CLIPascene: Scene Sketching with Different Types and Levels of Abstraction

Yael Vinker
Tel Aviv University
yaelvi116@gmail.com

Yuval Alaluf
Tel Aviv University
yuvalalaluf@gmail.com

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

Ariel Shamir
Reichman University
arik@runi.ac.il

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*. The sketches on the left were selected from a complete *matrix* generated by our method (an example is shown on the right), encompassing a broad range of possible sketch abstractions for a given image. Our sketches are generated in vector form, which can be easily used by designers for further editing.

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.* https://clipascene.github.io/CLIPascene/

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 [13, 3]. 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, which requires the ability to understand, analyze, and interpret the complexity of the visual scene. A scene consists of multiple objects of varying complexity, as well as relationships between the foreground and background (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 different *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 [34, 25] or *portrait* sketching [1], and often simply use the number of strokes to define the level of

Figure 2. 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, and (C) contains both foreground and background that include many details. Our work tackles all types of scenes.

Figure 3. Drawings of different scenes by different artists. Notice the significant differences in style and level of abstraction between the drawings — moving from more detailed and precise (left) to more abstract (right). The second row shows how the level of abstraction not only varies *between* drawings, but also *within* the *same* drawing. Where each drawing contains areas that are relatively more detailed (red) and more abstract (blue).

abstraction. We are not aware of any previous work that attempts to separate different *types* of abstractions. Moreover, existing works for *scene* sketching often focus on producing sketches based on certain styles, without taking into account the abstraction level, which is an essential concept in sketching. Lastly, most existing methods for scene sketching do not produce sketches in vector format. Providing vector-based sketches is a natural choice for sketches as it allows further editing by designers (such as in Fig. 6).

We define two axes representing two types of abstractions and produce sketches by gradually 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. An example is shown in Figure 4, where the leftmost sketch follows the contours of the mountains on the horizon, and as we move right, the mountains and the flowers in the front gradually deviate from the edges present in the input, but still convey the correct semantics of the scene. The second axis governs the level of details of the sketch and moves from detailed to sparse 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

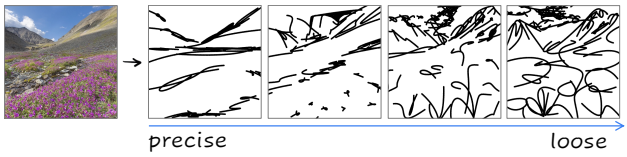

Figure 4. The *fidelity* axis. From left to right, using the same number of strokes the sketches gradually depart from the geometry of the input image, but still convey the semantics of the scene.

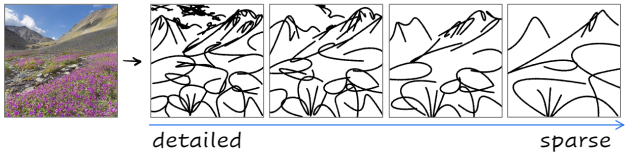

Figure 5. The *simplicity* axis. On the left, we start with a more detailed sketch and as we move to the right the sketch is gradually simplified while still remaining consistent with the overall appearance of the initial sketch.

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 and sketch each of them separately. This explicit separation and the disentanglement into two abstraction axes provide a more flexible framework for computational sketching, where users can choose the desired sketch from a range of possibilities, according to their goals and personal taste.

We define a sketch as a set of Bézier curves, and train a simple multi-layer perceptron (MLP) network to learn the stroke parameters. Training is performed per image (*e.g.* without an external dataset) and is guided by a pre-trained CLIP-ViT model [29, 7], 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 emphasize the scene's semantics.

To realize the *simplicity* axis, we jointly train an additional MLP network that learns how to best discard strokes gradually and smoothly, without harming the recognizability of the sketch. As shall be discussed, the use of the networks over a direct optimization-based approach allows us to define the level of details *implicitly* in a learnable fashion, as opposed to explicitly determining the number of strokes.

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, as shown in Figure 1. We compare our results with existing methods for 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.

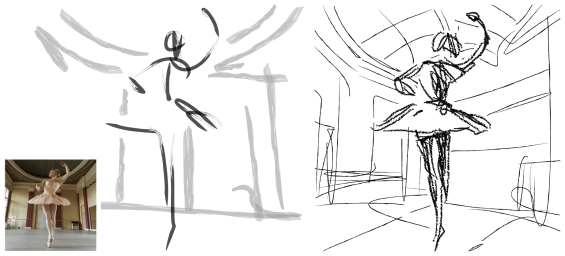

Figure 6. Artistic stylization of the strokes using Adobe Illustrator.

2. Related Work

Free-hand sketch generation differs from edge-map extraction [4, 37] in that it attempts to produce sketches that are representative of the style of human drawings. Yet, there are significant differences in drawing styles among individuals depending on their goals, skill levels, and more (see Figure 3). As such, computational sketching methods 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 [16, 38, 33, 19], to those that aim to produce sketches that are more abstract [2, 9, 34, 11, 26, 10, 28, 23, 42]. Several works have attempted to develop a unified algorithm that can output sketches with a variety of styles [5, 41, 21]. There are, however, only a few works that attempt to provide various levels of abstraction [1, 25, 34]. In the following, we focus on scene-sketching approaches, and we refer the reader to [39] for a comprehensive survey on computational sketching techniques.

Photo-Sketch Synthesis Various works formulate this task as an image-to-image translation task using paired data of corresponding images and sketches [18, 40, 16, 22]. Others approach the translation task via unpaired data, often relying on a cycle consistency constraint [41, 31, 5]. Li *et al*. [16] introduce a GAN-based contour generation algorithm and utilize multiple ground truth sketches to guide the training process. Yi *et al*. [41] generate portrait drawings with unpaired data by employing a cycle-consistency objective and a discriminator trained to learn a specific style.

Recently, Chan *et al*. [5] 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 to achieve semantically-aware sketches. 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. In contrast, 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 [29]. Additionally, our work is the only one among the alternative scene sketching approaches that provides sketches with multiple levels of abstraction and in vector form, which allows for a wider range of editing and manipulation.

Sketch Abstraction 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 done so over an entire scene. Berger *et al*. [1] collected portrait sketches at different levels of abstraction from seven artists to learn a mapping from a face photograph to a portrait sketch. Their method is limited to faces only and requires a new dataset for each desired level of abstraction. Muhammad *et al*. [25] train a reinforcement learning agent to remove strokes from a given sketch without harming the sketch's recognizability. The recognition signal is given by a sketch classifier trained on nine classes from the QuickDraw dataset [11]. Their method is therefore limited only to objects from the classes seen during training and requires extensive training.

CLIPasso Most similar to our work is CLIPasso [34] 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 [29] 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. In contrast to CLIPasso, 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 fidelity of the sketch. Moreover, in CLIPasso, the user is required to *explicitly* define the number of strokes needed to obtain the desired abstraction. However, different images require a different number of strokes, which is difficult to determine in advance. In contrast, we *implicitly* learn the desired number of strokes by training two MLP networks to achieve a desired trade-off between simplicity and fidelity with respect to the input image.

3. Method

Given an input image $\mathcal{I}$ of a scene, our goal is to produce a set of corresponding sketches at n levels of *fidelity* and m levels of *simplicity*, forming a sketch abstraction matrix of size $m \times n$. We begin by producing a set of sketches along the *fidelity* axis (Sections 3.1 and 3.2) with no simplification, thus forming the top row in the abstraction matrix. Next, for each sketch at a given level of fidelity, we perform an iterative visual *simplification* by learning how to best remove select strokes and adjust the locations of the remaining strokes (Section 3.3). For clarity, in the following we describe our method taking into account the entire scene as a whole. However, to allow for greater control over the

appearance of the output sketches, and to tackle the high complexity presented in a whole scene, our final scheme splits the image into two regions – the salient foreground object(s), and the background. We apply our 2-axes abstraction method to each region separately, and then combine them to form the matrix of sketches (details in Section 3.4).

3.1. Training Scheme

We define a sketch as a set of n 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 $Z_{init} \in R^{n\times4\times2}$ (marked in blue) and returns a vector of offsets $MLP_{loc}(Z_{init}) = \Delta Z \in \mathbb{R}^{n\times4\times2}$ 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 $\mathcal{R}$ [17] that outputs the rasterized sketch,

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

For initializing the locations of the n strokes, we follow the saliency-based initialization introduced in Vinker *et al*. [34], in which, strokes are initialized in salient regions based on a relevancy map extracted automatically [6].

To guide the training process, we leverage a pre-trained CLIP model due to its capabilities of encoding shared information from both sketches and natural images. As opposed to Vinker *et al*. [34] that use the ResNet-based [12] CLIP model for the sketching process (and struggles with depicting a scene image), we find that the ViT-based [7] CLIP model is able to capture the global context required for generating a coherent sketch of a whole scene, including both foreground and background. This also follows the observation of Raghu *et al*. [30] that ViT models better capture more global information at lower layers compared to ResNet-based models. We further analyze this design choice in the supplementary material.

The loss function is then defined as the L2 distance between the activations of CLIP on the image $\mathcal{I}$ and sketch $\mathcal{S}$ at a layer ℓ_k:

$$\mathcal{L}_{CLIP}(\mathcal{S}, \mathcal{I}, \ell_k) = \left\| CLIP_{\ell_k}(\mathcal{S}) - CLIP_{\ell_k}(\mathcal{I}) \right\|_2^2. \tag{2}$$

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, as 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.

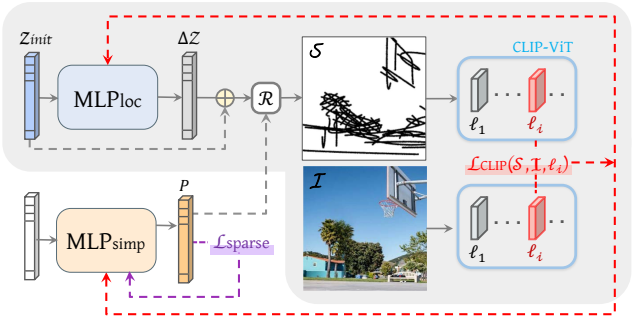

Figure 7. Single sketch generation scheme. In gray, we show our training scheme for producing a single sketch image at a single level of fidelity. In the bottom left we show the additional components used to generate a single sketch at a single level of simplicity.

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 CLIP-ViT model for computing the loss defined in Equation (2). 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. Specifically, in all our examples we train a separate MLP_{loc} using layers $\{\ell_2, \ell_7, \ell_8, \ell_{11}\}$ of CLIP-ViT and set the number of strokes to $n = 64$. Note that it is possible to use the remaining layers to achieve additional fidelity levels (see the supplementary material).

3.3. Simplicity Axis

Given a sketch $\mathcal{S}_k$ at fidelity level k, our goal is to find a set of sketches $\{\mathcal{S}_k^1, ..., \mathcal{S}_k^m\}$ that are visually and conceptually 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 network, denoted as MLP_{simp} (marked in orange), that receives a random-valued vector and is tasked with learning an n-dimensional vector $P = \{p_i\}_{i=1}^n$, where $p_i \in [0, 1]$ represents the probability of the i-th stroke appearing in the rendered sketch. P is passed as an additional input to $\mathcal{R}$ which outputs the simplified sketch $\mathcal{S}_k^j$ in accordance.

To implement the probabilistic-based removal or addition of strokes (which are discrete operations) into our learning framework, we multiply the width of each stroke

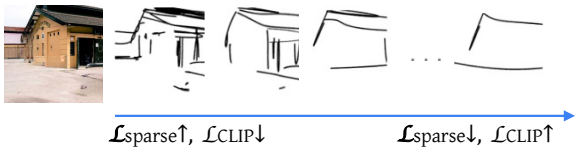

Figure 8. Trade-off between $\mathcal{L}_{sparse}$ and $\mathcal{L}_{CLIP}$. As the sketch becomes sparser, $\mathcal{L}_{sparse}$ obtains lower score. However, the sketch also becomes less recognizable with respect to the input image, resulting in a higher penalty for $\mathcal{L}_{CLIP}$.

z_i by p_i. When rendering the sketch, strokes with a very low probability will be "hidden" due to their small width.

Similar to Mo *et al*. [24], to encourage a sparse representation of the sketch (*i.e*. one with fewer strokes) we minimize the normalized L1 norm of P:

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

To ensure that the resulting sketch still resembles the original input image, we additionally minimize the $\mathcal{L}_{CLIP}$ loss presented in Equation (2), and continue to fine-tune MLP_{loc} during the training of MLP_{simp}. Formally, we minimize the sum:

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

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).

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 the optimization to restore strokes that may have been previously removed. If we were to use direct optimization, the gradients of deleted strokes would remain removed since they were multiplied by a probability of 0.

Here, both MLP networks are simple 3-layer networks with SeLU [14] activations. For MLP_{simp} we append a Sigmoid activation to convert the outputs to probabilities.

Balancing the Losses. Naturally, there is a trade-off between $\mathcal{L}_{CLIP}$ and $\mathcal{L}_{sparse}$, which affects the appearance of the simplified sketch (see Figure 8). We utilize this trade-off to gradually alter the level of simplicity.

Finding a 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:

$$\mathcal{L}_{ratio} = \| \frac{\mathcal{L}_{sparse}}{\mathcal{L}_{CLIP}} - r \|_2^2, \tag{5}$$

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 generating

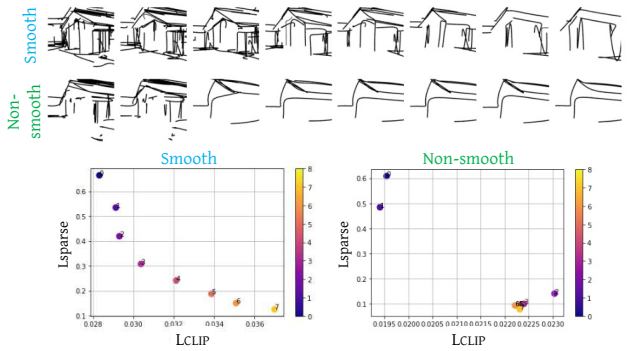

Figure 9. Smooth v.s. non-smooth simplification. In the first row, the simplification appears perceptually smooth, where a consistent change in the degree of abstraction is performed. The second row demonstrates a non-smooth simplification, as there is a visible "jump" between the second and third sketches. These visual patterns are illustrated quantitatively in the corresponding graphs, where each dot in the graph represents a single sketch.

a single simplified sketch $\mathcal{S}_k^j$, is then given by:

$$\mathcal{L}_{simp} = \mathcal{L}_{CLIP} + \mathcal{L}_{sparse} + \mathcal{L}_{ratio}. \tag{6}$$

To achieve the set of gradually simplified sketches $\{\mathcal{S}_k^1, ..., \mathcal{S}_k^m\}$, we define a set of corresponding factors $\{r_k^1, ..., r_k^m\}$ to be applied in Equation (5). The first factor r_k^1, is derived directly from Equation (5), aiming to reproduce the strength of simplification present in $\mathcal{S}_k$:

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

where $\mathcal{L}_{sparse}$ equal to 1 means that no simplification is performed. 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 the set of factors r_k^j, we aim to achieve a *smooth* simplification. By *smooth* we mean that there is no large change perceptually between two consecutive steps. This is illustrated in Figure 9, where the first row provides an example of *smooth* transitions, and 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.

We find that the simplification appears more smooth when $\mathcal{L}_{sparse}$ is exponential with respect to $\mathcal{L}_{CLIP}$. The two graphs at the bottom of Figure 9 describe this observation quantitatively, illustrating the trade-off between $\mathcal{L}_{sparse}$ and $\mathcal{L}_{CLIP}$ for each sketch. The smooth transition 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 right graph.

Given this, we define an exponential function recursively

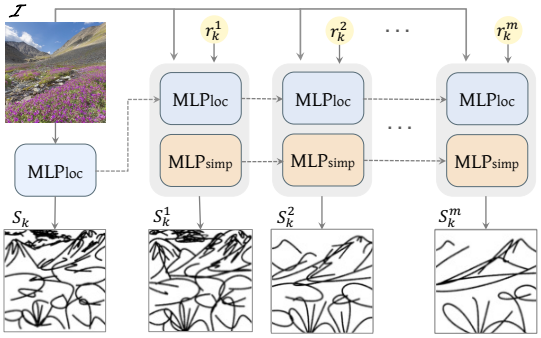

Figure 10. Iterative simplification of the sketch $\mathcal{S}_k$. To produce a simplified sketch $\mathcal{S}_k^j$ we iteratively fine-tune MLP_{loc} (blue) and MLP_{simp} (orange) w.r.t $\mathcal{L}_{ratio}$ loss defined by each r_k^j.

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 define the following set of factors $\{r_k^2, ..r_k^m\}$ we sample the function f_k, where for each k, the sampling step size is set proportional to the strength of the $\mathcal{L}_{CLIP}$ loss 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. This observation aligns well with the Weber-Fechner law [36, 8] which states that human perception is linear with respect to an exponentially-changing signal. An analysis of the factors and additional details regarding our design choices are provided in the supplementary material.

Generating the Simplified Sketches To generate the set of simplified sketches $\{\mathcal{S}_k^1...\mathcal{S}_k^m\}$, we apply the training procedure iteratively, as illustrated in Figure 10. We begin with generating $\mathcal{S}_k^1$ w.r.t r_k^1, by fine-tuning MLP_{loc} and training MLP_{simp} from scratch. After generating $\mathcal{S}_k^1$, we sequentially generate each $\mathcal{S}_k^j$ for $2 \leq j \leq m$ by continuing training both networks for 500 steps and applying $\mathcal{L}_{ratio}$ with the corresponding factor r_k^j.

3.4. Decomposing the Scene

The process described above takes the entire scene as a whole. However, in practice, we separate the scene's foreground subject from the background and sketch each of them independently. We use a pretrained U^2-Net [27] to extract the salient object(s), and then apply a pretrained LaMa [32] inpainting model to recover the missing regions (see Figure 11, top right).

We find that this separation helps in producing more visually pleasing and stable results. When performing object sketching, we additionally compute $\mathcal{L}_{CLIP}$ over layer l_4. This helps in preserving the object's geometry and finer details. On the left part of Figure 11 we demonstrate the artifacts that may occur when scene separation is not applied. For example, over-exaggeration of features of the subject at a low fidelity level, such as the panda's face, or, the object

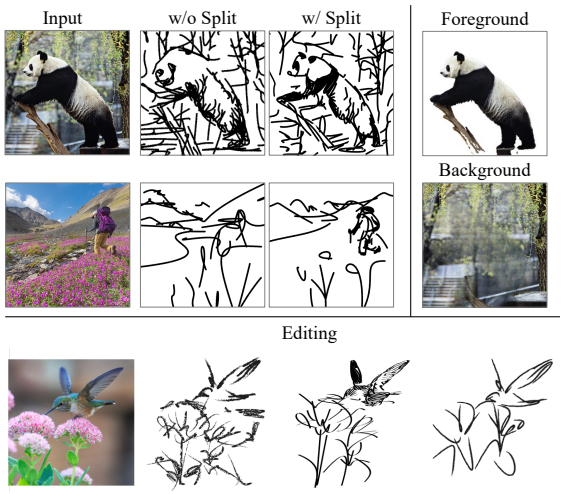

Figure 11. Scene decomposition. Top right – an example of the separation technique. Left – scene sketching results obtained with and without decomposing the scene. Bottom – examples of sketch editing by modifying the style of strokes.

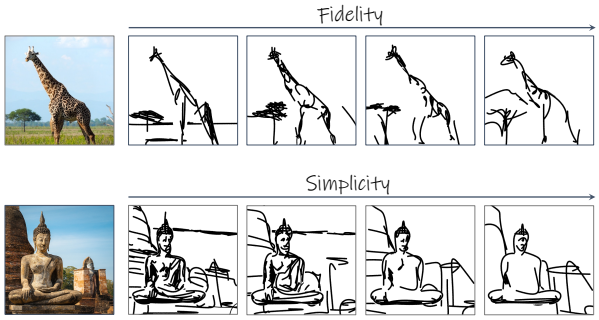

Figure 12. Sketches along the two abstraction axes.

might "blend" into the background. Additionally, this explicit separation provides users with more control over the appearance of the final sketches (Figure 11, bottom right). For example, users can easily edit the vector file by modifying the brush's style or combine the foreground and background sketches at different levels of abstraction.

4. Results

In the following, we demonstrate the performances of our scene sketching technique qualitatively and quantitatively, and provide comparisons to state-of-the-art sketching methods. Further analysis, results, and a user study are provided in the supplementary material.

4.1. Qualitative Evaluation

In Figures 1 and 13 we show sketches at different levels of abstraction on various scenes generated by our method. Notice how it is easy to recognize that the sketches are depicting the same scene even though they vary significantly in their abstraction level.

In Figures 1, 4 and 12 (top) we show sketch abstractions along the *fidelity* axis, where the sketches become less pre-

cise as we move from left to right, while still conveying the semantics of the images (for example the mountains in the background in Figure 1 and the tree and giraffe's body in Figure 12). In Figures 1, 5 and 12 (bottom) we show sketch abstractions along the *simplicity* axis. Our method successfully simplifies the sketches in a smooth manner, while still capturing the core characteristics of the scene. For example, notice how the shape of the Buddha sculpture is preserved across all levels. Observe that these simplifications are achieved *implicitly* using our iterative sketch simplification technique. Please refer to the supplemental file for many more results.

4.2. Comparison with Existing Methods

In Figure 13 we present a comparison to CLIPasso [34]. For a fair comparison, we use our scene decomposition technique to separate the input images into foreground and background and use CLIPasso to sketch each part separately before combining them. Also, since CLIPasso requires a predefined number of strokes as input, we set the number of strokes in CLIPasso to be the same as that learned implicitly by our method. For each image we show two sketches with two different levels of abstraction. As expected, CLIPasso is able to portray objects accurately, as it was designed for this purpose. However, in some cases, such as the sofa, CLIPasso fails to depict the object at a higher abstraction level. This drawback may result from the abstraction being learned from scratch per a given number of strokes, rather than gradually. Additionally, in most cases CLIPasso completely fails to capture the background, even when using many strokes (*e.g.* in the first and fourth rows). Our method captures both the foreground and the background in a visually pleasing manner, thanks to our *learned* simplification approach. For example, our method is able to convey the notion of the buildings in the first row or mountains in the second row with only a small number of simple scribbles. Similarly, our approach successfully depicts the subjects across all scenes.

Note that the main advantage over CLIPasso does not stem only from using the CLIP-ViT architecture, but is rooted in our neural simplification approach that gradually removes strokes by considering the complexity of each image, as well as the tradeoff between fidelity and simplicity in the abstraction process. In Figures 6 and 8 of the supplementary, we show simplifications obtained when using CLIP-ViT with a pre-defined number of strokes on background images (which is equivalent to applying CLIPasso with ViT instead of ResNet). The simplification using CLIPasso does not succeed in the more abstract levels of 16 or fewer strokes. We provide additional sketch results comparing our method and CLIPasso at different abstraction levels in the supplementary.

In Figure 14 we present a comparison with three state-of-the-art methods for scene sketching [41, 16, 5]. On the left, as a simple baseline, we present the edge maps of the input images obtained using XDoG [37]. On the right, we present three sketches produced by our method depicting three representative levels of abstraction.

Figure 13. Comparison to CLIPasso [34]. Note how CLIPasso fails to capture the background in most cases, especially at higher abstraction levels, despite having the same stroke budget.

The sketches produced by UPDG [41] and Chan *et al*. [5] are detailed, closely following the edge maps of the input images (such as the buildings in row 2). These sketches 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 Photo-Sketching [16] 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. Importantly, none of the alternative *scene* sketching approaches can produce sketches with varying abstraction levels, nor can they produce sketches in vector format. We note that in contrast to the methods considered in Figure 14, our method operates per image and requires no training data. However, this comes with the disadvantage of longer running time, taking 6 minutes to produce one sketch on a commercial GPU.

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 axes. To this end, we collected a variety of images spanning five classes of scene imagery: people, urban, nature, indoor, and animals, with seven images for each class. For each image, we created the 4×4 sketch abstraction matrix – resulting in a total of 560 sketches, and created sketches using the different methods presented in Section 4.2. To make a fair comparison with CLIPasso we generated sketches with four levels of abstrac-

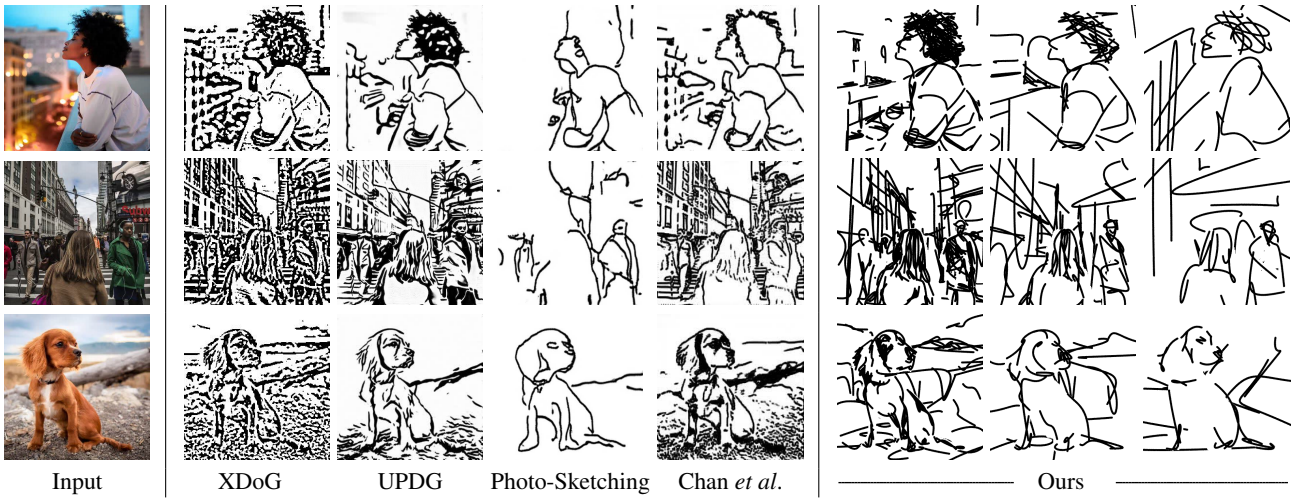

Figure 14. Scene sketching results and comparisons. From left to right are the sketches obtained using XDoG [37], UPDG [41], Photo-Sketching [16], and Chan *et al*. [5]. On the right, are three representative sketches produced by our method depicting three levels of abstraction. Note that UPDG and Chan *et al*. can produce sketches with three different styles, however all the sketches represent a similar level of abstraction. We choose one representative style but provide more style comparisons in the supplementary material.

tion, using the average number of strokes obtained by our method at the four simplicity levels. For UPDG and Chan *et al*., we obtained sketches with three different styles, and averaged the quantitative scores across the three styles, as they represent the same abstraction level. For Photo-Sketching only one level of abstraction and one style is supported.

Fidelity Changes To measure the fidelity level of the generated sketches, we compute the MS-SSIM [35] score between the edge map of each input image (extracted using XDoG) and the corresponding sketch. In Table 1 we show the average resulting scores among all categories, where a higher score indicates a higher similarity. Examining the results matrix of our method, as we move right along the fidelity axis, the scores gradually decrease. This indicates that the sketches become "looser" with respect to the input geometry, as desired. The sketches by UPDG and Chan *et al*. obtained high scores, which is consistent with our observation that their method produces sketches that follow the edges of the input image. The scores for CLIPasso show that the fidelity level of their sketches does not change much across simplification levels and is similar to the fidelity of sketches of our method at the last two levels (the two rightmost columns). This suggests that CLIPasso is not capable of producing large variations of fidelity abstractions.

Sketch Recognizability A key requirement for successful abstraction is that the input scene will remain recognizable in the sketches across different levels of abstraction. To evaluate this, we devise the following recognition experiment on the set of images described above. 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 resulting

Table 1. Comparison of the average MS-SSIM score, computed between the edge map of the input images and generated sketches.

	Ours (Fidelity)				CLIPasso	UPDG	Chan et al.	Photo-Sketch
Simplicity	0.39	0.23	0.22	0.17	0.21	0.57	0.55	0.27
	0.37	0.23	0.21	0.19	0.21			
	0.36	0.22	0.20	0.18	0.15			
	0.34	0.22	0.18	0.14	0.13			

sketches from the different methods. We use a set of 200 class names taken from commonly used image classification and object detection datasets [20, 15] and compute the percent of sketches where at least 2 of the top 5 classes predicted for the input image were also present in the sketch's top 5 classes. We consider the top 5 predicted classes since a scene image naturally contains multiple objects.

Table 2 shows the average recognition rates across all images for each of the described methods. The recognizability of the sketches produced by our method remains relatively consistent across different simplicity and fidelity levels, with a naturally slight decrease as we increase the simplicity level. We do observe a large decrease in the recognition score in the first column. This discrepancy can be attributed to the first fidelity level following the image structure closely, which makes it more difficult to depict the scene with fewer strokes. CLIPasso's fidelity level is most similar to our two rightmost columns (as shown in Table 1). When comparing our recognition rates along these columns to the results of CLIPasso, one can observe that at higher simplicity levels, their method looses the scene's semantics.

Table 2. Recognizability scores, using a CLIP ViT-B/16 model for zero-shot classification on the input image and generated sketches.

	Ours				CLIPasso	UPDG	Chan et al.	Photo-Sketch
	Fidelity							
Simplicity	0.92	1.00	0.95	0.97	0.92	0.87	0.91	0.62
	0.54	0.97	1.00	0.91	0.83			
	0.54	0.93	0.94	0.89	0.70			
	0.44	0.79	0.91	0.85	0.43			

Table 3. Results of our user study. We compare 30 sketches produced by our method to three alternative methods: CLIPasso [34], Chan *et al*. [5], and Photo-Sketching [16]. For each method, we specify the percent of responses that preferred our sketch, the sketch of the alternative method, or found the sketches to be similar in their ability to capture the scene semantics.

Ours v.s. CLIPasso		Ours v.s. Chan *et al*.		Ours v.s. Photo-Sketching	
Ours	84.8%	Ours	52.5%	Ours	75.6%
CLIPasso	7.0%	Chan *et al*.	27.3%	Photo-Sketching	15.8%
Equal	8.2%	Equal	20.2%	Equal	8.6%

User Study As opposed to the fidelity measure, which can be determined by measuring the distance from the edge map of the input scene, validating the recognizability is more challenging. To this end, we also conducted a user study to further validate the findings presented by the CLIP zero-shot classification approach.

The user study examines how well the sketches depict the input scene, considering both the foreground and background. Using 30 images from the set described in Section 4.3, we compared our sketches with three alternative methods: CLIPasso [34], Chan *et al*. [5], and Photo-Sketching [16]. The participants were presented with the input image along with two sketches, one produced by our method and the other by the alternative method (with the sketches presented in random order). An example is provided in Figure 15. The following question was posed to participants: *Which sketch better depicts the image content? In your answer, please relate to: (1) Preservation of both foreground and background. (2) Semantic preservation - i.e., reflecting the meaning of the elements.* Participants could choose between three options: "A", "B", and "A and B at a similar level".

To make a fair comparison, we compared the methods which produce abstract sketches (CLIPasso [34] and Photo-Sketching [16]) with our more abstract sketches (highest abstraction level of our abstraction matrix). Conversely, we compared the sketches of Chan *et al*. [5], which are more detailed and have greater fidelity, to our most detailed results (top left corner of our matrix). We applied CLIPasso using our scene decomposition technique, and the same number of strokes as learned by our method, and for Chan *et al*. we used the contour style sketches. We collected responses from 25 participants for the survey, which contained a total of 90 questions ($2,250$ responses in total).

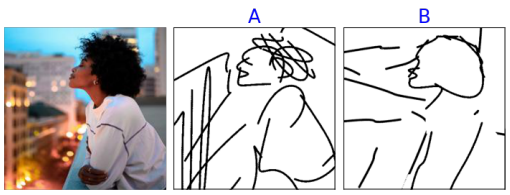

Figure 15. Example of how images were presented to participants in the user study.

The average resulting scores among all participants and images are shown in Section 4.3. In the first, second, and third columns, we show the scores obtained when comparing to CLIPasso [34], Chan *et al*. [5], and Photo-Sketching [16] respectively. Compared to CLIPasso and Photo-Sketching, our method achieved significantly higher rates (84.8% and 75.6% of responses favored our method over the respective alternatives). Conversely, only 7% and 15.8% of responses preferred the results of the alternative method, respectively. Although sketches produced by Chan *et al*. [5] are highly detailed, 52.5% of the responses preferred our sketches, while only 20% considered our sketches and Chan *et al*.'s sketches to be similar.

The results of the user study demonstrate that sketches produced by our method faithfully capture both the foreground and background elements in the scene among varying abstraction levels.

5. Conclusions

We presented a method for performing scene sketching with different types and multiple levels of abstraction. We disentangled 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 and the advantage of using vector representation and scene decomposition to allow for greater artistic control. It is our hope that our work will open the door for further research in the emerging area of computational generation of visual abstractions. Future research could focus on further extending these axes and formulate innovative ideas for controlling visual abstractions.

Acknowledgements We would like to thank Itamar Berger and Nir Ben-Zvi for their early feedback and insightful discussions. This work was supported in part by the Israel Science Foundation under grant No. 2492/20 and grant No. 1390/19.

References

[1] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Trans. Graph.*, 32(4), jul 2013. 1, 3

[2] 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

[3] Irving Biederman and Ginny Ju. Surface versus edge-based determinants of visual recognition. *Cognitive psychology*, 20(1):38–64, 1988. 1

[4] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, pages 679–698, 1986. 3

[5] 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, 7, 8, 9

[6] 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. 4

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

[8] Gustav Theodor Fechner. Elemente der psychophysik. 1860. 6

[9] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *CoRR*, abs/2106.14843, 2021. 3

[10] Songwei Ge, Vedanuj Goswami, Larry Zitnick, and Devi Parikh. Creative sketch generation. In *International Conference on Learning Representations*, 2021. 3

[11] David Ha and Douglas Eck. A neural representation of sketch drawings. In *International Conference on Learning Representations*, 2018. 3

[12] 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

[13] Aaron Hertzmann. Why do line drawings work? a realism hypothesis. *Perception*, 49(4):439–451, mar 2020. 1

[14] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017. 5

[15] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 8

[16] 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, 7, 8, 9

[17] 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

[18] 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

[19] 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

[20] 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. 8

[21] 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

[22] Vijish Madhavan. Artline. https://github.com/vijishmadhavan/ArtLineTechnical-Details, 2020. 3

[23] Daniela Mihai and Jonathon Hare. Learning to draw: Emergent communication through sketching. *Advances in Neural Information Processing Systems*, 34:7153–7166, 2021. 3

[24] Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021)*, 40(4):51:1–51:14, 2021. 5

[25] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning deep sketch abstraction. *CoRR*, abs/1804.04804, 2018. 1, 3

[26] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang 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

[27] 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. 6

[28] 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

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 2, 3

[30] 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

[31] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Learning to sketch with shortcut cycle consistency, 2018. 3

[32] 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. 6

[33] 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

[34] 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. 1, 3, 4, 7, 9

[35] Zhou Wang, Eero P Simoncelli, and Alan C Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003. 8

[36] E H Weber. De pulsu, resorptione, auditu et tactu. 1834. 6

[37] 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, 7, 8

[38] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 3

[39] 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

[40] 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

[41] 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, 7, 8

[42] 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