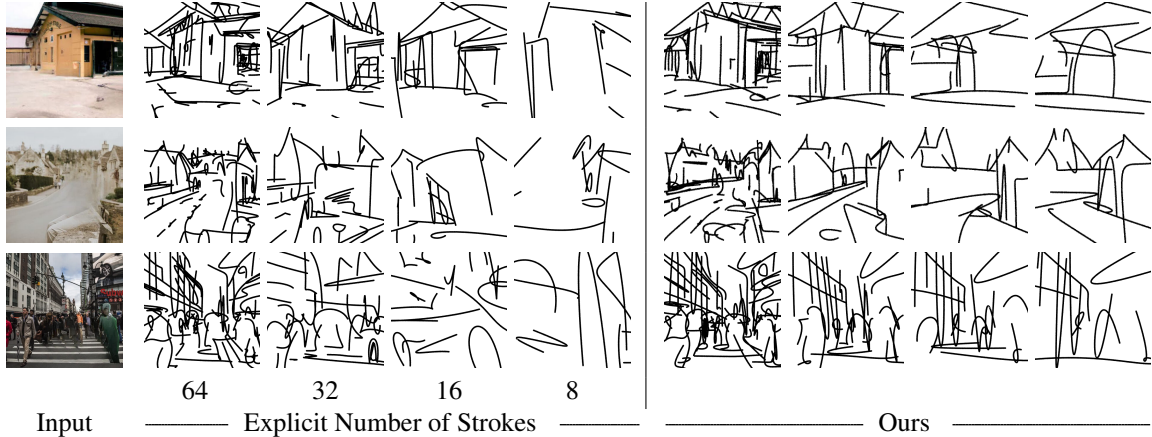


Figure 30. Ablation results of explicitly defining the number of strokes (left) compared to using our implicit simplification scheme (right). The presented sketches of all three input images are of the last fidelity level (obtained using layer 11 of CLIP-ViT).

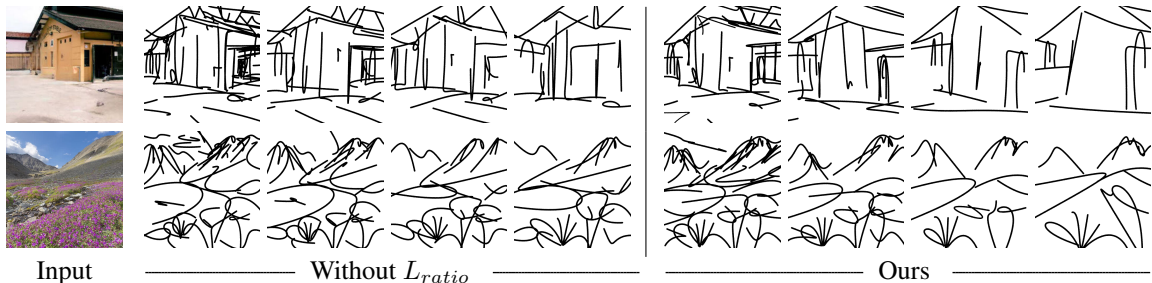


Figure 31. Ablation results of replacing our \mathcal{L}_{ratio} loss with an alternative loss $||\mathcal{L}_{sparse} - n_{target}||$ guided by a desired number of strokes n_{target} .

E. Ablation Study: Simplicity Axis

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

E.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 30 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 im-

age, 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.

E.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 Appendix E.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:

$$\mathcal{L}_{ratio} = ||\mathcal{L}_{sparse} - n_{target}||, \quad (8)$$

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 31 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.

E.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 32, 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 32. 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.

E.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 \mathcal{L}_{CLIP} and \mathcal{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 using a f_k using 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 $\{r_k^1, \dots, r_k^m\}$ represent a constant step size by encouraging the removal of 8 strokes in each step.

In Figure 33 we present the results of this alternative setup. For each set of generated sketches, we additionally present two graphs: (1) the resulting \mathcal{L}_{sparse} as a function of \mathcal{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)

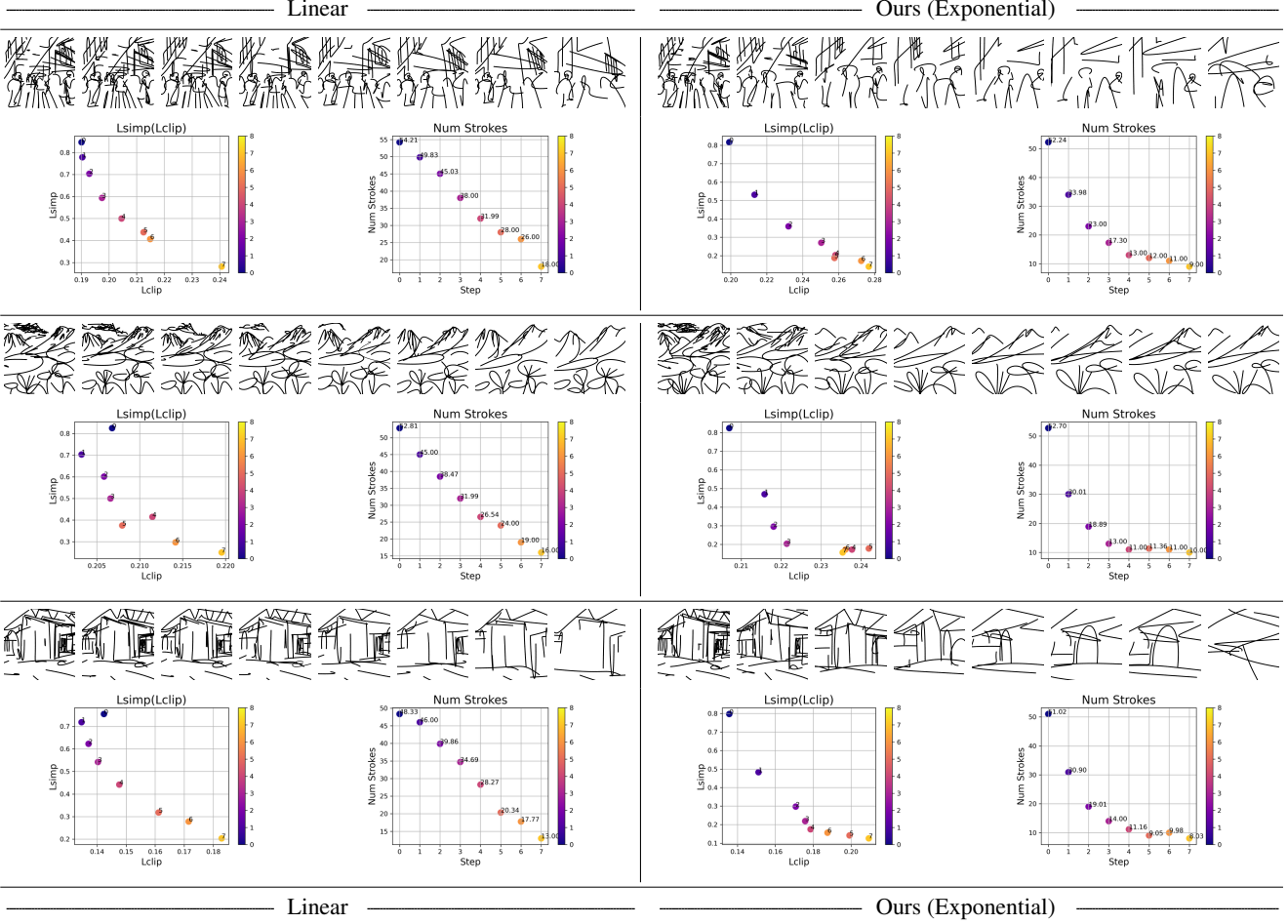


Figure 33. Ablation of the defining an exponential f_k when performing an iterative simplification of the sketch. On the right-hand side, we define a linear relation between the two loss objectives. On the left side, we show sketch results obtained when defining an exponential relation between \mathcal{L}_{CLIP} 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).

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 33 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).

E.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 $\{r_k^1, \dots, r_k^m\}$ 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 34 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

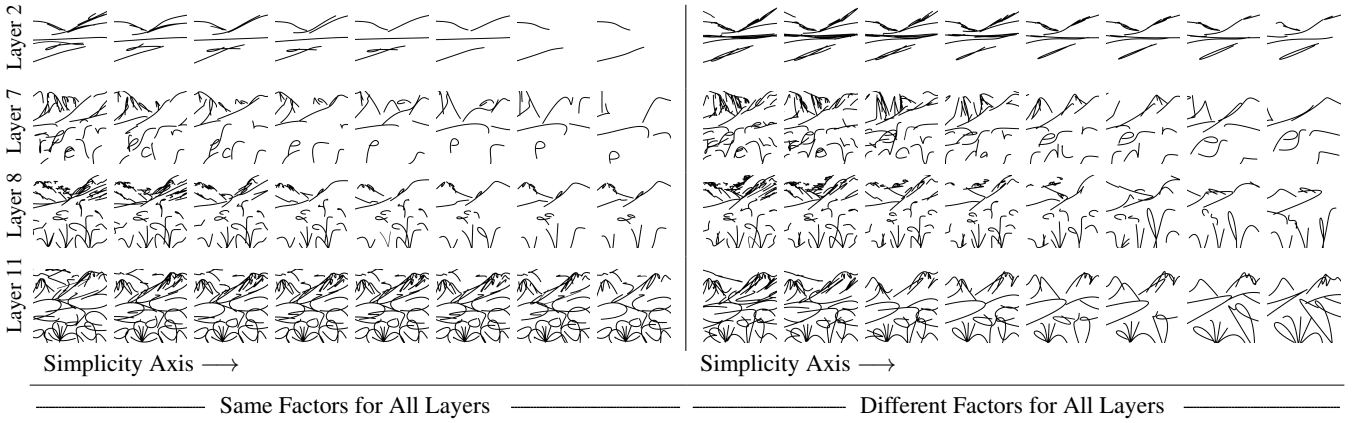


Figure 34. Ablation study on using a different set of factors for each fidelity level k when defining our \mathcal{L}_{ratio} loss. On the left, we should simplification results obtained when applying the same set of factors across all levels. On the right side, we should our simplification results obtained by adjusting the factors for each fidelity level.

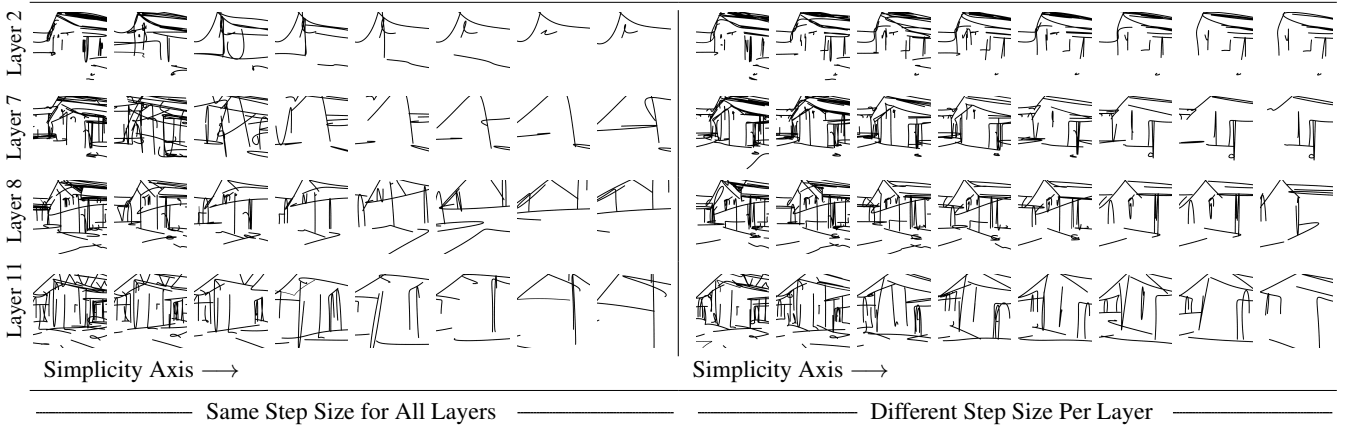


Figure 35. Ablation study on applying the same step size for sampling each function f_k during simplification. On the left, we apply the same step size across all fidelity levels, resulting in non-smooth and non-uniform simplifications between the different fidelity levels. On the right, we show our 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.

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 34, 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.

E.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 35, 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.

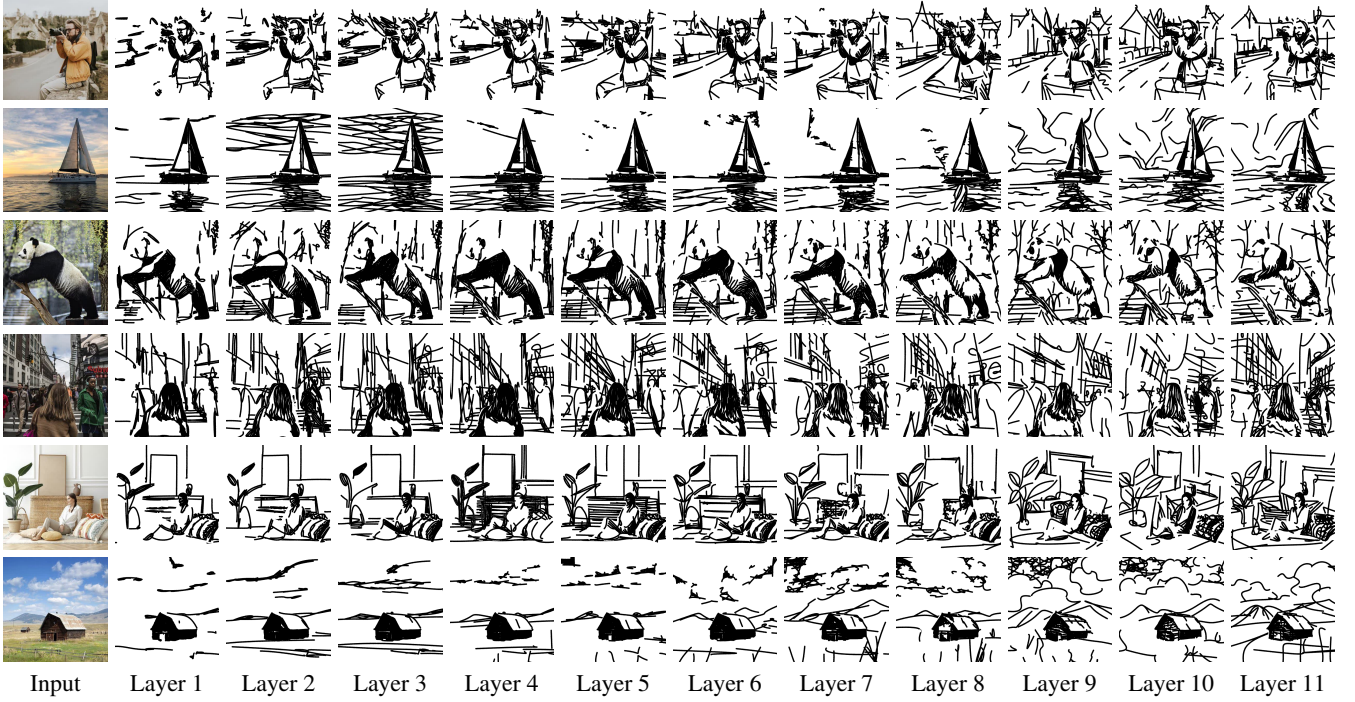


Figure 36. Ablation study on using different ViT layers for computing \mathcal{L}_{CLIP} for generating sketches at different levels of fidelity.

F. 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.

F.1. Using 1-4 for Object Sketching

When decomposing the scene and sketching for the foreground image, we additionally compute \mathcal{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 37. 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.

F.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 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 36.



Figure 37. Ablation results for object sketching when additionally using layer ℓ_4 when computing \mathcal{L}_{CLIP} for object sketching.

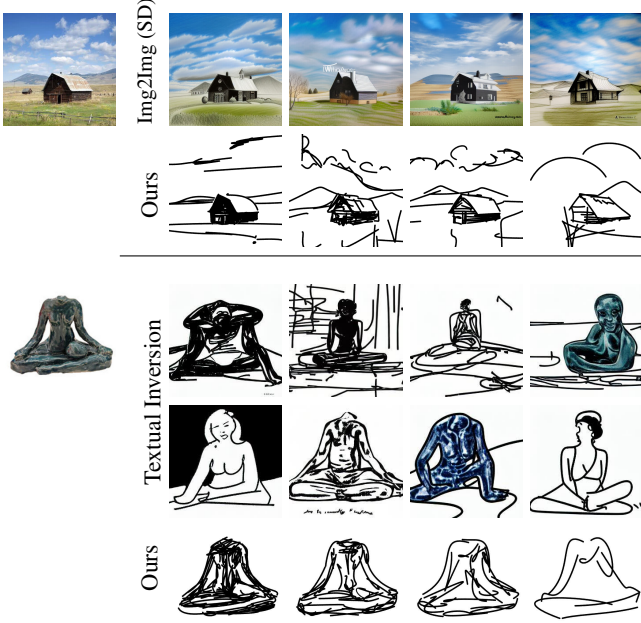


Figure 38. Comparison to various diffusion model-based techniques. At the top, we compare our sketch results to those obtained using Stable Diffusion [38] image-to-image technique guided by the text prompt “A black and white sketch image”. At the bottom, we compare with Textual Inversion [12] by learning 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.

G. Additional Comparisons

G.1. Diffusion Models

Recent advancements in diffusion models [17] have demonstrated an unprecedented ability to generate amazing imagery guided by a target text prompt or image [30, 36–39, 52]. 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 [38]. 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 38. 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.* [12]. 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 [38] 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 38). 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 38. 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.

G.2. Scene Sketching Approaches

In Figure 39 we provide additional scene sketching comparisons to alternative scene sketching methods. In Figure 40 we provide additional sketch comparisons to all styles supported by UPDF [51] and Chan *et al.* [6]. Finally, in Figures 41 and 42 we provide additional comparisons to CLIPasso for both scene sketching and object sketching, including a comparison without our scene decomposition technique.



Figure 39. Scene sketching results and comparisons.

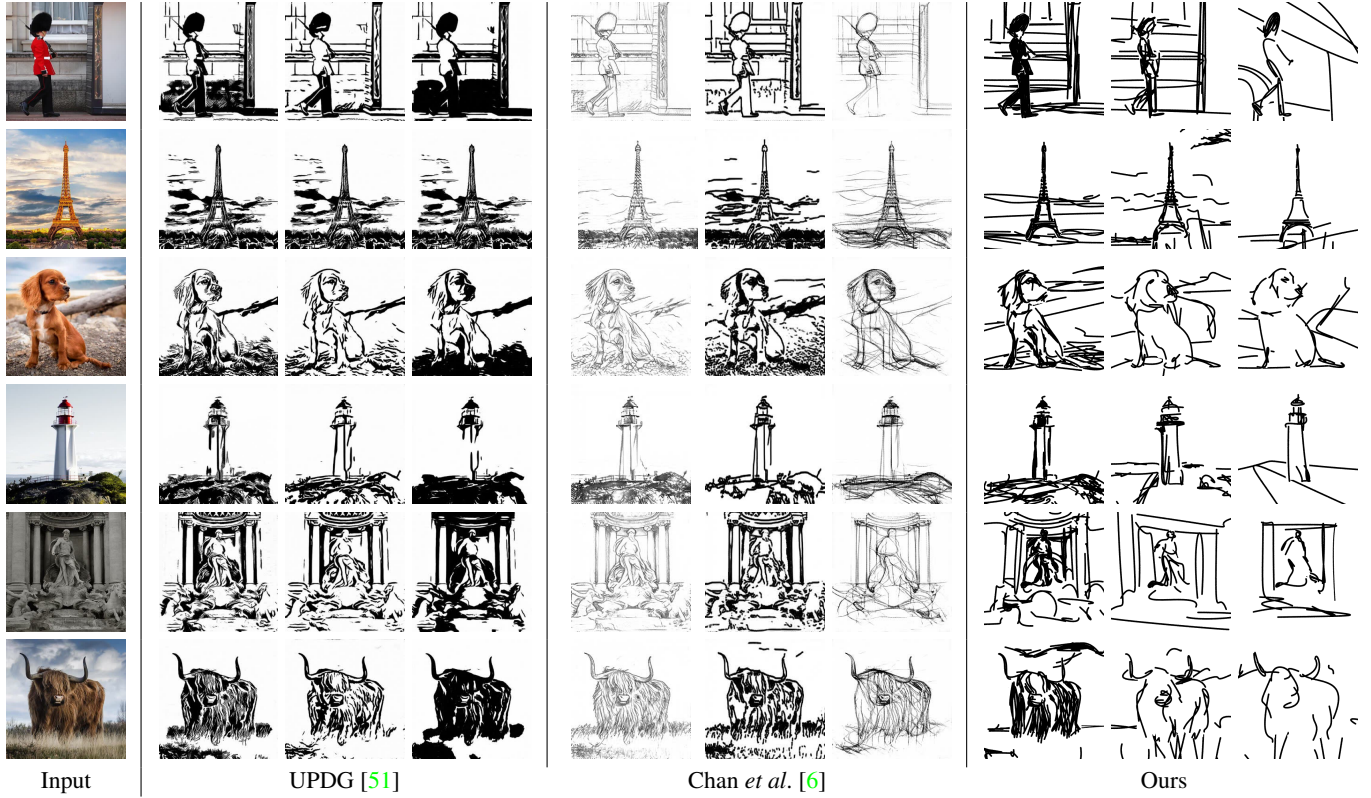


Figure 40. Scene sketching comparison to Chan *et al.* [6] and UPDG [51] across three different styles supported by each of their methods. For our results, we show three sketches illustrating the various levels of abstraction that our method is capable of achieving.

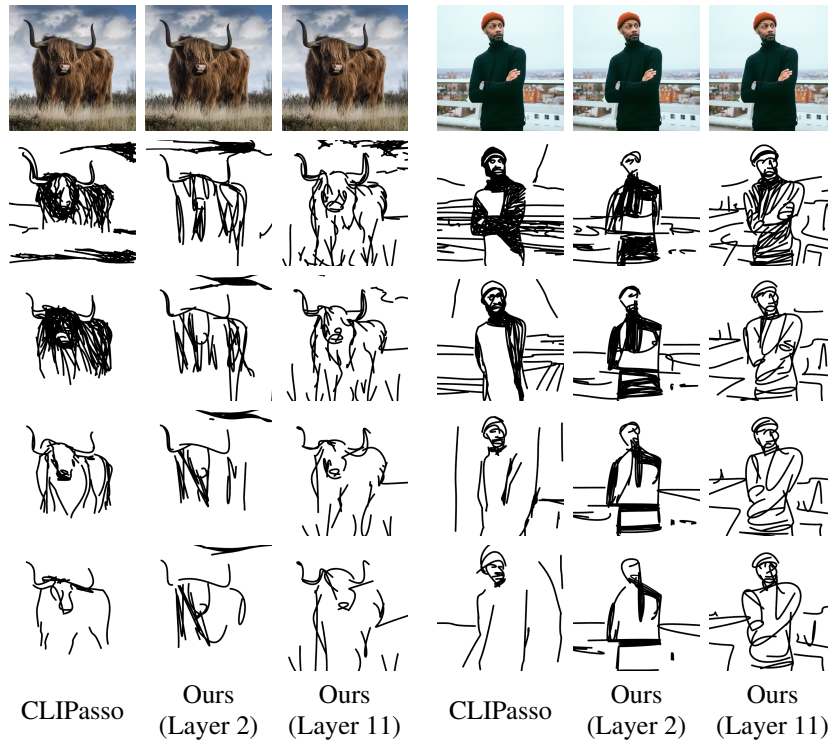
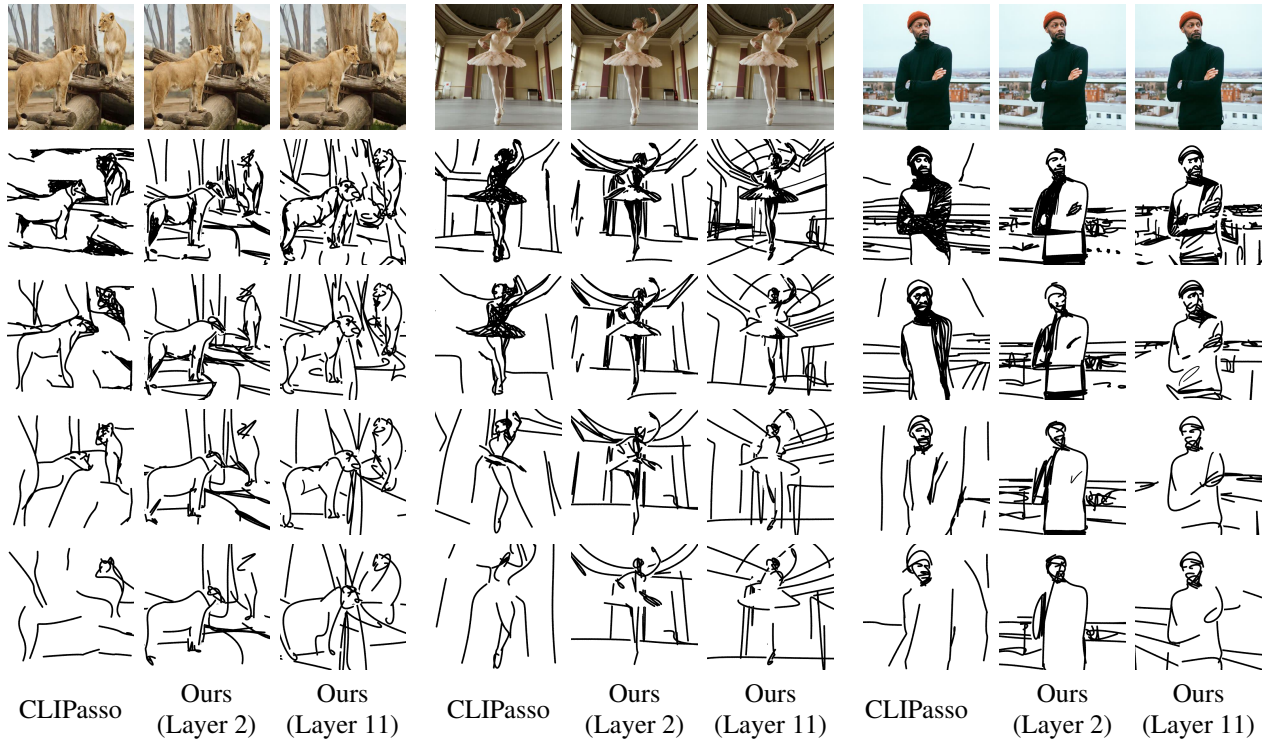


Figure 41. Comparison to CLIPasso. For CLIPasso, we generate the sketches using 128, 64, 32, and 16 strokes. In the top set of results, we use our scene decomposition technique and apply our implicit simplification starting with 64 strokes for the foreground and background sketches. For the bottom set 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.

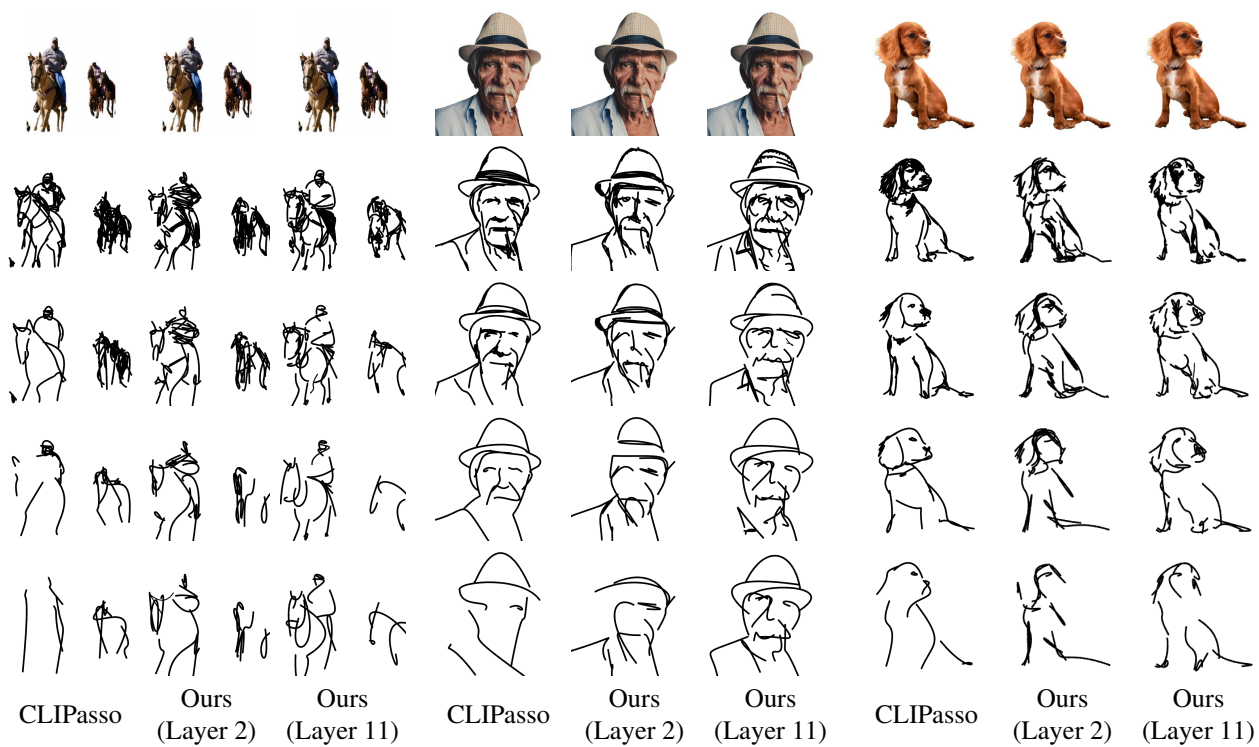


Figure 42. Comparison to CLIPasso on object sketching. For CLIPasso, we generate the sketches using 64, 32, 16, and 8 strokes for the right image. For our approach, we use our implicit simplification starting with 64 strokes. We show our results for layers 2 and 11.