DOI: 10.1111/cgf.15042
EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Region-Aware Simplification and Stylization of 3D Line Drawings

Vivien Nguyen[1] Matthew Fisher[2] Aaron Hertzmann[2] and Szymon Rusinkiewicz[1]

[1]Princeton University
[2]Adobe Research

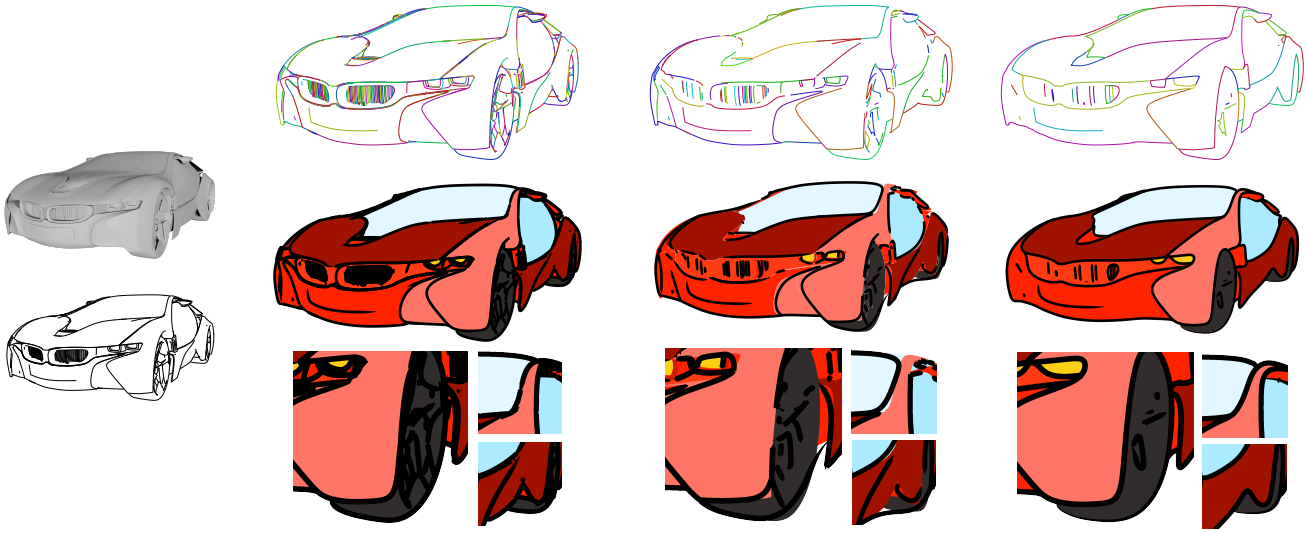


Figure 1: *(a) Original lines extracted from complex 3D models contain unnecessarily complex topology. (b) Attempting to apply stylizations that mimic artistic styles creates unappealing drawings. (c) Naive line drawing simplification reduces line complexity, but arbitrarily introduces gaps between regions, prohibiting automatic region filling and joint line-region stylization. This figure shows simplified lines with the original regions, which have unintended inconsistencies. (d) Our region-aware simplification method utilizes the fact that deleting a line separating two regions equivalently results in merging those regions together. By simultaneously considering the cost of merging regions and removing lines, our method preserves region structure to allow the resulting lines and regions to be consistently stylized together.*

Abstract

Shape-conveying line drawings generated from 3D models normally create closed regions in image space. These lines and regions can be stylized to mimic various artistic styles, but for complex objects, the extracted topology is unnecessarily dense, leading to unappealing and unnatural results under stylization. Prior works typically simplify line drawings without considering the regions between them, and lines and regions are stylized separately, then composited together, resulting in unintended inconsistencies. We present a method for joint simplification of lines and regions simultaneously that penalizes large changes to region structure, while keeping regions closed. This feature enables region stylization that remains consistent with the outline curves and underlying 3D geometry.

CCS Concepts

• ***Computing methodologies*** → *Non-photorealistic rendering; Image manipulation;*

1. Introduction

Line drawings generated from 3D models can reproduce many artistic styles [WS94, GTDS10, EWHS08] and accurately predict human artists' drawings [CSD*09, LNHK20]. Such methods are based on extracting occluding contours [BH19], together with other curves [DeC12]. Considerable amounts of non-photorealistic rendering (NPR) research involves stylization of these curves.

These shape-conveying lines partition the image plane into closed *regions* that are crucial to how the geometry appears. For example, two regions separated by a curve appear to be separate objects or object parts. However, if a gap is added to the curve, then the two regions appear to be a single connected object. Moreover, these regions are crucial to stylization. Indeed, in his classic taxonomy, the artist and psychologist John Willats [Wil97] put lines and regions on equal footing in his grouping of denotational systems. His study catalogues many ways that the handling of regions (as well as of lines and points) distinguishes different artistic styles. Accordingly, many NPR styles specifically focus on stylizing regions, for example, filling each region with a different solid color, or filling 2D regions with strokes [GTDS10, WS94, KWH06, EWHS08, AVR*22].

This paper introduces a method for line drawing simplification that takes region structure into account. Our main observation is that respecting region structure is crucial for simplification, to guarantee that key perceptual and connectivity properties of a shape are preserved.

Simplification is an important step in producing artistic line drawings from complex objects. For complex objects, the precise contours may be quite intricate, with many small curves delineating fine-scale object structures. In small-scale renderings, this produces a dense network of curves. In contrast, a human artist drawing a 3D object or scene often reduces detail and simplifies regions [Chi22]. This artistic trade-off between simplicity and detail reflects both individual style and the context of the drawing. For example, an object depicted at a very small scale, or in the background of a large scene, would typically be depicted with just a few strokes, even if it has many fine details.

Previous methods for simplifying line drawings of 3D models do not consider this region structure when simplifying and deleting strokes, thereby introducing gaps in the silhouette that make it impossible to fill the simplified regions separately from the background, and merging distinct parts. One may stylize filled regions separately from the curve stylization, but this creates mismatches between their shapes, e.g., [EWHS08], that may be objectionable for situations requiring fine-scale precision, such as graphic design and coherent animation.

In this work, we introduce a differentiable, region-aware simplification technique that considers the structures of lines and regions simultaneously. Given a line drawing generated from a 3D model, our method optimizes a loss designed to reduce line drawing density, while preserving the region structure and minimizing the perceptual distance to the original curves. The region structure is represented as a simple planar map, and the region cost terms penalize removing strokes that merge large, perceptually-salient regions. We show that a differentiable loss can produce effective results for this discrete problem.

As we show, there are several benefits to region-aware stylization, as compared to methods that simplify drawings without considering region structure. First, we can preserve object structure that may otherwise be lost if strokes are modified or deleted independent of topology. Second, preserving region structures allows regions to be directly stylized. For example, a method that fills each region with a solid color will not work if curve gaps are created between large, distinct objects and each other or the background. Methods that separately stylize curves and the input geometry create visual inconsistencies between the curve rendering and the object rendering.

Our region-aware simplification and stylization pipeline can create stylized drawings from 3D models that mimic simplified artistic styles, while retaining consistency between the resulting stylized lines and regions. We further demonstrate that considering lines and regions simultaneously is also useful at the stylization stage; our line-driven region stylization pipeline can create drawings with consistent lines and regions, even when the lines have been significantly transformed. The lines resulting from our simplification method still directly correspond to the 3D model that generated them, so the original geometry of the surface can be used when stylizing the drawing.

2. Related Work

Non-photorealistic rendering researchers have long observed the value of simplifying line renderings of 3D models. Winkenbach and Salesin [WS94] noted that "Illustrations can convey information better by omitting extraneous detail, by focusing attention on relevant features, by clarifying and simplifying shapes," among other benefits. Yet, performing effective simplifications remains an open problem. The vast majority of line drawing simplification methods are designed to simplify and process hand-drawn or sketched artwork, which is a different problem from the one this paper addresses. While these related methods can be applied to the same inputs, they must infer the drawing's connectivity from the inputs and may make mistakes in doing so. On the other hand, our method is designed to carefully preserve and simply this structure.

The earliest methods adjust detail in textures [WS94, PHWF01]. Grabli et al. [GDS04] remove strokes based on image-space density, and Shesh et al. [SC08] collapse lines by proximity. These methods do not attempt to preserve structure, which leads to, for example, gaps in silhouettes or broken shapes. Several methods simplify the complexity of contour strokes in order to produce appealing strokes [NM00, IFH*03, BHK14]; these are relatively conservative, and not designed to fundamentally reduce the complexity of the drawing. Some methods [KSG03, NJLM06] simplify the 3D geometry directly, but they are not adaptive to image space density or perception.

Raster-based stroke rendering can adjust the number of strokes by rendering at different scales, e.g. [ST90, LMLH07], but does not produce vector graphics suitable for shape-based stylization or editing. Methods that aim to extract the perceived or intended curve network from human-drawn sketches, either from input

raster images (i.e., line drawing vectorization [FLB16,SBBB20]) or from input vector curves (i.e., stroke consolidation [BTS05, OK11, SSISI16, PvMLV*21, LABS23]) require some degree of line simplification due to the overdrawn nature of human sketches. However, simplification is limited to aggregating nearby lines, rather than simplifying overall structure and regions. Prior work has posed vectorization or line drawing simplification as a trade-off between complexity and accuracy [FLB16] and can adjust scale in various ways [BSM*13]. Several recent methods that generate line drawings [LNHK20,LFHK21,MSSG*21,CDI22,VPB*22] employ learned perceptual metrics as we do, to capture the notion of a drawing's "accuracy". However, these methods do not consider the regions created by the connectivity of the resulting line drawing, nor do they natively allow a user to express constraints on this connectivity.

Several works have demonstrated the importance of junctions and closed regions in the perception of line drawings [YLL*22] and some methods incorporate them in the line simplification process [FLB16,LWH15,PBM18,PCS21]. However, these works were not designed to simplify line drawings from 3D models, where correct connectivity or topology is already known, but the challenge lies in *preserving* this connectivity. [FLB16, LWH15] utilize newly created curves that interpolate or approximate the original lines, leading to potential inconsistencies between the resulting lines and the original 3D model; maintaining this correspondence is necessary for our goal of 3D line drawing stylization. Similarly, running a gap closure method such as [YLL*22] is one way of re-creating closed regions, but there is no guarantee these regions will correspond to the original drawing or underlying 3D geometry without modifying these methods with additional bookkeeping. Our method can output line drawings that have been drastically simplified only by deleting lines and merging regions; we represent these as continuous operations, allowing for differentiable optimization. Our approach is designed to explicitly consider regions throughout the simplification process, allowing for direct stylization of the resulting regions.

Some works, like ours, rely on the concept of merging regions as the basis for line drawing simplification. However, [PBM18] only removes very small regions deemed "unintended" by the original artist, and [PCS21] requires considerable user input. Neither of these address the 3D drawing problem we consider here, and our method aims to simplify drawings in a manner that potentially requires merging comparatively large regions if they are inappropriate at a lower level of detail.

Bernstein et al. [BL15] simplifies vector graphics icons, but requires laborious user authoring in order to capture the semantic aspects of iconography. Eisemann et al. [EWHS08] explore region-based stylization of vector graphics generated from 3D models, but similarly rely on manually tagging or simple classification to determine how regions should be merged together in the final output.

3. Optimization Objectives for Line Drawing Simplification

Our method takes as input a line drawing of a 3D model from a particular viewpoint, represented as a connected set of polylines. We use a combination of the object's occluding mesh contours, surface boundary curves, and creases. We use mesh contours to obtain closed and gap-free regions, which could alternatively be produced by methods such as ConTesse [LBHH23] or quadratic contours [CDHZ23]. We also render a depth map of the object from the same view, which is used as an input feature, described further in Section 3.1.

Given these input lines, our goal is to produce a simplified version of the original drawing, removing redundant strokes and details, and merging regions, while preserving the overall appearance of the drawing. We simplify by optimizing a continuous priority score $z_i \in [0..1]$ for each input curve segment c_i, where a higher z_i represents a higher priority to keep the corresponding line. Given the $Z \equiv \{z_i\}$ values that are output by optimization, a user may then select a simplification threshold τ, so that all lines with $z_i < \tau$ are removed from the drawing.

Our optimization loss function combines three terms:

$$S(Z) = w_1 R(Z) + w_2 P(Z) + w_3 D(Z) \quad (1)$$

The first term, $R(Z)$, is our novel region-preserving loss, and the second $P(Z)$ measures raster perceptual similarity between the input drawing and the generated drawing. The third term, $D(Z)$, measures the density of the drawing in order to encourage removing strokes to make the drawing simpler. Strokes are rendered using z_i values as opacities during the optimization; this makes the objective differentiable using DiffVG [LLMRK20].

As a preprocess, we compute a planar map that represents the topology of the input line drawing (Figure 2), for use in the region loss $R(Z)$. A key step is to compute the set of *regions* bounded by curves in the drawing. To do so, we compute a Constrained Delaunay Triangulation [Che89] of the input line segments, and then define the two triangles as directly connected if they share an edge that is not an input line segment. Then, the set of regions are the connected components of the triangulation. The connectivity of planar map is then directly defined by the triangulation.

In the optimization, we require two curve segments that separate the same regions to have the same priority. That is, if c_a and c_b each separate region i from region j, then $z_a = z_b$, and one variable is used for both. This prevents the outline of a region from being broken into many segments.

The objective terms are described in more detail in the rest of this section.

3.1. Region Merging Loss

We define a loss function $R(Z)$ to measure structural changes between the input and output drawing. This loss discourages merging large regions, and discourages merging regions with very different depth ranges; both types of merges would create more drastic changes to perceived object shape.

Determining merged regions. To compute the loss for a given Z, we must first determine which regions are merged by these values. If a curve separates regions i and j, and $z = 0$ for the curve, then the curve is being deleted, and thus the two regions are merged. Moreover, the transitive closure applies: if i and j are merged, and j and k are merged, then i and k are also merged.

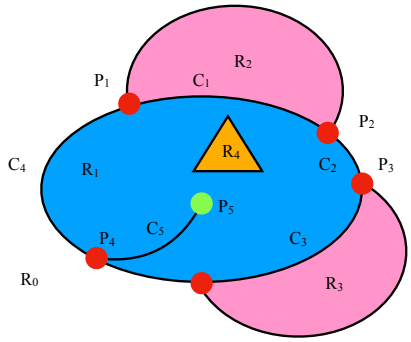


Figure 2: *Planar map topology. The image plane is separated into 5 regions, R_0, R_1, R_2, R_3,and R_4. Curve C_1 (created by junctions P_1 and P_2) is an example of a curve that separates two regions, R_1 and R_2. Curves C_2 and C_4 both separate R_0 and R_1, so we require these curves to have the same priority ($z_2 = z_4$) throughout the optimization process. C_5 is an example of a curve that does not separate two regions; it lies entirely within region R_1. Finally, R_4 is an example of an "island" region, where a region has exactly one neighbor; it is surrounded entirely by another region.*

However, since we use real-valued z's, we must generalize these concepts. Specifically, we introduce a continuous variable $m_{ij} \in [0..1]$, where $m_{ij} = 1$ means that regions i and j are merged, $m_{ij} = 0$ means they are not merged, and in-between values indicate a "soft" merge.

To compute m from Z, we first set $m_{ii} = 1$ for all i. For all pairs of adjacent regions i and j, we set $m_{ij} = m_{ji} = 1 - z$, where z is the score of the curve(s) that separate them, and $m_{ij} = 0$ for non-adjacent regions. We then iterate the following to convergence:

$$\text{for all } i,j: \quad m_{ij}, m_{ji} \leftarrow \max_k (m_{ik} m_{jk}) \tag{2}$$

Our implementation of this procedure is inspired by [Str21].

Loss function. Once m is computed from Z, we can then compute the region loss. The loss specifies penalties for merging each pair of regions i and j:

$$R(Z) = \sum_{ij} m_{ij} A_{ij} \Delta_{ij} \tag{3}$$

with two factors A_{ij} and Δ_{ij}. The first penalizes merging two large regions:

$$A_{ij} = \alpha^{-2} \min(\text{Area}(i), \text{Area}(j)) \tag{4}$$

where $\text{Area}(i)$ is the screen space area of region i, and α is a user-provided simplification scale parameter.

The second factor discourages merging two regions with very different depth ranges, in order to reduce the probability of merging different objects or very separate parts of the same object:

$$\Delta_{ij} = 1 + \max(d_{ij} - d_i, d_{ij} - d_j) \tag{5}$$

where d_i is the difference between the largest and smallest depth value within region i, and d_{ij} is the difference between the largest and smallest depth values across the entireties of regions r_i and r_j. Conceptually, this measures how much the range of depths has expanded by merging regions r_i and r_j.

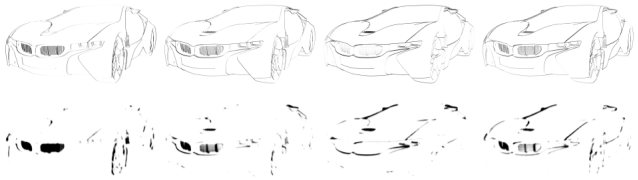


Figure 3: *Image buffers corresponding to each orientation $\theta \in \{-\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}\}$. Each buffer is then blurred and thresholded. Conceptually, this measures the density of strokes in a given direction.*

3.2. Image Similarity

We use a conventional raster-based perceptual similarity function $P(Z)$ to measure the difference between the current drawing and the original drawing. The LPIPS metric [ZIE*18] uses deep network activations to measure the perceptual similarity between two images. In order to measure the similarity between our current drawing and the original, the current values of Z are once again treated as opacities for their corresponding curves, and used to render the current drawing. Then, $P(Z)$ is computed as the LPIPS [ZIE*18] score between the current drawing and the original drawing. We also experimented with a CLIP loss [RKH*21], as used in some previous line drawing abstraction work [VPB*22, VACOS22], but did not find a benefit in our examples.

3.3. Line Density

Finally, the density metric measures the average density of strokes in the image and is the driving force for simplification. We adapt the metric introduced by Grabli et al. [GDS04], which measures density as a function of both scale and orientation.

Specifically, we render four separate image buffers, one for each orientation $\theta \in \{-\frac{\pi}{4}, 0, \frac{\pi}{4}, \frac{\pi}{2}\}$. For a given buffer, each line segment of each curve is rendered with opacity $z_i(1 - 2\phi/\pi)$, where ϕ is the angular difference between the segment's orientation and θ. Each buffer is then blurred with a Gaussian and then thresholded, producing an oriented density image $T(\theta)$. The density cost is computed by summing these buffers together, then taking the mean over all pixels:

$$D(Z) = \sum_{\theta} T(\theta)/N \tag{6}$$

where N is the number of pixels in the image. We visualize these intermediate buffers in Figure 3.

This process has two user-defined parameters, based on desired scale and thickness. First, the Gaussian blur uses standard deviation $\sigma = 0.15\alpha + 0.35$, where α is a user-provided simplification scale. Second, the threshold is performed by $\text{ReLU}(p\ \sigma\sqrt{2\pi}/v - 1)$, where p is an input pixel intensity, and v is a user-provided stroke width parameter. By setting v here to the actual stroke width used to

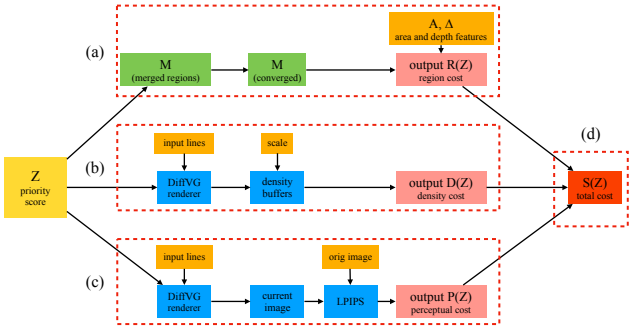


Figure 4: *Computation of loss function $S(Z)$ (Eq (1)). (a) The current values of Z are used to set M, which is iterated according to Eq (2) to represent which regions are currently merged. M is used with area and depth factors (Eq (4) and Eq (5)) to compute the cost of currently merged regions, $M(Z)$. (b) Values of Z are interpreted as stroke opacities to render density buffers (Section 3.3), used to compute the density cost score, $D(Z)$. (c) Values of Z are interpreted as stroke opacities to render the current line drawing, which is compared to the original drawing using LPIPS (Section 3.2) to compute the perceptual similarity score $P(Z)$. (d) The final cost $S(Z)$ is a weighted sum of $R(Z), D(Z)$, and $P(Z)$ (Eq (1)).*

rasterize the lines, we can penalize only parts of the drawing with multiple nearby strokes; single lines do not incur a cost. If v is less than the actual stroke width, then all lines incur some density cost, allowing us to optimize for a sparser overall drawing.

3.4. Optimization

The loss in Equation 1 is optimized for Z using gradient descent and a step size of 0.1 (Figure 4). Gradients are computed using DiffVG [LLMRK20] and PyTorch's auto-differentiation engine. When computing the derivative of $R(Z)$, we treat M as fixed, rather than computing the derivative of M with respect to Z.

For the experiments in this paper, unless stated otherwise, we use the following values for the user-set parameters: $\alpha = 35, w_1 = 10^3, w_2 = 0.1, w_3 = 10$, and $v = 1.0$, along with 80 rounds of optimization and 5 inner optimization steps per round. For final results, we threshold Z at $\tau = 0.1$.

Intermittent thresholding. Removing lines and merging regions are discrete operations, which we approximate here as continuous decisions. In order to better approximate the binary decision of including or not including a stroke in the final rendering, we intermittently binarize a subset of the Z values before proceeding with further iterations of optimization.

The lines in the drawing are sorted by their current priority z_k. Starting from the lowest z_k, z values less than a certain threshold are set to 0. After a line's z value has been altered, we store the line's bounding box, expanded by the simplification scale parameter a. Within a given round, a line's z value can only be forced-set if its bounding box does not intersect with another line that was already thresholded in this round. We then perform a similar set of operations, instead starting from the highest z_k and forcibly setting z_k to 1. We use a lower threshold of 0.1 and an upper threshold of 0.9.

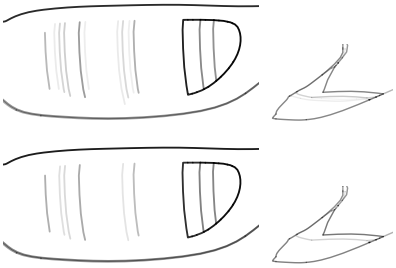


Figure 5: *Lines in collections of dense, parallel lines before (top) and after (bottom) intermediate thresholding step.*

In practice, we observe that this thresholding more effectively reduces the density in collections of parallel lines in the optimization procedure, such as in the car's grille, shown in Figure 5.

3.5. Input and preprocessing steps

Given a 3D model and camera viewpoint, we extract input curves [BH19] using the Blender Freestyle plugin [GTDS10]: occluding mesh contours [DFRS03], boundary edges, and creases. These are represented as a set of polylines, which can be rasterized at arbitrary scale. We render all images at 1920x1080 resolution. We also compute the depth map of the object as an image buffer given the camera view, which is used to compute the depth cost described in Section 3.1. To reduce sampling errors, we render the depth map at twice the resolution (3840x2160).

To efficiently reduce the initial complexity of the drawing as much as possible, we delete all lines shorter than a given length (we use $\sqrt{\alpha}$ as a threshold). We also delete all regions smaller than a given area (we use α as a threshold) by removing all lines that neighbor that region. Lines can only be deleted at this stage if they lie completely within one region, while regions can only be deleted if they are an "island" and are entirely surrounded by another region (see Figure 2). All remaining lines are associated with a corresponding z variable, which is initialized to a random value between 0.1 and 0.2. Since z is used to rasterize the lines, a low initial value here allows the raster-based loss functions we use to better differentiate between lines that are nearly exactly overlapping.

To compute the depth features described in Section 3.1, we normalize the depth map such that the depth values within the object range from 0 to 1. Then, we set the depth of the "background" to the user provided value b. Since merging a region of the object into the background effectively deletes that region, this provides additional control over such an operation. We sample this depth map and, in practice, compute the depth ranges d_i and d_j in Equation 5 using the 25th and 75th percentiles within a region to further reduce sampling errors.

4. Results

Our method can drastically simplify line drawings from complex models (Figure 6). Our simplification procedure yields a set of pri-

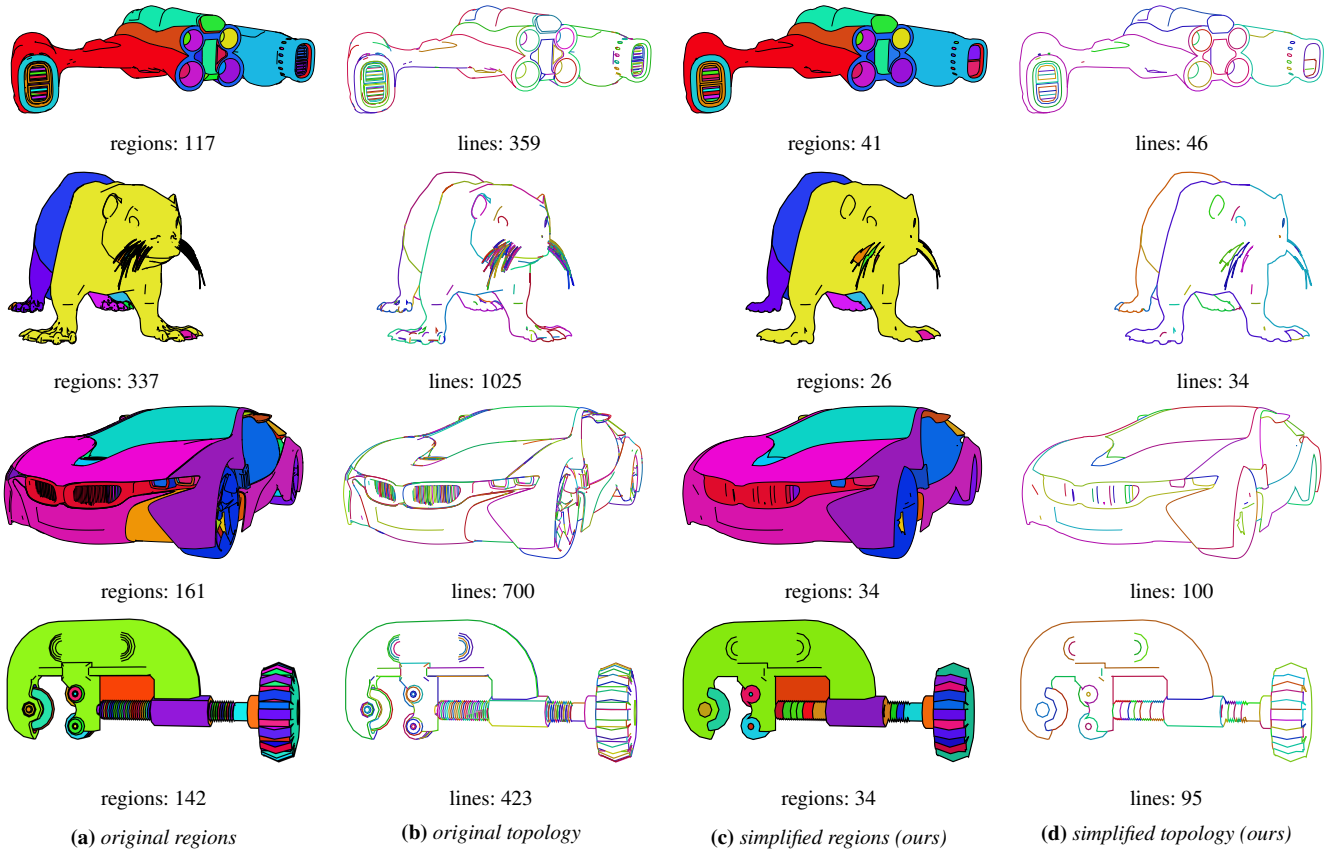


Figure 6: *The curves extracted directly from the 3D model are unnecessarily complex and create many small regions in a drawing, which impede further editing or stylization. (a,b) raw regions and curves from 3D models, show with random colors for visualization. (c,d) Our region-aware simplification method drastically reduces the number of lines in the drawing, improving the visual clarity of the drawing.*

orities Z on each input line; we can apply a final threshold τ on Z to determine the final set of lines to be used. For the results shown, we use $\tau = 0.1$, removing all lines where $Z < \tau$ and yielding a drawing with much simpler topology (Figure 6d) without sacrificing the overall structure of the drawing. In Figure 6a, we assign each region in the original drawing a random color; by tracking region merging with our method, we can directly transfer these colors to our simplified regions (Figure 6c).

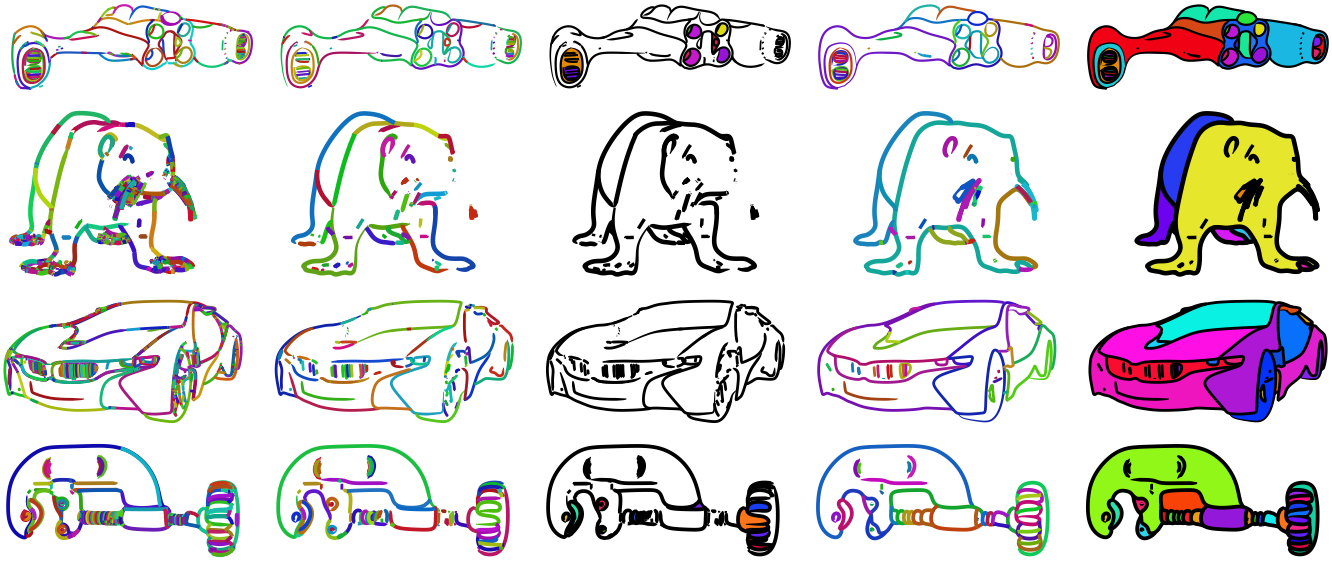
4.1. Stylization

One goal of stylizing line drawings from 3D models is to achieve a more natural appearance. In reality, stylizations frequently further highlight the unnecessary complexity of the input (Figure 7a). Naive methods of simplification that don't incorporate region processing may reduce the number of lines in the drawing, but since gaps are introduced in the drawing, the results do not allow for consistent region stylization (Figure 7c). We produce these naive simplification results by using our own method without region-aware pre-processing or tracking (see Section 4.2 for more details).

In contrast, our region-aware method tracks regions at both the simplification and stylization stage. First, removing small lines often eliminates junctions that contribute to ambiguity about what curves can be chained together, while optimizing for closed regions prevents curves from being broken into short, disconnected sections. As a result, we can produce longer chains of strokes, which behave better under stylization (Figure 7d) compared to the original or naively simplified lines (Figure 7a and Figure 7b).

Second, our method merges together regions from the original drawing, and the resulting regions are still bounded by curves in the simplified drawing. Deforming a curve through stylization implicitly deforms the regions on either side of that curve. By tracking what regions have been merged and continuing to preserve the relationship between simplified regions and simplified lines, we can accurately fill the regions created in the stylized drawing, even when lines have been significantly displaced (Figure 7e).

Simplified drawings with gaps don't have clearly bounded regions that correspond to regions in the original drawing. As a result, region information such as color cannot be transferred directly to the simplify drawing, and stylizations rely on compositing stylized lines with the original regions (Figure 8b). However, this creates an inconsistent drawing: colors drift between regions, extend beyond the silhouette of the drawing, or depict structures of the object that no longer exist.



(a) *original* **(b)** *naive topology* **(c)** *naive regions filled* **(d)** *our topology* **(e)** *our regions filled*

Figure 7: *Results of simplification under simple stylization to create more artistic effects. Additional merging heuristics have been applied to both the original and simplified lines. Naive simplification (i.e., not considering region constraints) arbitrarily introduces gaps between regions.*

Our method allows us to color regions that are consistent with the stylized lines. A simplified line drawing should reasonably also have simplified region styles; for example, an original rendering of the model may utilize the model's surface geometry and directional lighting to shade the drawing, but a simplified shading model would be more suitable for a simplified drawing. With our closed regions, we can approximate the shading per region, allowing for a more simplified appearance that still incorporates the underlying surface geometry, as in Figure 14.

We present results of our method on a variety of objects in Figure 16 and Figure 17. Note that even seemingly simple objects can have complex line drawings, resulting in a dense mess of curves when rendered with stylized strokes. After simplification, stroke stylizations are more effective at creating smooth, natural curves, while our region-aware stylization allows us to transfer colors from the original drawing directly to the newly merged regions.

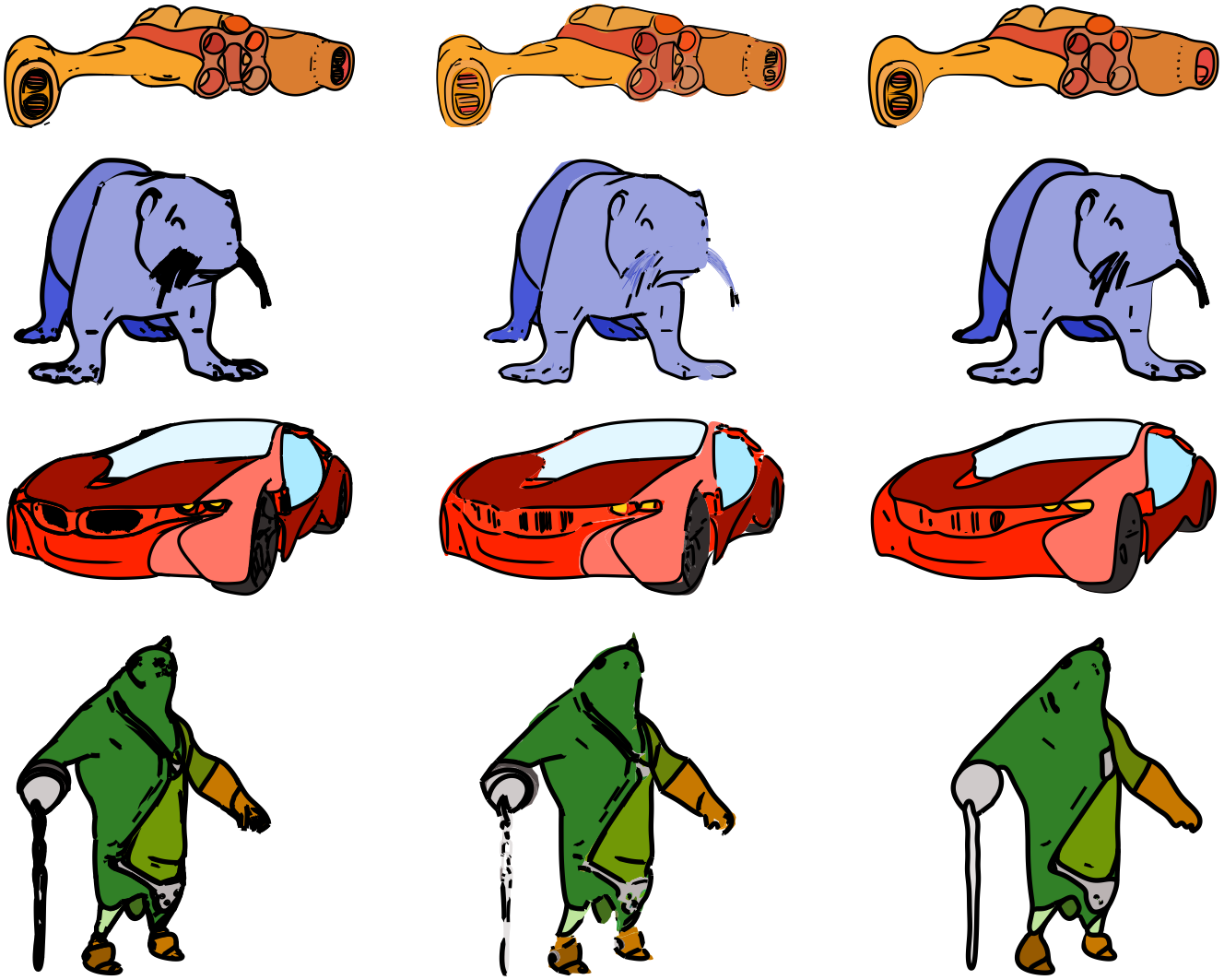
4.2. Ablations

We demonstrate the importance of region-aware simplification by comparing the effects of each loss term for the simplification process. In Figure 9a, we run our optimization algorithm without region-aware pre-processing and without region-aware cost terms. We skip the steps described in Section 3.5 and remove the constraint on Z such that even if curves c_a and c_b both separate regions r_i and r_j, their corresponding z_a and z_b values do not need to be equal (Section 3). We then optimize $S(Z) = D(Z) + P(Z)$, removing any penalties on merging regions. This yields a simplified line drawing that is pixel-wise perceptually similar to the original line drawing, but lines are removed in an un-restricted manner that introduces large gaps between regions.

In the following experiments, we utilize our region pre-processing to restrict Z such that for curves c_a and c_b separating the same regions, $z_a = z_b$. In Figure 9b, we optimize $S(Z) = D(Z) + P(Z)$ with these restrictions on Z. This somewhat improves the resulting connectivity of the simplified drawing, since the removal of any line separating r_i and r_j implies the removal of all lines separating the regions, yielding a higher perceptual cost overall. However, the optimization procedure itself still does not directly optimize for region-preservation, so the simplified drawing still has large gaps.

Utilizing a region cost function that only considers the depth ranges ($R_\Delta(Z) = \sum_{ij} m_{ij}\Delta_{ij}$) then optimizing $S(Z) = D(Z) + R_\Delta(Z)$ still significantly constrains merging regions when those regions correspond to distinct object parts that differ in depth (Figure 9c), but regions that occupy similar depths are likely to be removed, even if they contribute to significant regions in screen space. For example, the depth cost strengthens the boundary (i.e., increases the opacity) of the otter's silhouette, and the occluding contour that separates the front and hind leg. However, maintaining the regions created by the whiskers is given a lower priority, since they occupy a similar depth to the rest of the otter's face. Similarly, the regions corresponding to the dense, parallel lines of the tool in the bottom row have very little depth disparity, but are significant to the drawing overall.

On the other hand, those regions are more likely to be preserved when considering region areas in the cost formulation (setting $R_A(Z) = \sum_{ij} m_{ij}A_{ij}$, then optimizing $S(Z) = D(Z) + R_A(Z)$), but without additional penalties on the relative depths of regions, the optimization procedure is prone to merging regions into the



(a) *original* **(b)** *naive* **(c)** *ours*

Figure 8: *Non-region-aware simplification and stylization methods create drawings that have inconsistent lines and regions, while our region-aware simplification and stylization allows us to directly transfer colors from the original drawing to the simplified and stylized drawing using the remaining topology.*

background resulting in disconnected regions (e.g. the legs of the otter in Figure 9d).

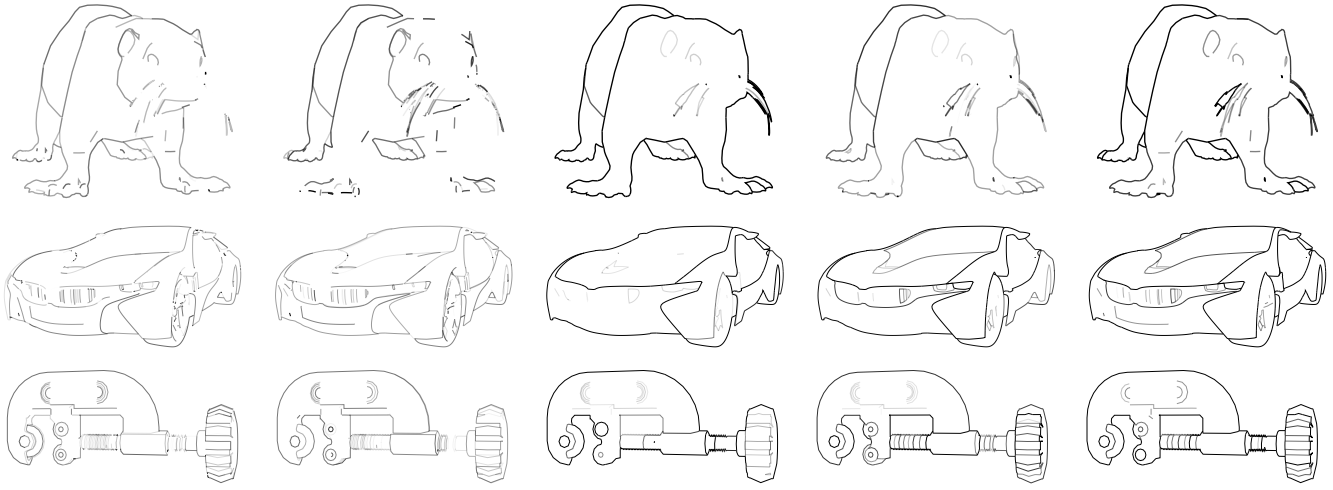
Our proposed score function (Equation 1) incorporates region-aware processing and each of these loss terms in order to produce the simplified drawings in Figure 9.

4.3. Varying User Parameters

We also demonstrate how user provided parameters α and v can both be used to control the desired level of simplification. α is used as a scale parameter in both the density function and the region function. v is used as a stroke width threshold parameter in the density function.

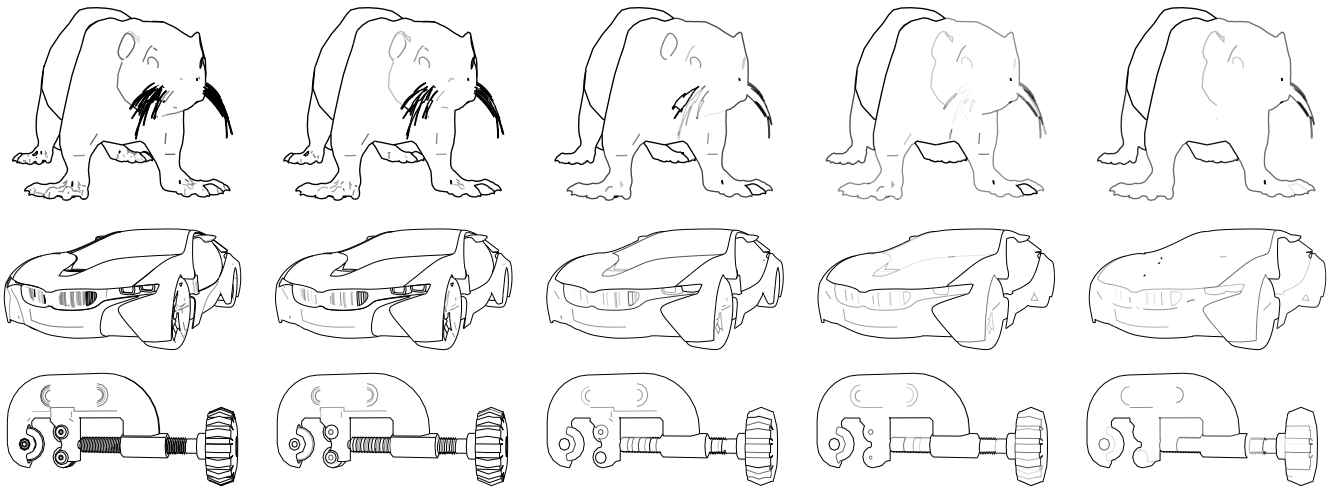
Since the scale parameter α is used in both the density cost $D(Z)$ and the region merging cost $R(Z)$, varying α directly impacts both the simplification of entire regions as well as the resulting spacing in dense regions. For example, in Figure 10, the resulting spacing between the threads of the screw and the ridges on the knob increases as α increases. At the same time, increasing α allows more regions to be merged together, such as in the wheels of the car. Regions are more freely merged into the background (i.e., apparently removing parts of the object altogether), which can be desirable depending on the target resolution that the final drawing will be rendered at.

Varying v has a more subtle effect regarding the overall presence of lines in the drawing. v is used to threshold the density image



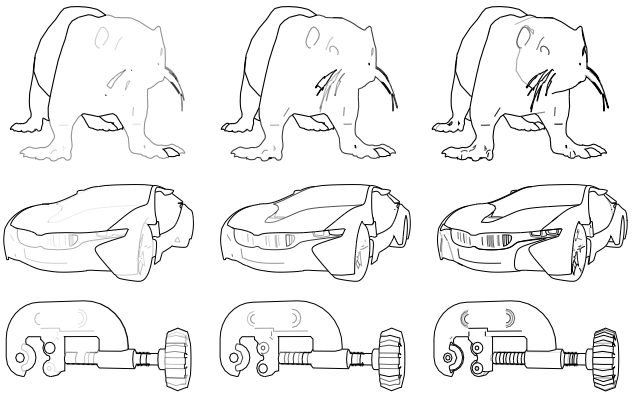
(a) *lpips only (no regions)* **(b)** *lpips only* **(c)** *depth region loss only* **(d)** *area region loss only* **(e)** *all loss terms*

Figure 9: *We demonstrate the impact of our region-aware pre-processing and our various cost terms on restricting the simplification encouraged by $D(Z)$. (a) We remove the restriction on Z where if curves c_a and c_b separate the same regions, $z_a = z_b$. In other words, z_a and z_b are defined as two separate variables. We optimize $S(Z) = D(Z) + P(Z)$. (b) We utilize region pre-processing to restrict Z such that for curves c_a and c_b separating the same regions, $z_a = z_b$, and optimize $S(Z) = D(Z) + P(Z)$. (c) We utilize region pre-processing, and consider only the depth factor for the cost of merging regions. (d) We only consider only the area factor for the cost of merging regions. (e) We use region pre-processing and optimize our proposed combined score function $S(Z) = D(Z) + R(Z) + P(Z)$, as defined in Equation 1.*



(a) $\alpha = 5$ **(b)** $\alpha = 15$ **(c)** $\alpha = 35$ **(d)** $\alpha = 55$ **(e)** $\alpha = 75$

Figure 10: *Results varying the scale parameters α. To more clearly visualize the differences, strokes are rendered with transparency using the Z values for opacity, rather than thresholding. Increasing α both increases the cost of keeping lines and decreases the costs related to merging regions (since terms in $R(Z)$ are inversely scaled by α).*



(a) $v = 0.5$ **(b)** $v = 1$ **(c)** $v = 2$

Figure 11: *Results vary with the stroke width threshold v. Decreasing v reduces the number of strokes preserved in the drawing overall, while increasing v increases the number of strokes preserved in dense regions.*

as described in Section 3.3. Without it, all lines will incur some cost according to $D(Z)$, even if they do not contribute to a drawing's density as defined for our purposes; the optimization procedure may be unnecessarily aggressive in removing such lines. In these experiments, we render strokes with stroke width $= 1$. As seen in Figure 11, setting $v < 1$ optimizes for a drawing with fewer lines overall, even if those lines are not nearby other lines. Setting $v > 1$ is less conservative, allowing *very* nearby lines to mutually persist, while otherwise reducing the density of lines as influenced by α.

4.4. Comparisons With Related Works

We compare our method to related works that aim to create simplified line drawings, either by consolidating raw vector sketches or by generating vector drawings from raster input.

StripMaker. StripMaker [LABS23] is a sketch consolidation method that replaces overdrawn strokes with a single curve corresponding to a perceived intended curve. We preprocess our input vector drawings as described in Section 3.5, i.e., we filter out short lines and small regions. We run additional preprocessing used by StripMaker, based on StrokeStrip [PvMLV*21], then run StripMaker with default parameters and an input stroke width of 2.0.

Integer-Grid Sketch Simplification and Vectorization. Integer-Grid Sketch Simplification and Vectorization [SBBB20] extracts a vector curve network from raster images by parameterizing the drawing with integer grids, which aims to extract clean junctions and group nearby strokes together. We rasterize our vector line drawings at 1920x1080 resolution with a stroke width of 2 pixels to provide raster input to this method.

Virtual Sketching. Virtual Sketching [MSSG*21] learns a mapping from raster to vector space by modeling a virtual pen. Like our method, their differentiable rasterization loss relies on a learned perceptual metric to maintain similarity between the resulting and original line drawings. They use a stroke regularization mechanism to promote longer strokes for simplicity. To provide raster input to their method, we rasterize our vector line drawings at 1920x1080 resolution and with a stroke width of 2 pixels. We run their "vectorization" model (trained on clean line drawings) with default parameters, and select the best visual result out of 10 trials.

Discussion. While these methods can be applied to the same inputs as those used in this paper, they are designed to process hand-drawn artwork. First, these works intend to extract an *intended* drawing given a hand-drawn artwork that may include unintended lines. As a result, they are most successful in reducing complexity in parts of the drawing that are analogous to overdrawn strokes. In contrast, our work intends to reduce the complexity in the overall structure of the drawing, and therefore directly incorporates a scale parameter to influence the degree of simplification; other methods may better replicate the original drawing, but do not apply the same type of simplification (Figure 12).

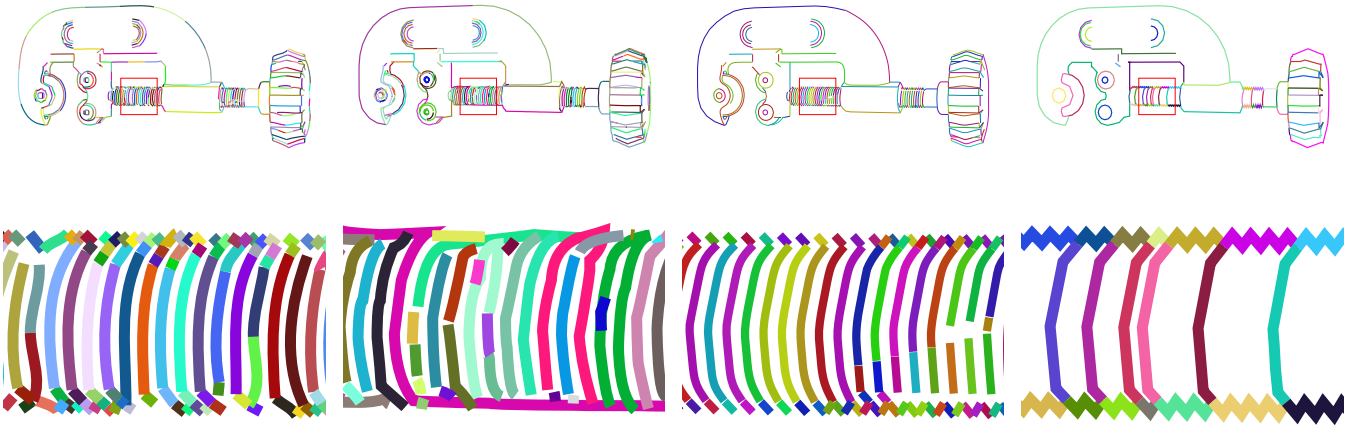
Second, these methods do not consistently produce outputs with closed regions, arbitrarily introducing gaps even where lines appear to intersect (Figure 13a). Junction detection and closing methods can ameliorate this somewhat, but there is inherent ambiguity in predicting correct stroke intersections. Moreover, this post-processing requires additional bookkeeping to find correspondences between the initial regions and those in the resulting drawing. In contrast, our method encodes the initial regions and tracks them throughout the simplification process. We preserve closed regions that directly correspond to the original drawing and do not rely on gap-closure post-processing, which makes applying region-based stylization straightforward (Figure 13b).

5. Limitations and Future Work

We are able to drastically reduce the number of lines and regions in a drawing, but there are some complex topologies that our method is not well-suited to handle. In Figure 15, one potentially desirable simplification would completely delete region R_0, while preserving the remaining 4 regions. Our method does not support "collapsing" regions in this way.

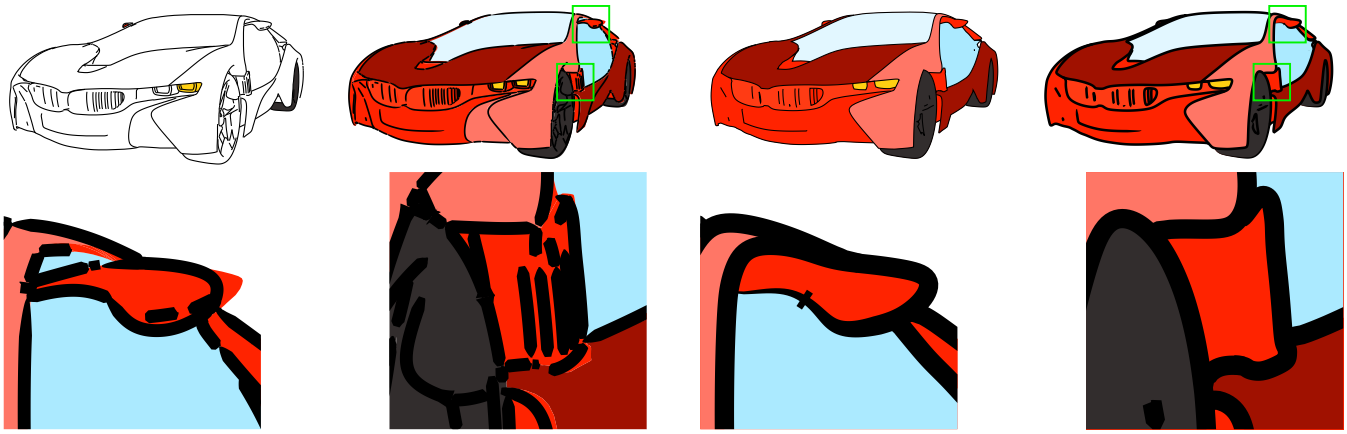
While our method successfully captures the general principles of line drawing simplification, it only utilizes a small set of crafted cost functions. However, the simplified set of lines can be optimized using any differentiable cost function. Related work (e.g. [LNHK20, LFHK21]) have successfully incorporated data-driven score functions into their methods; learning a score function from viewer preferences for simplified drawings could help capture more nuanced aspects of drawing simplification.

Even for initial drawings of lower complexity, such as the spray bottle (Figure 17) presented in this paper, our method (implemented in Python and with little optimization, e.g. spatial hierarchies for more efficient intersection) requires around 2 minutes of preprocessing, and reaches a converged result in less than 7 minutes. For much more complex drawings, such as the car drawing, preprocessing can take up to 10 minutes, and optimization up to 25 minutes. The main bottleneck during optimization is in rendering then differentiating through the density image, which requires ren-



(a) *Integer Grid [SBBB20]* **(b)** *Virtual Sketching [MSSG*21]* **(c)** *StripMaker [LABS23]* **(d)** *our method*

Figure 12: *Compared to related approaches to line drawing simplification, our method can drastically simplify the complex structure of this drawing while maintaining connected lines and preserving closed regions without requiring additional post-processing.*



(a) *StripMaker [LABS23]* **(b)** *ours*

Figure 13: *Unless additional post-processing is applied, StripMaker [LABS23] produces outputs containing gaps between regions such that finding correspondences to the original regions is difficult. Compositing a stylized version of this line drawing with the original regions creates inconsistencies. Our method tracks regions throughout the simplification process, so we can map the original regions onto the simplified ones, and stylize regions and lines simultaneously.*

dering every individual segment of the drawing (since the rendering opacity of the segment depends on its individual orientation).

Finally, our method is largely motivated by the goal of coherent region-line stylization, yet there are a number of artistic styles where the final regions are not perfectly aligned with the stylized lines. However, this is not necessarily contradictory with our goals; such styles should still exhibit some correspondence between the stylized regions and lines, so additional region deformation could be applied as a final step. Another interesting direction would be to add more complex rules around region merging, for example to allow some gaps between regions without considering the two regions merged; this would require defining some notion of a "virtual" underlying region, rather than just the region literally defined by the lines currently present in the drawing.

6. Conclusion

We present a pipeline for region-aware 3D line drawing simplification and stylization. Region simplification and merging is a key effect of line drawing simplification; we incorporate differentiable region merging at the simplification stage to handle the joing simplification of lines and regions. This results in drastically simplified line drawings with fewer overall curves and regions, which further enables more apparent and appealing stylizations. Since regions have been tracked throughout the simplification and stylization stages,

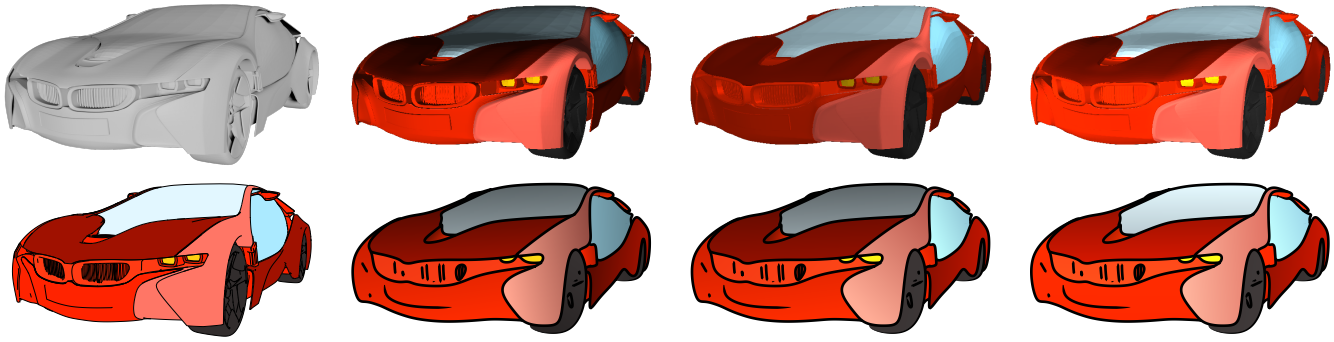


Figure 14: *Carefully tracking regions throughout the simplification and stylization process allows us to utilize 3D information (e.g. surface normals) from the underlying geometry when stylizing the regions. In this case, we can apply different directional lights to the model to achieve different appearances. However, the final stylization is ultimately based on the actual simplified regions in the drawing, resulting in coherent stylization.*

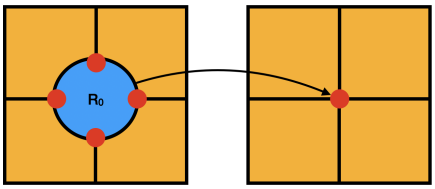


Figure 15: *A potential simplification of this drawing is to remove R_0 while preserving the connectivity of the remaining four regions, which our method is not capable of.*

we can propagate information from the original drawing (e.g. color) or from the original underlying 3D model (e.g. surface normals).

Acknowledgements. We thank Chenxi Liu for help in running comparisons with StripMaker. All models in this paper are licensed through TurboSquid.

References

[AVR*22] ARAÚJO C., VINING N., ROSALES E., GORI G., SHEFFER A.: As-locally-uniform-as-possible reshaping of vector clip-art. *ACM Transaction on Graphics 41*, 4 (2022). doi:https://doi.org/0.1145/3528223.3530098. 2

[BH19] BÉNARD P., HERTZMANN A.: Line drawings from 3D models. *Foundations and Trends in Computer Graphics and Vision 11*, 1-2 (2019), 1–159. doi:10.1561/0600000075. 2, 5

[BHK14] BÉNARD P., HERTZMANN A., KASS M.: Computing smooth surface contours with accurate topology. *ACM Trans. Graph. 33*, 2 (2014), 19:1–19:21. doi:10.1145/2558307. 2

[BL15] BERNSTEIN G. L., LI W.: Lillicon: Using transient widgets to create scale variations of icons. *ACM Trans. Graph. 34*, 4 (jul 2015). URL: https://doi.org/10.1145/2766980, doi:10.1145/2766980. 3

[BSM*13] BERGER I., SHAMIR A., MAHLER M., CARTER E., HODGINS J.: Style and abstraction in portrait sketching. *ACM Trans. Graph. 32*, 4 (jul 2013). URL: https://doi.org/10.1145/2461912.2461964, doi:10.1145/2461912.2461964. 3

[BTS05] BARLA P., THOLLOT J., SILLION F. X.: Geometric Clustering for Line Drawing Simplification. In *Proceedings of the Eurographics Symposium on Rendering* (Konstanz, Germany, June 2005), Deussen O., Keller A., Bala K., Dutré P., Fellner D. W., Spencer S. N., (Eds.), pp. 183–192. URL: https://inria.hal.science/inria-00362893. 3

[CDHZ23] CAPOUELLEZ R., DAI J., HERTZMANN A., ZORIN D.: Algebraic smooth occluding contours. In *ACM SIGGRAPH 2023 Conference Proceedings* (New York, NY, USA, 2023), SIGGRAPH '23, Association for Computing Machinery. URL: https://doi.org/10.1145/3588432.3591547, doi:10.1145/3588432.3591547. 3

[CDI22] CHAN C., DURAND F., ISOLA P.: Learning to generate line drawings that convey geometry and semantics. In *CVPR* (2022). 3

[Che89] CHEW L. P.: Constrained delaunay triangulations. *Algorithmica 4*, 1 (Jun 1989), 97–108. URL: https://doi.org/10.1007/BF01553881, doi:10.1007/BF01553881. 3

[Chi22] CHIE T. Y.: Simplify, draw, repeat, 2022. URL: https://www.youtube.com/watch?v=TevDGBilpNw. 2

[CSD*09] COLE F., SANIK K., DECARLO D., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S., SINGH M.: How well do line drawings depict shape? *ACM Trans. Graph. 28*, 3 (2009), 28:1–28:9. doi:10.1145/1531326.1531334. 2

[DeC12] DECARLO D.: Depicting 3d shape using lines. In *Proc. SPIE* (2012), vol. 8291, pp. 8291 – 8291 – 16. doi:10.1117/12.916463. 2

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (2003), 848–855. doi:10.1145/882262.882354. 5

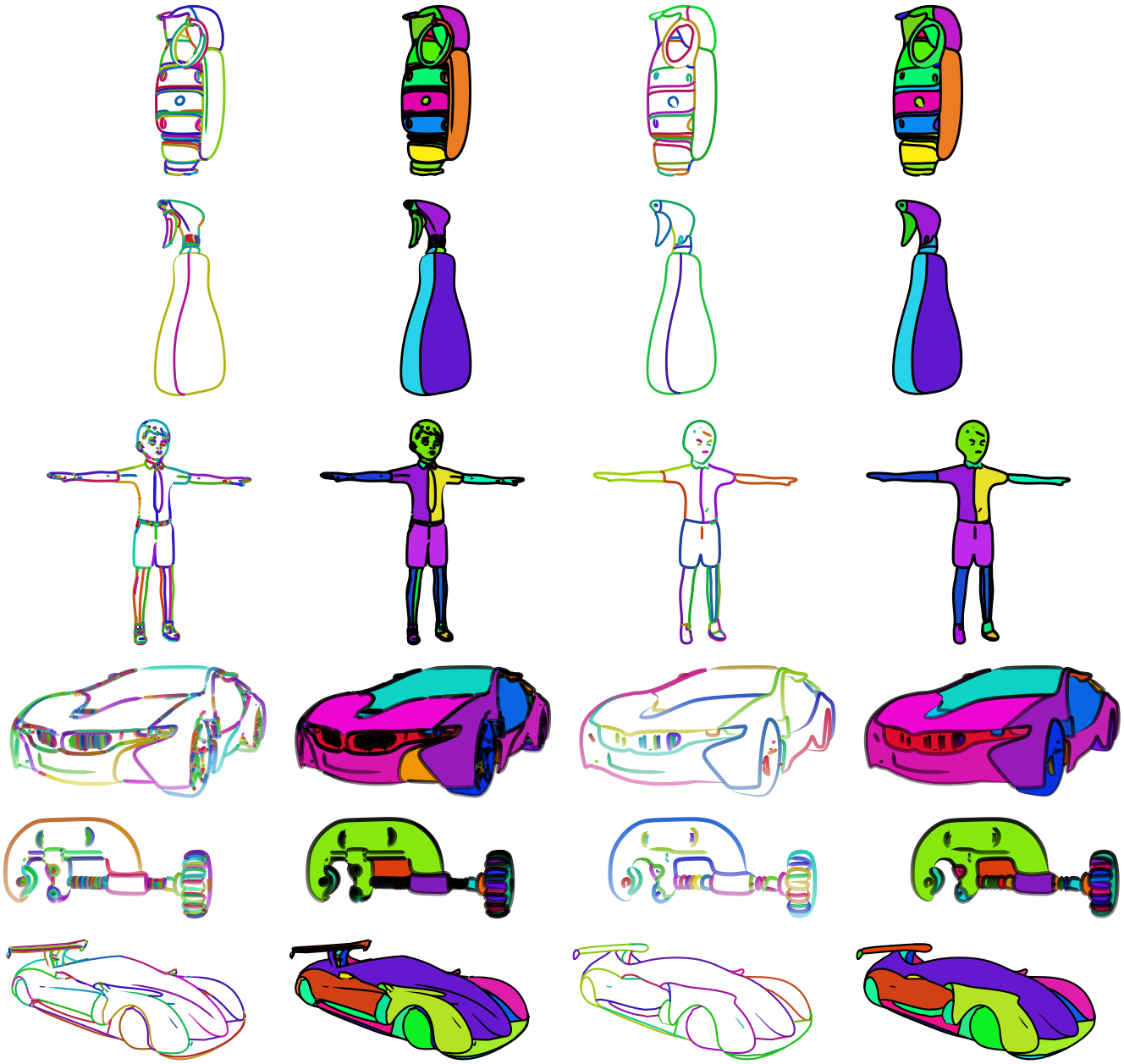
[EWHS08] EISEMANN E., WINNEMÖLLER H., HART J. C., SALESIN D.: Stylized vector art from 3d models with region support. In *Proceedings of the Nineteenth Eurographics Conference on Rendering* (2008), EGSR '08, Eurographics Association, pp. 1199–1207. doi:10.1111/j.1467-8659.2008.01258.x. 2, 3

[FLB16] FAVREAU J.-D., LAFARGE F., BOUSSEAU A.: Fidelity vs. simplicity: A global approach to line drawing vectorization. *ACM Trans. Graph. 35*, 4 (2016), 120:1–120:10. doi:10.1145/2897824.2925946. 3

[GDS04] GRABLI S., DURAND F., SILLION F. X.: Density measure for line-drawing simplification. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on* (2004), pp. 309–318. doi:10.1109/PCCGA.2004.1348362. 2, 4

(a) *original lines* **(b)** *original regions* **(c)** *our lines* **(d)** *our regions*

Figure 16: *Additional results.*



(a) *original lines* **(b)** *original regions* **(c)** *our lines* **(d)** *our regions*

Figure 17: *Additional results.*

[GTDS10] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable rendering of line drawing from 3d scenes. *ACM Trans. Graph. 29*, 2 (2010), 18:1–18:20. doi:10.1145/1731047.1731056. 2, 5

[IFH*03] ISENBERG T., FREUDENBERG B., HALPER N., SCHLECHTWEG S., STROTHOTTE T.: A developer's guide to silhouette algorithms for polygonal models. *IEEE Computer Graphics and Applications 23*, 4 (2003), 28–37. doi:10.1109/MCG.2003.1210862. 2

[KSG03] KIRSANOV D., SANDER P. V., GORTLER S. J.: Simple silhouettes for complex surfaces. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (2003), SGP '03, Eurographics Association, pp. 102–106. 2

[KWH06] KOLLIOPOULOS A., WANG J. M., HERTZMANN A.: Segmentation-based 3d artistic rendering. In *Proceedings of the 17th Eurographics Conference on Rendering Techniques* (2006), EGSR '06, Eurographics Association, pp. 361–370. doi:10.2312/EGWR/EGSR06/361-370. 2

[LABS23] LIU C., AOKI T., BESSMELTSEV M., SHEFFER A.: Stripmaker: Perception-driven learned vector sketch consolidation. *ACM Trans. Graph. 42*, 4 (jul 2023). URL: https://doi.org/10.1145/3592130, doi:10.1145/3592130. 3, 10, 11

[LBHH23] LIU C., BÉNARD P., HERTZMANN A., HOSHYARI S.: Contesse: Accurate occluding contours for subdivision surfaces. *ACM Trans. Graph. 42*, 1 (Feb. 2023). URL: https://doi.org/10.1145/3544778, doi:10.1145/3544778. 3

[LFHK21] LIU D., FISHER M., HERTZMANN A., KALOGERAKIS E.: Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021). 3, 10

[LLMRK20] LI T.-M., LUKÁČ M., MICHAËL G., RAGAN-KELLEY J.: Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 39*, 6 (2020), 193:1–193:15. 3, 5

[LMLH07] LEE Y., MARKOSIAN L., LEE S., HUGHES J. F.: Line drawings via abstracted shading. *ACM Trans. Graph. 26*, 3 (2007). doi:10.1145/1276377.1276400. 2

[LNHK20] LIU D., NABAIL M., HERTZMANN A., KALOGERAKIS E.: Neural contours: Learning to draw lines from 3d shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2020). 2, 3, 10

[LWH15] LIU X., WONG T.-T., HENG P.-A.: Closure-aware sketch simplification. *ACM Transactions on Graphics (SIGGRAPH Asia 2015 issue) 34*, 6 (November 2015), 168:1–168:10. 3

[MSSG*21] MO H., SIMO-SERRA E., GAO C., ZOU C., WANG R.: General virtual sketching framework for vector line art. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021) 40*, 4 (2021), 51:1–51:14. 3, 10, 11

[NJLM06] NI A., JEONG K., LEE S., MARKOSIAN L.: Multi-scale line drawings from 3d meshes. In *Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games* (2006), I3D '06, ACM, pp. 133–137. doi:10.1145/1111411.1111435. 2

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes: A hybrid approach. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering* (2000), NPAR '00, ACM, pp. 31–37. doi:10.1145/340916.340920. 2

[OK11] ORBAY G., KARA L. B.: Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (may 2011), 694–708. URL: https://doi.org/10.1109/TVCG.2010.105, doi:10.1109/TVCG.2010.105. 3

[PBM18] PARAKKAT A. D., BONDI PUNDARIKAKSHA U., MUTHUGANAPATHY R.: A delaunay triangulation based approach for cleaning rough sketches. *Computers & Graphics 74* (2018), 171–181. URL: https://www.sciencedirect.com/science/article/pii/S0097849318300761, doi:https://doi.org/10.1016/j.cag.2018.05.011. 3

[PCS21] PARAKKAT A. D., CANI M.-P. R., SINGH K.: Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2021), CHI '21, Association for Computing Machinery. URL: https://doi.org/10.1145/3411764.3445215, doi:10.1145/3411764.3445215. 3

[PHWF01] PRAUN E., HOPPE H., WEBB M., FINKELSTEIN A.: Real-time hatching. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), SIGGRAPH '01, ACM, pp. 581–. doi:10.1145/383259.383328. 2

[PvMLV*21] PAGUREK VAN MOSSEL D., LIU C., VINING N., BESSMELTSEV M., SHEFFER A.: Strokestrip: Joint parameterization and fitting of stroke clusters. *ACM Transactions on Graphics 40*, 4 (2021). doi:10.1145/3450626.3459777. 3, 10

[RKH*21] RADFORD A., KIM J. W., HALLACY C., RAMESH A., GOH G., AGARWAL S., SASTRY G., ASKELL A., MISHKIN P., CLARK J., KRUEGER G., SUTSKEVER I.: Learning transferable visual models from natural language supervision. *CoRR abs/2103.00020* (2021). URL: https://arxiv.org/abs/2103.00020, arXiv:2103.00020. 4

[SBBB20] STANKO T., BESSMELTSEV M., BOMMES D., BOUSSEAU A.: Integer-grid sketch simplification and vectorization. *Computer Graphics Forum (Proc. SGP) 39*, 5 (7 2020). doi:10.1111/cgf.14075. 3, 10, 11

[SC08] SHESH A., CHEN B.: Efficient and dynamic simplification of line drawings. *Comput. Graph. Forum 27*, 2 (2008), 537–545. URL: http://dblp.uni-trier.de/db/journals/cgf/cgf27.html#SheshC08. 2

[SSISI16] SIMO-SERRA E., IIZUKA S., SASAKI K., ISHIKAWA H.: Learning to simplify: fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph. 35*, 4 (jul 2016). URL: https://doi.org/10.1145/2897824.2925972, doi:10.1145/2897824.2925972. 3

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques* (1990), SIGGRAPH '90, ACM, pp. 197–206. doi:10.1145/97879.97901. 2

[Str21] STRAUBINGER T.: Learning to reduce graphs with differentiable graph merging, 2021. URL: https://timstr.github.io/machine_learning/graph_merging.html. 4

[VACOS22] VINKER Y., ALALUF Y., COHEN-OR D., SHAMIR A.: Clipascene: Scene sketching with different types and levels of abstraction, 2022. URL: https://arxiv.org/abs/2211.17256, doi:10.48550/ARXIV.2211.17256. 4

[VPB*22] VINKER Y., PAJOUHESHGAR E., BO J. Y., BACHMANN R. C., BERMANO A. H., COHEN-OR D., ZAMIR A., SHAMIR A.: Clipasso: Semantically-aware object sketching, 2022. arXiv:2202.05822. 3, 4

[Wil97] WILLATS J.: *Art and Representation: New Princples in the Analysis of Pictures*. Princeton University Press, 1997. 2

[WS94] WINKENBACH G., SALESIN D. H.: Computer-generated pen-and-ink illustration. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), SIGGRAPH '94, ACM, pp. 91–100. doi:10.1145/192161.192184. 2

[YLL*22] YIN J., LIU C., LIN R., VINING N., RHODIN H., SHEFFER A.: Detecting viewer-perceived intended vector sketch connectivity. *ACM Transactions on Graphics 41* (2022). 3

[ZIE*18] ZHANG R., ISOLA P., EFROS A. A., SHECHTMAN E., WANG O.: The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR* (2018). 4