DRAW WHAT MATTERS: DECOUPLING STROKE CAPACITY FROM SKETCH COMPLEXITY IN PHOTO-TO-VECTOR GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A good sketch is an act of abstraction: it keeps what makes a subject recognizable and drops the rest. Most photo-to-vector methods, however, fix a stroke budget and render every stroke, so a single number sets both the *capacity* of a sketch, how much it can express, and its *complexity*, how much it must show. Too few strokes lose recognizable parts, too many add clutter, and the right number changes from one image to the next. We present AdaSketch, which decouples the two: the budget only bounds capacity, and each stroke has to earn its visibility. AdaSketch draws one closed contour that secures the silhouette, plus a pool of detail strokes with learnable opacity. Which details survive is decided by coverage rather than by count: salient interior points must be covered (faithfulness), each by as few strokes as possible (conciseness), and a continuous parameter sets how much image-relevant detail to keep. A progressive contour-to-detail schedule settles where to draw before deciding which strokes to keep. Under the same budget, AdaSketch outperforms five existing methods on all five semantic, structural and perceptual metrics, adapts the rendered complexity to both content and the chosen abstraction level, and extends to target-specific sketching in multi-object scenes.

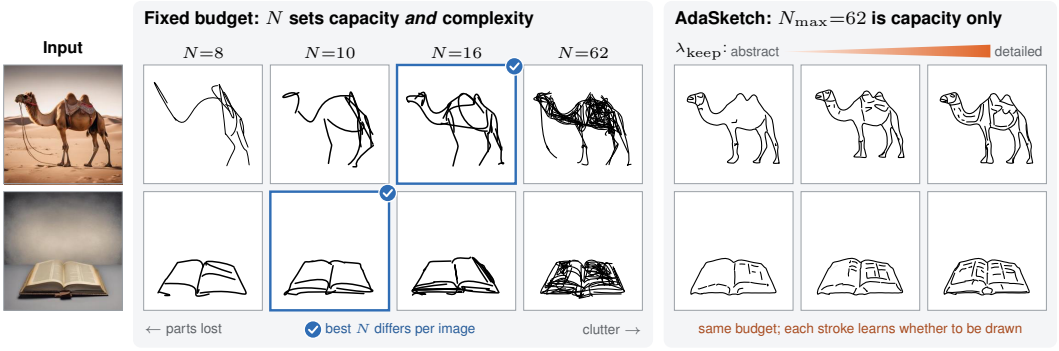


Figure 1: A fixed stroke budget couples capacity and complexity. Left: a method that renders all N strokes loses recognizable parts when N is small and clutters the sketch when N is large, and the best N (blue) differs from image to image. Right: AdaSketch treats $N_{\max} = 62$ as capacity only. Each detail stroke learns whether to be drawn, and a single parameter λ_{keep} moves from abstract to detailed sketches without changing the budget.

1 INTRODUCTION

Sketching is visual abstraction: a few strokes convey an object by keeping what identifies it and omitting the rest (Muhammad et al., 2018; Vinker et al., 2022; Inoue et al., 2019). For photo-to-vector sketch generation, this means representing a photograph with a compact set of vector strokes that is both *faithful*, preserving the silhouette and the internal structures that make the subject

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

recognizable, and *concise*, containing no stroke that says nothing new. How many strokes this takes depends on the content: an open book needs far more internal strokes than a camel (Figure 1).

Early approaches (Li et al., 2019a; Yi et al., 2019; Song et al., 2018; Chan et al., 2022; Li et al., 2019b) learn mappings from photographs to contour drawings, artistic sketches or line drawings, without explicit control over which strokes are drawn. Vector methods optimize strokes directly through differentiable rasterization (Li et al., 2020), guided by vision–language and diffusion priors (Vinker et al., 2022; Xing et al., 2023; Radford et al., 2021; Poole et al., 2023; Zhang et al., 2025; Frans et al., 2022; Jain et al., 2023; Xing et al., 2024), or learn to generate them (Arar et al., 2025). These methods fix a stroke budget N and render all of it. N therefore plays two roles at once: it sets the *capacity* of the sketch, how much it can express, and its *complexity*, how much it must show. The two cannot be tuned separately. Raising N buys capacity at the price of redundant strokes and clutter; lowering it buys conciseness at the price of recognizable parts (Figure 1). Nor does any single N suit every image. Methods that abstract a sketch by removing strokes (Muhammad et al., 2018; Vinker et al., 2023) leave this coupling in place: removal is driven by a global sparsity or stroke-count objective, so it asks *how few* strokes to keep rather than *which* strokes are redundant.

We argue that the budget should bound what a sketch *can* say, not dictate what it *must* say. AdaSketch treats N as capacity only and lets every detail stroke earn its visibility: each stroke carries a learnable opacity, optimized jointly with its geometry under diffusion guidance (Poole et al., 2023), and strokes that add nothing fade out. The number of rendered strokes then follows from the image and from one continuous abstraction parameter λ_{keep}; it is not an input.

Turning this principle into a working method raises three questions. *What must always be drawn?* The silhouette carries most of an object's identity, whereas internal details are optional and depend on the content. AdaSketch therefore separates one closed contour, a chain of Bézier segments fitted to the object boundary and kept fully opaque, from a pool of independent detail strokes, each with a learnable opacity, that compete for visibility. *How is visibility decided without a target count?* We cast it as coverage. Guidance points are sampled from a detail map, CLIP attention restricted to the object interior, so they concentrate where internal structure matters. A stroke contributes to a point in proportion to its opacity times its proximity, so it counts only if it is both near and visible. Every point must receive enough total contribution, which protects faithfulness, while overlapping contributions from several strokes are penalized, so redundant strokes lose opacity and fade out. Together, the two terms relax an exact-cover problem, so the number of visible strokes follows from the structure to be covered rather than from a budget. A keep term, weighted by each stroke's image relevance and scaled by λ_{keep}, sets how much detail survives, so abstraction is specified as a level of detail rather than a number of strokes. *When should a stroke be switched off?* Deciding too early discards strokes before they have moved to where they would be useful. A progressive contour-to-detail schedule first optimizes and freezes the contour, then refines the detail strokes, and only then pushes their opacities towards binary values: it decides *where to draw* before *which strokes to keep*.

Under the same budget as prior methods, AdaSketch obtains the best scores on all five metrics, and varying λ_{keep} moves smoothly from concise abstraction to rich detail without changing the budget. Our main contributions are summarized as follows:

- We decouple stroke capacity from sketch complexity in photo-to-vector generation: the budget bounds capacity, and learnable per-stroke visibility decides what is drawn.
- We introduce a contour–detail representation with a coverage-based selection objective that decides visibility without a target count and exposes abstraction as a continuous parameter.
- We propose a progressive contour-to-detail optimization that settles geometry before visibility, and show state-of-the-art faithfulness under a fixed budget, controllable abstraction, and target-specific sketching.

2 Related Work

Photo-to-vector sketch generation. Early methods learn mappings from photographs to contour drawings, line drawings or category-specific sketches (Li et al., 2019a; Yi et al., 2019; Song et al., 2018; Chan et al., 2022; Inoue et al., 2019; Li et al., 2019b). Differentiable rasterization (Li et al., 2020) allows vector strokes to be optimized directly. CLIP (Radford et al., 2021) guidance drives object sketching in CLIPasso (Vinker et al., 2022), and text-to-drawing in CLIPDraw (Frans et al.,

2022); diffusion priors, typically through score distillation (Poole et al., 2023), drive text-to-vector generation (Jain et al., 2023; Xing et al., 2024; 2023) and image-conditioned sketching (Zhang et al., 2025; Magne et al., 2026). SwiftSketch (Arar et al., 2025) instead trains a diffusion model to generate vector strokes directly. In all of these methods the stroke budget sets the level of abstraction and every allocated stroke is rendered, so capacity and complexity cannot be tuned separately. AdaSketch keeps score-distillation guidance but lets the budget bound capacity only.

Sketch abstraction and stroke selection. Another line of work reduces an existing drawing to its essential strokes, using closure cues or learned stroke importance (Liu et al., 2015; Ogawa et al., 2016), or a stroke-removal policy that trades recognizability against stroke count (Muhammad et al., 2018). These methods start from a finished drawing rather than deciding which strokes to generate from a photograph. Closest to our setting, the simplification axis of CLIPascene (Vinker et al., 2023) learns a keep probability for each stroke, driven by a global sparsity objective whose target ratio defines each abstraction level. AdaSketch instead decides visibility locally, by whether a stroke is needed to cover salient structure that no other stroke covers, and never specifies how many strokes to keep; abstraction is set as a level of detail through λ_{keep}.

3 METHODOLOGY

Given an input image I, AdaSketch aims to represent its essential visual content with a concise set of vector strokes while preserving recognizable structure. Our key idea is to decouple the available stroke capacity from the complexity of the rendered sketch: rather than requiring every initialized stroke to appear in the final rendering, AdaSketch maintains a stable contour scaffold and adaptively selects informative detail strokes through learnable visibility. As illustrated in Figure 2, semantic and spatial priors are used to initialize the strokes, after which AdaSketch adaptively selects detail strokes according to their contribution to informative internal structures, with a user-specified continuous abstraction parameter controlling the degree of detail preservation. A progressive contour-to-detail optimization strategy further stabilizes this selection process by establishing the global structure before making strong stroke visibility decisions.

3.1 ADAPTIVE STROKE REPRESENTATION

We decompose a vector sketch into two complementary components,

$$S = S_c \cup S_d, \qquad S_c = \{S_{c,i}\}_{i=1}^{N_c}, \qquad S_d = \{S_{d,i}\}_{i=1}^{N_d}, \tag{1}$$

where S denotes the complete vector sketch. The contour component S_c consists of N_c connected cubic Bézier segments, which are joined end-to-end to form a single closed contour curve. The detail component S_d consists of N_d independent strokes, each represented by a cubic Bézier segment.

The central distinction of AdaSketch is that the initialized detail capacity does not dictate how many detail strokes must remain visible in the final rendering. For each detail stroke $S_{d,i}$, we introduce a learnable opacity logit $a_i \in \mathbb{R}$ and parameterize its opacity as

$$\alpha_i = \text{sigmoid}\left(\frac{a_i}{T}\right), \qquad \alpha_i \in [0, 1], \tag{2}$$

where T is temperature controlling the softness of the opacity. The opacity α_i determines the visible contribution of stroke $S_{d,i}$ to the rendered sketch: strokes with large opacity remain visible, whereas strokes whose opacity approaches zero are effectively suppressed. Therefore, detail visibility can be optimized jointly with stroke geometry instead of being fixed by the initial segment budget.

AdaSketch further uses λ_{keep} as a user-specified continuous abstraction control parameter. Rather than specifying a target number of strokes, λ_{keep} controls the degree to which informative internal structures are preserved during detail stroke selection. Increasing λ_{keep} encourages the retention of more informative details, whereas decreasing it allows more detail strokes to be suppressed. Consequently, the complexity of the rendered sketch adapts to both the input content and the desired abstraction level, while the initialized stroke set only defines the available representational capacity.

3.2 SEMANTIC SPATIAL GUIDANCE AND STROKE INITIALIZATION

To provide spatial and semantic guidance for the two stroke groups, we derive guidance maps from the input image I. We first generate a caption using BLIP-2 (Li et al., 2023) and extract the target

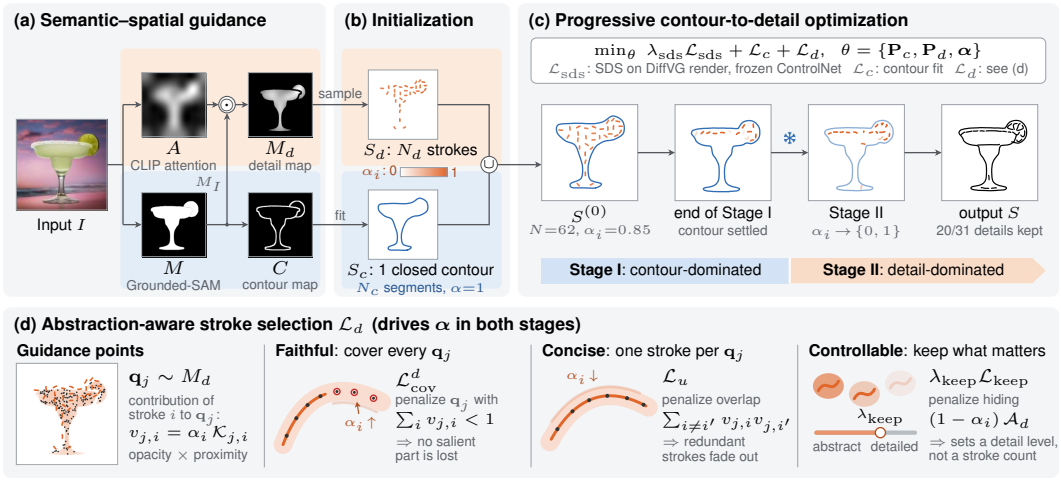


Figure 2: Overview of AdaSketch. (a) Grounded-SAM, prompted with the BLIP-2 object name, gives a mask M, from which we derive a contour map C and an eroded interior M_I; CLIP attention A restricted to M_I gives the detail map M_d. (b) The budget $N = N_c + N_d$ sets capacity only: one closed contour S_c is fitted along C and always rendered, while N_d detail strokes S_d are sampled from M_d, each with a learnable opacity α_i. (c) Geometry and opacity are optimized jointly on the DiffVG render. Stage I settles the contour, which is then frozen; Stage II refines the detail strokes and binarizes their opacities. Only visible strokes are rendered (here 20 of 31 detail strokes). (d) $\mathcal{L}_d$ decides which strokes stay visible, using guidance points $\mathbf{q}_j \sim M_d$ and visible contributions $v_{j,i} = \alpha_i \mathcal{K}(S_{d,i}, \mathbf{q}_j)$: every point must be covered (faithful), by as few strokes as possible (concise), while λ_{keep} sets how strongly image-relevant strokes are kept (shading indicates relevance $\mathcal{A}_d$).

object name from the caption using a lightweight noun-phrase parser. The target name is then used as the prompt for Grounded-SAM (Liu et al., 2024; Kirillov et al., 2023; Ren et al., 2024), which produces a binary foreground mask M. Based on M and a semantic attention map A, we construct

$$C = \text{Norm}\left(\text{Dilate}(M) - \text{Erode}(M)\right), \ M_I = \text{Erode}(M), \ M_d = \text{Norm}\left(A \odot M_I\right), \tag{3}$$

where dilation and erosion are morphological operations, $\odot$ denotes element-wise multiplication, and $\text{Norm}(\cdot)$ normalizes map values. The contour map C provides boundary information, M_I defines an eroded foreground interior, and M_d restricts semantic attention to interior regions. S_c is initialized along C as a closed chain of connected Bézier segments, which provides a structural scaffold near the object boundary. S_d are initialized as independent cubic Bézier curves whose initial locations are sampled according to M_d, concentrating detail capacity on internal details.

3.3 Abstraction-Aware Stroke Selection

Given the initialized detail strokes, AdaSketch must decide which of them to draw. We cast this decision as a covering problem. Salient interior structure is represented by guidance points sampled from M_d, and each detail stroke covers the points near it. A good set of visible strokes covers every point *exactly once*: at least once, so that no salient structure is lost (faithfulness), and at most once, so that no structure is drawn twice (conciseness). Selecting such a subset is an exact-cover problem, in which the number of selected strokes follows from the structure to be covered rather than from a budget. We optimize a continuous relaxation: the binary selection of stroke i becomes its opacity $\alpha_i \in [0, 1]$, set membership becomes a soft proximity kernel, and the two covering conditions become the penalties below; opacity binarization (Sec. 3.4) finally returns the relaxed selection to a discrete one. Unlike classical set cover, the sets are themselves learnable, since stroke geometry is optimized jointly: strokes can move to cover more structure with fewer strokes.

Computing the Contribution. We sample N_q detail guidance points $\{\mathbf{q}_j\}_{j=1}^{N_q}$ from M_d, with higher-response regions sampled more frequently. For a detail stroke $S_{d,i}$ and a point $\mathbf{q}_j$, we define

their spatial proximity as

$$\mathcal{K}(S_{d,i}, \mathbf{q}_j) = \exp\left(-\frac{d^2(\mathbf{q}_j, S_{d,i})}{2\sigma_{\text{cov}}^2}\right), \tag{4}$$

where $d(\mathbf{q}_j, S_{d,i})$ denotes the minimum distance from $\mathbf{q}_j$ to the sampled points on stroke $S_{d,i}$, and σ_{cov} controls the spatial tolerance. The visible contribution of stroke i to $\mathbf{q}_j$ is $v_{j,i} = \alpha_i \mathcal{K}(S_{d,i}, \mathbf{q}_j)$, the relaxed counterpart of "stroke i is selected and covers $\mathbf{q}_j$"; a stroke contributes strongly only when it is both close and visible.

Constraint on Faithfulness. Every guidance point must be covered at least once. We penalize points whose total visible contribution falls below 1:

$$\mathcal{L}_{\text{cov}}^{d} = \frac{1}{N_q}\sum_{j=1}^{N_q}\left[\max\left(0, 1 - \sum_{i=1}^{N_d} v_{j,i}\right)\right]^2. \tag{5}$$

For binary visibilities and memberships, this term vanishes exactly when every point is covered, so salient regions of I cannot be lost as detail strokes are suppressed.

Constraint on Conciseness. Every guidance point should also be covered at most once. We penalize pairwise overlap between the contributions of different strokes:

$$\mathcal{L}_{\text{u}} = \frac{1}{2N_q}\sum_{j=1}^{N_q}\left[\left(\sum_{i=1}^{N_d} v_{j,i}\right)^2 - \sum_{i=1}^{N_d} v_{j,i}^2\right]. \tag{6}$$

For binary values, the bracket equals $k(k-1)$ when k strokes cover $\mathbf{q}_j$ and vanishes exactly when $k \leq 1$. Minimizing $\mathcal{L}_{\text{u}}$ therefore concentrates the coverage of each point on a single stroke, and strokes that only duplicate others lose opacity and fade out.

Adaptive to User-specified Abstraction Level. Several covers can satisfy both conditions, and neither condition says which image-specific details matter most. We therefore add a relevance prior. For stroke $S_{d,i}$, we define

$$\mathcal{A}_d(S_{d,i}, M_d) = \frac{1}{2K}\sum_{k=1}^{K} M_d(\mathbf{y}_{i,k}) + \frac{1}{2}\max_k M_d(\mathbf{y}_{i,k}), \tag{7}$$

where K denotes the number of sampled points on the Bézier curve. The first term measures how consistently the stroke overlaps with salient regions indicated by M_d, while the second preserves the response when only part of the stroke passes through such a region. We then encourage relevant strokes to remain visible through

$$\mathcal{L}_{\text{keep}} = \frac{1}{N_d}\sum_{i=1}^{N_d}(1-\alpha_i)\,\text{sg}\left[\mathcal{A}_d(S_{d,i}, M_d)\right], \tag{8}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator. Hiding a stroke thus incurs a penalty proportional to its relevance, weighted by the **user-specified** abstraction parameter λ_{keep} and traded against the overlap penalty of $\mathcal{L}_{\text{u}}$. A larger λ_{keep} keeps more image-relevant detail, even at the cost of some overlap, while a smaller value yields a sparser cover and a more abstract sketch. The set of visible detail strokes is therefore determined jointly by the image content and the chosen abstraction level[1].

3.4 Progressive Contour-to-Detail Optimization

Making strong visibility decisions before stroke geometry has sufficiently adapted may prematurely suppress useful detail strokes. We therefore adopt a progressive contour-to-detail optimization strategy. The contour and detail stroke groups are optimized jointly, while the early optimization places greater emphasis on stabilizing the global contour. Once the contour converges, its best state is restored and frozen, and the optimization emphasis is gradually shifted toward detail refinement.

After this transition, opacity binarization is activated to encourage clear stroke visibility decisions. This allows detail strokes to first adapt their geometry before being strongly encouraged to remain fully visible or become nearly suppressed. In this way, AdaSketch prioritizes *where to draw* before determining *which strokes to retain*.

[1]The full objective of AdaSketch is given in Appendix B.

Table 1: Quantitative comparison with existing photo-to-vector sketch generation methods. Metrics are computed between the rasterized vector sketches and XDoG edge references extracted from the masked input images. Best results are highlighted in bold. Segment budget $N = 62$ for all methods.

Methods	CLIP ViT/16 ↑	CLIP ViT/32 ↑	MS-SSIM ↑	DreamSim ↓	LPIPS ↓
CLIPasso	0.8645	0.8815	0.4992	0.3577	0.2299
CLIPascene	0.8928	0.9116	0.4263	0.2627	0.2740
ControlSketch	0.8742	0.8884	0.5344	0.2938	0.3612
SwiftSketch	0.8283	0.8512	0.5382	0.3621	0.2577
SLDgen	0.8339	0.8578	0.4113	0.3826	0.3792
Ours	**0.9083**	**0.9130**	**0.6674**	**0.2370**	**0.1987**



Figure 3: Qualitative comparison with existing photo-to-vector sketch generation methods under the same segment budget $N = 62$. AdaSketch preserves recognizable global structures while selectively retaining salient internal details and suppressing redundant strokes, producing concise yet faithful vector sketches across diverse input contents.

4 Experiments

4.1 Evaluation and Implementation

Evaluation. For a fair comparison, we evaluate all methods on the same set of 16,990 images spanning 85 object categories from the unseen-category split of the ControlSketch dataset intro-

duced by SwiftSketch (Arar et al., 2025). We compare AdaSketch with five vector sketch generation methods: CLIPasso (Vinker et al., 2022), CLIPascene (Vinker et al., 2023), ControlSketch and SwiftSketch (Arar et al., 2025), and SLDgen (Magne et al., 2026). Following the edge-based evaluation approach of CLIPascene (Vinker et al., 2023), we extract an XDoG (Winnemöller, 2011) edge map from each target image and use it as the reference for evaluating the rasterized vector sketches. We report CLIP image–image similarity using ViT/16 and ViT/32 (Radford et al., 2021), MS-SSIM (Wang et al., 2003), DreamSim (Fu et al., 2023), and LPIPS (Zhang et al., 2018) to assess semantic, structural, and perceptual similarity to these references.

Implementation. By default, we initialize each sketch with a maximum budget of $N = 62$ Bézier segments, comprising $N_c = 31$ connected segments that jointly form a single closed contour and $N_d = 31$ independent detail strokes, each represented by one Bézier segment. Optimization is performed for up to 3,000 iterations on a single NVIDIA A800 GPU.

4.2 Quantitative Results

Table 1 reports the quantitative comparison with existing methods. AdaSketch achieves the best performance across all five metrics, demonstrating consistently high similarity to the visual reference at semantic, structural, and perceptual levels. In particular, AdaSketch obtains CLIP similarities of 0.9083 and 0.9130 with ViT-B/16 and ViT-B/32, respectively. The improvement is more pronounced in structural and perceptual metrics: compared with the strongest baseline for each metric, AdaSketch improves MS-SSIM by 0.1292 and reduces DreamSim and LPIPS by 0.0257 and 0.0312, respectively. These results indicate that AdaSketch better preserves the characteristic structures of the input while maintaining strong semantic and perceptual consistency.

4.3 Qualitative Results

Figure 3 compares AdaSketch with existing photo-to-vector sketch generation methods under the same segment budget. Across diverse inputs, AdaSketch preserves the global shape and characteristic structure of the depicted subject while maintaining a compact sketch representation. In particular, our method selectively retains salient internal structures, such as the cone pattern of the ice cream and the facial and clothing structures of the human subject, while avoiding excessive or repetitive strokes. In contrast, some baselines either omit characteristic structures under stronger abstraction or introduce dense and redundant local lines. These results illustrate the benefit of decoupling the available stroke capacity from the rendered sketch complexity: AdaSketch can preserve content-dependent details without requiring all initialized strokes to remain visible, leading to a favorable balance between faithfulness and conciseness.

4.4 More Results

We provide more results to further examine the behavior of AdaSketch under different abstraction levels and segment budgets, as well as its applicability to more complex multi-object images.

Controllable Sketch Abstraction. We first investigate how the user-specified parameter λ_{keep} controls the abstraction level of the generated sketch. As shown in Figure 4, we keep the maximum segment budget fixed and vary only λ_{keep}. At a small λ_{keep}, the preservation constraint on detail strokes is weak, allowing redundant or less relevant strokes to be suppressed. The resulting sketches are therefore highly abstract, relying primarily on the global contour and a small number of characteristic details to convey object identity. As λ_{keep} increases, stronger preservation *encourages more image-relevant detail strokes to remain visible*. Importantly, the additional details are content-dependent rather than uniformly added across different inputs. For example, increasing λ_{keep} progressively introduces body and hump structures for the camel, facial and ear structures for the elephant, wing and body details for the duck, and page structures for the book. These results show that λ_{keep} provides a direct and intuitive control over the faithfulness–conciseness trade-off.

Robustness to segment budget. We next investigate the effect of the maximum segment budget N while keeping λ_{keep} fixed. As shown in Figure 5, we vary the number of initialized Bézier segments from $N = 46$ to 94. Across all budgets, AdaSketch consistently preserves the recognizable global shape of the target objects, while the amount of internal detail gradually adapts to the available representational capacity. With a small budget ($N = 46$), the limited capacity is primarily allocated to



Figure 4: Controlling sketch abstraction with user-specified abstraction level λ_{keep}. Rows correspond to $\lambda_{\text{keep}} = 0.01$, 0.1 and 1.0. Under the same segment budget $N = 62$, increasing λ_{keep} encourages more image-relevant detail strokes to remain visible, enabling continuous control from concise abstraction to richer detail.



Figure 5: Effect of different maximum segment budgets under a fixed $\lambda_{\text{keep}} = 0.3$. Rows correspond to $N = 46$, 62, 78 and 94. Larger budgets provide additional capacity for representing finer details, while AdaSketch adaptively suppresses unnecessary strokes to maintain compact sketches.

the global contour and a small number of characteristic structures, yielding concise but recognizable sketches. As the budget increases, additional capacity allows finer image-relevant structures to be represented. This behavior is particularly visible for the flower, hat, backpack, and broccoli, where progressively richer internal structures emerge at larger budgets. In contrast, objects that can already be represented compactly, such as the boat and tomato, exhibit relatively smaller changes as N increases. These results demonstrate that AdaSketch remains stable across a broad range of initial segment budgets and that N primarily determines the available representational capacity rather than prescribing the final rendered complexity.

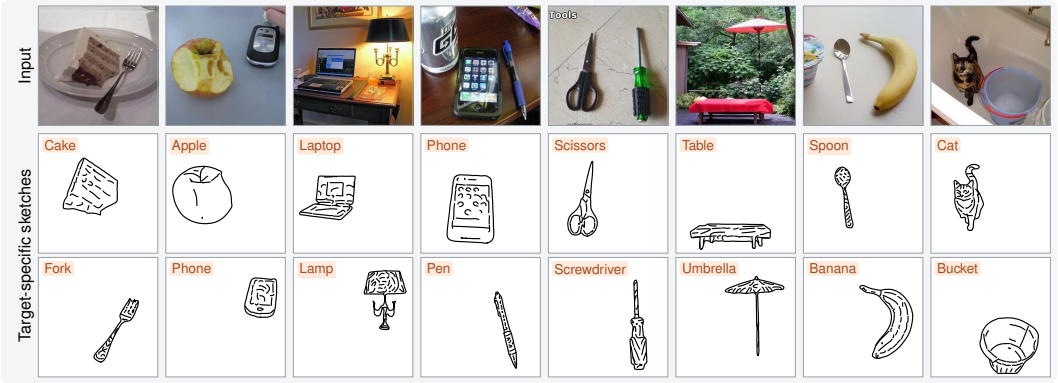


Figure 6: Target-specific vector sketch generation in multi-object images. Given the same input image, users can specify different target objects for sketch generation. AdaSketch generates recognizable vector sketches of the selected objects while excluding unrelated scene content.

Target-Specific Sketch Generation in Multi-Object Images. Finally, we evaluate whether AdaSketch can extend beyond isolated single-object inputs to more complex images containing multiple objects. The multi-object images used in this evaluation are selected from the COCO 2017 dataset (Lin et al., 2014). Given a user-specified target object, AdaSketch uses the corresponding object name as the target phrase to obtain an object-specific foreground mask, and subsequently generates a vector sketch for the selected object rather than the entire image. As shown in Figure 6, AdaSketch can generate different object-level sketches from the same multi-object image according to the specified target. For example, from the same input photo, the method can separately sketch the cake or fork, the apple or phone, and the laptop or lamp. Despite the presence of other objects and background content, the generated sketches preserve the recognizable shape and characteristic details of the selected target while excluding unrelated scene content. These results demonstrate that AdaSketch can support user-directed, target-specific vector sketch generation in multi-object images, extending its applicability beyond isolated object inputs.

5 Limitations

There remain several aspects that could be further improved. AdaSketch relies on foreground segmentation to initialize the contour and provide spatial guidance for detail strokes, making the results somewhat sensitive to inaccurate masks, especially when object regions are severely occluded or missing. In addition, although adaptive stroke selection makes the rendered complexity less dependent on a prescribed stroke count, the maximum segment budget still bounds the available representational capacity, and an overly small budget may limit fine structures for complex objects. Finally, AdaSketch adopts per-image optimization to adapt stroke geometry and visibility to individual inputs, which introduces additional computational cost compared with feed-forward approaches. Future work could explore more robust foreground guidance, content-adaptive stroke capacity, and more efficient optimization to further improve the robustness and efficiency of the framework.

6 Conclusion

We presented AdaSketch, which treats the stroke budget in photo-to-vector sketching as capacity rather than as the complexity of the rendered sketch. A closed contour secures the silhouette, and detail strokes earn their visibility through a coverage-based objective: salient structure must be covered, by as few strokes as possible, with a single parameter λ_{keep} setting how much image-relevant detail survives. A progressive contour-to-detail schedule decides where to draw before deciding which strokes to keep. Under the same budget, AdaSketch achieves the best semantic, structural and perceptual fidelity among the compared methods, offers continuous control over abstraction, remains robust across budgets, and extends to target-specific sketching in multi-object scenes. More broadly, our results suggest that a stroke budget should bound what a sketch can say, not dictate what it must say.

AI USE STATEMENT

The authors used AI-assisted tools for language polishing, grammar refinement, and improving the clarity of manuscript writing. All research ideas, method design, experiments, analyses, and final manuscript content were reviewed, verified, and approved by the authors.

ETHICS STATEMENT

This work studies photo-to-vector sketch generation and does not involve human subjects, private user data, or newly collected sensitive datasets. The method uses pretrained vision-language, segmentation, and diffusion models, and may therefore inherit biases or failure modes from these models. We use the evaluated datasets only for research purposes and follow their intended usage.

REPRODUCIBILITY STATEMENT

We provide the full optimization objective and auxiliary regularizers in Appendix B, and report implementation details, hyperparameters, prompt construction, optimization schedule, and stopping criteria in Appendix D. The experimental setup, datasets, baselines, and evaluation metrics are described in Section 4. To support reproducibility, we include the key optimization code and the contour-detail objective terms in the supplementary material.

REFERENCES

Ellie Arar, Yarden Frenkel, Daniel Cohen-Or, Ariel Shamir, and Yael Vinker. SwiftSketch: A diffusion model for image-to-vector sketch generation. In *SIGGRAPH*, 2025.

Caroline Chan, Frédo Durand, and Phillip Isola. Learning to generate line drawings that convey geometry and semantics. In *CVPR*, pp. 7905–7915, 2022.

Kevin Frans, Lisa Soros, and Olaf Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. In *NeurIPS*, volume 35, pp. 5207–5218, 2022.

Stephanie Fu, Netanel Y. Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. DreamSim: learning new dimensions of human visual similarity using synthetic data. In *NeurIPS*, volume 36, 2023.

Naoto Inoue, Daichi Ito, Ning Xu, Jimei Yang, Brian Price, and Toshihiko Yamasaki. Learning to trace: Expressive line drawing generation from photographs. *Comput. Graph. Forum*, 38(7): 69–80, 2019.

Ajay Jain, Amber Xie, and Pieter Abbeel. VectorFusion: Text-to-SVG by abstracting pixel-based diffusion models. In *CVPR*, pp. 1911–1920, 2023.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *ICCV*, pp. 3992–4003, 2023.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pp. 19730–19742, 2023.

Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-Sketching: Inferring contour drawings from images. In *WACV*, pp. 1403–1412, 2019a.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph.*, 39(6), 2020.

Yijun Li, Chen Fang, Aaron Hertzmann, Eli Shechtman, and Ming-Hsuan Yang. Im2Pencil: Controllable pencil illustration from photographs. In *CVPR*, pp. 1525–1534, 2019b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollar, and Larry Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pp. 740–755, 2014.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection. In *ECCV*, pp. 38–55, 2024.

Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. Closure-aware sketch simplification. *ACM Trans. Graph.*, 34(6), 2015.

Tanguy Magne, Alexandre Binninger, Ruben Wiersma, and Olga Sorkine-Hornung. Single line drawing generation via semantics-driven optimization. *Comput. Graph. Forum*, pp. e70502, 2026.

Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning deep sketch abstraction. In *CVPR*, pp. 8014–8023, 2018.

Toru Ogawa, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa. Sketch simplification by classifying strokes. In *ICPR*, pp. 1065–1070, 2016.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.

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. In *ICML*, pp. 8748–8763, 2021.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. arXiv preprint arXiv:2401.14159, 2024.

Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning to sketch with shortcut cycle consistency. In *CVPR*, pp. 801–810, 2018.

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), 2022.

Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. CLIPascene: Scene sketching with different types and levels of abstraction. In *ICCV*, pp. 4123–4133, 2023.

Zhou Wang, Eero P. Simoncelli, and Alan C. Bovik. Multiscale structural similarity for image quality assessment. In *ACSSC*, volume 2, pp. 1398–1402, 2003.

Holger Winnemöller. XDoG: advanced image stylization with extended difference-of-gaussians. In *NPAR*, pp. 147–156, 2011.

Ximing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. DiffSketcher: text guided vector sketch synthesis through latent diffusion models. In *NeurIPS*, volume 36, 2023.

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. SVGDreamer: Text guided SVG generation with diffusion model. In *CVPR*, pp. 4546–4555, 2024.

Ran Yi, Yong-Jin Liu, Yu-Kun Lai, and Paul L. Rosin. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *CVPR*, pp. 10735–10744, 2019.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, 2018.

Zhanjie Zhang, Quanwei Zhang, Junsheng Luan, Mengyuan Yang, Yun Wang, and Lei Zhao. VectorSketcher: Learning to create a vector-based free-hand sketch. *Eng. Appl. Artif. Intell.*, 156: 111005, 2025.

A Additional Experiment Results

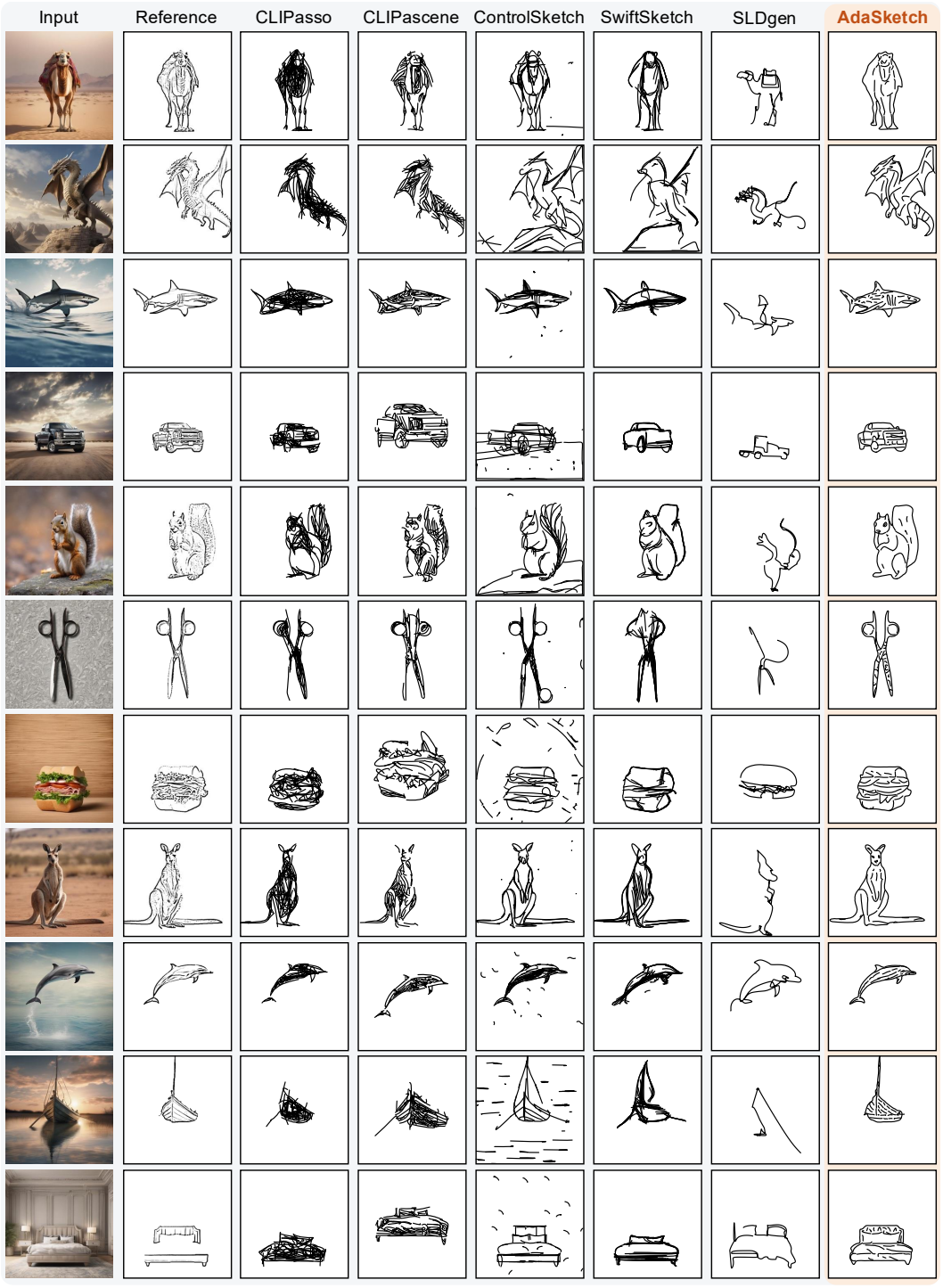


Figure 7: Additional qualitative results on photo-to-vector sketch generation under the same segment budget $N = 62$. Across diverse object categories, AdaSketch preserves recognizable silhouettes and salient internal structures while avoiding excessive hatching, clutter, or fragmented strokes. Compared with existing methods, our results provide a cleaner balance between structural faithfulness and compact vector representation.

B TRAINING OBJECTIVES

This section provides the complete training objective used in AdaSketch. The main paper introduces the contour-detail representation, the spatial guidance maps, and the core detail selection terms. Here, we describe how these components are combined during optimization and define the auxiliary terms that are omitted from the main paper for compactness.

B.1 OVERALL OBJECTIVE AND SDS GUIDANCE

At optimization step e, AdaSketch optimizes

$$\mathcal{L}^{(e)} = \lambda_{\text{sds}}^{(e)} \mathcal{L}_{\text{sds}} + \mathcal{L}_c^{(e)} + \mathcal{L}_d^{(e)}. \tag{9}$$

The contour objective $\mathcal{L}_c^{(e)}$ stabilizes the global silhouette, and the detail objective $\mathcal{L}_d^{(e)}$ optimizes internal structure, stroke selection, and detail geometry. The SDS term provides high-level generative guidance from a frozen diffusion model.

Let θ denote the optimizable sketch parameters, including Bézier control points and detail-stroke opacity logits. A differentiable rasterizer maps the current vector sketch to a raster image I_θ, which is encoded by the VAE of the frozen diffusion model into a latent representation z_θ. At each optimization step, we sample a diffusion timestep t and Gaussian noise η, and obtain the noisy latent $z_{\theta,t}$ using the forward diffusion process.

Conditioned on the text prompt p and the ControlNet condition image I_{cond}, the frozen diffusion model predicts the noise $\eta_\phi(z_{\theta,t}, t, (p, I_{\text{cond}}))$, where ϕ denotes the fixed model parameters. The SDS gradient is

$$\nabla_\theta \mathcal{L}_{\text{sds}} = \mathbb{E}_{t,\eta}\left[w(t) \left(\eta_\phi(z_{\theta,t}, t, (p, I_{\text{cond}})) - \eta \right) \frac{\partial z_\theta}{\partial \theta} \right], \tag{10}$$

where $w(t) = \sqrt{\bar{\alpha}_t}(1 - \bar{\alpha}_t)$ follows the diffusion schedule. For single-object inputs, the text prompt p is the BLIP-2 caption. For multi-object inputs, p is the user-specified target object name.

B.2 STROKE SAMPLING AND GUIDANCE EVALUATION

Both contour and detail strokes are represented by cubic Bézier curves. For a segment S_i with control points $\mathbf{P}_i = [\mathbf{p}_{i,0}, \mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3}]^\top \in \mathbb{R}^{4\times 2}$, we use

$$S_i(\gamma) = (1-\gamma)^3 \mathbf{p}_{i,0} + 3(1-\gamma)^2 \gamma \mathbf{p}_{i,1} + 3(1-\gamma)\gamma^2 \mathbf{p}_{i,2} + \gamma^3 \mathbf{p}_{i,3}, \quad \gamma \in [0,1]. \tag{11}$$

The parameter γ specifies a position along the curve. We evaluate each curve at K uniformly spaced parameter values,

$$\gamma_k = \frac{k-1}{K-1}, \qquad k = 1, \ldots, K. \tag{12}$$

For contour segments, the sampled stroke points are denoted by $\mathbf{x}_{i,k} = S_{c,i}(\gamma_k)$. For detail strokes, they are denoted by $\mathbf{y}_{i,k} = S_{d,i}(\gamma_k)$. These points lie on the vector strokes and are used to query guidance maps, compute distances to sampled guidance locations, and estimate local geometry.

The contour map C and the detail map M_d are dense guidance maps, while the sketch consists of sparse vector curves. Direct pixel-wise comparison is sensitive to small shifts between a stroke and the map response. We therefore sample guidance points from each map and evaluate whether these points are covered by nearby strokes. For a guidance map E, where $E = C$ for contour guidance and $E = M_d$ for detail guidance, a candidate image location $\mathbf{q}$ is sampled with probability

$$P(\mathbf{q}) = \frac{E(\mathbf{q})}{\sum_{\mathbf{u}} E(\mathbf{u})}. \tag{13}$$

The denominator sums over all image locations $\mathbf{u}$. This sampling places more guidance points in high-response regions while preserving the relative importance encoded by the map.

For a curve S_i, the coverage of a sampled guidance point $\mathbf{q}_j$ is

$$\mathcal{K}(S_i, \mathbf{q}_j) = \max_k \exp\left(-\frac{\|\mathbf{q}_j - \mathbf{z}_{i,k}\|_2^2}{2\sigma_{\text{cov}}^2} \right), \tag{14}$$

where $\mathbf{z}_{i,k}$ is the k-th sampled point on the curve. For contour segments, $\mathbf{z}_{i,k} = \mathbf{x}_{i,k}$; for detail strokes, $\mathbf{z}_{i,k} = \mathbf{y}_{i,k}$. Thus, a guidance point is considered well covered if it is close to at least one sampled point on the curve.

B.3 Contour Objective

The contour branch is optimized to follow the target silhouette while remaining a well-formed closed curve. For contour segment $S_{c,i}$, we first measure its agreement with the contour map C:

$$\mathcal{A}_c(S_{c,i}, C) = \frac{1}{2K} \sum_{k=1}^{K} C(\mathbf{x}_{i,k}) + \frac{1}{2} \max_k C(\mathbf{x}_{i,k}). \tag{15}$$

The average term measures how well the full segment follows the boundary, while the maximum term preserves a useful signal when only part of the segment touches the contour map. To encourage complete boundary coverage, we also use contour guidance points $\{\mathbf{q}_j^c\}_{j=1}^{N_q}$ sampled from C. The contour coverage term is

$$\begin{aligned} \mathcal{L}_{\text{cov}}^c = \lambda_{\mathcal{A}_c} \frac{1}{N_c} \sum_{i=1}^{N_c} [1 - \mathcal{A}_c(S_{c,i}, C)] \\ + \lambda_{\mathcal{K}_c} \frac{1}{N_q} \sum_{j=1}^{N_q} \left[\max \left(0, 1 - \sum_{i=1}^{N_c} \mathcal{K}(S_{c,i}, \mathbf{q}_j^c) \right) \right]^2 . \end{aligned} \tag{16}$$

Boundary guidance alone does not fully determine a stable parameterization of the closed contour chain. We therefore add a contour geometry regularizer

$$\mathcal{R}_c = \lambda_{\text{s}} \mathcal{R}_{\text{s}} + \lambda_{\text{len}} \mathcal{R}_{\text{len}} + \lambda_{\text{o}} \mathcal{R}_{\text{o}}. \tag{17}$$

To define $\mathcal{R}_{\text{s}}$, let $\mathbf{u}_i$ and $\mathbf{v}_{i+1}$ denote the outgoing tangent of segment i and the incoming tangent of segment $i+1$, respectively. The tangent regularizer is

$$\mathcal{R}_{\text{s}} = \frac{1}{N_c} \sum_{i=1}^{N_c} \left(1 - \mathbf{u}_i^\top \mathbf{v}_{i+1} \right)^2 . \tag{18}$$

Since adjacent contour segments share endpoints, the contour is already C^0-continuous. This term encourages C^1-like continuity by reducing sharp direction changes at segment junctions.

Let ℓ_i be the sampled arc length of segment i, and let $\bar{\ell}$ be the mean segment length. The length regularizer is

$$\mathcal{R}_{\text{len}} = \frac{1}{N_c} \sum_{i=1}^{N_c} \left(\frac{\ell_i - \bar{\ell}}{\text{sg}(\bar{\ell}) + \epsilon} \right)^2 . \tag{19}$$

It prevents the closed contour from being represented by a few overly long segments and many nearly collapsed ones.

For non-adjacent contour segments i and j, let R_{ij} measure their sampled spatial and tangent similarity. The overlap regularizer is

$$\mathcal{R}_{\text{o}} = \frac{1}{|\mathcal{N}_c|} \sum_{(i,j) \in \mathcal{N}_c} R_{ij}, \tag{20}$$

where $\mathcal{N}_c$ is the set of non-adjacent contour segment pairs. This term discourages distant parts of the contour chain from tracing the same local boundary region.

The contour objective at step e is

$$\mathcal{L}_c^{(e)} = \lambda_c^{(e)} \left(\mathcal{L}_{\text{cov}}^c + \mathcal{R}_c \right). \tag{21}$$

B.4 Detail Objective

The detail branch is optimized to cover salient internal guidance points, suppress redundant strokes, and preserve geometry that remains consistent with the foreground object. The main paper defines the detail coverage loss $\mathcal{L}_{\mathrm{cov}}^{d}$, the uniqueness loss $\mathcal{L}_{\mathrm{u}}$, and the keep loss $\mathcal{L}_{\mathrm{keep}}$. These terms form the core stroke-selection objective. Here, we add auxiliary terms for opacity and geometry stability.

Semi-transparent detail strokes can make the sketch visually ambiguous. We therefore include an opacity binarization term

$$\mathcal{L}_{\alpha} = \frac{1}{N_d} \sum_{i=1}^{N_d} \alpha_i (1 - \alpha_i). \tag{22}$$

This term is minimized when a detail stroke is either nearly invisible or nearly fully visible.

Visible detail strokes should also remain inside the valid object interior. We measure the average interior response of detail stroke $S_{d,i}$ by

$$m_i = \frac{1}{K} \sum_{k=1}^{K} M_I(\mathbf{y}_{i,k}). \tag{23}$$

The inside loss is

$$\mathcal{L}_{\mathrm{in}} = \frac{\sum_{i=1}^{N_d} \mathrm{sg}(\alpha_i)(1 - m_i)^2}{\sum_{i=1}^{N_d} \mathrm{sg}(\alpha_i) + \epsilon}. \tag{24}$$

The detached opacity gives larger weight to visible strokes while preventing a stroke from avoiding the penalty by reducing its opacity.

The attention-based initialization provides useful geometric structure for detail strokes. We preserve this structure with a soft anchor term. Let d_i^{a} be the Huber distance between the current and initialized control points of detail stroke $S_{d,i}$. The anchor loss is

$$\mathcal{L}_{\mathrm{a}}^{d} = \frac{\sum_{i=1}^{N_d} \mathrm{sg}(\alpha_i) d_i^{\mathrm{a}}}{\sum_{i=1}^{N_d} \mathrm{sg}(\alpha_i) + \epsilon}. \tag{25}$$

This term discourages unnecessary drift from useful initialized structures while still allowing local refinement.

We group the detail geometry terms as

$$\mathcal{R}_{d}^{(e)} = \lambda_{\mathrm{in}}^{(e)} \mathcal{L}_{\mathrm{in}} + \lambda_{\mathrm{a}}^{(e)} \mathcal{L}_{\mathrm{a}}^{d}. \tag{26}$$

This is the detail geometry regularizer removed in the ablation "w/o $\mathcal{R}_d$".

Combining guidance coverage, stroke selection, opacity binarization, and detail geometry regularization gives

$$\mathcal{L}_{d}^{(e)} = \lambda_{\mathrm{cov}}^{d,(e)} \mathcal{L}_{\mathrm{cov}}^{d} + \lambda_{\mathrm{u}}^{(e)} \mathcal{L}_{\mathrm{u}} + w_{\mathrm{keep}}^{(e)} \lambda_{\mathrm{keep}} \mathcal{L}_{\mathrm{keep}} + \lambda_{\alpha}^{(e)} \mathcal{L}_{\alpha} + \mathcal{R}_{d}^{(e)}. \tag{27}$$

The parameter λ_{keep} is kept explicit because it controls the user-specific abstraction level. The remaining coefficients are scheduled or fixed optimization weights.

B.5 Progressive Optimization and Convergence

Both sketch groups are optimized from the beginning, but their relative emphasis changes over time. We first emphasize the contour objective to stabilize the global silhouette. Once the contour-specific loss stops improving, the best contour state is restored and the contour branch is frozen.

Let e_f be the freezing step. After freezing, the transfer progress is

$$\rho_e = \mathrm{clip}\left((e - e_f) r_{\mathrm{tr}}, 0, 1\right). \tag{28}$$

For a scheduled coefficient with initial value λ^0 and final value λ^1, we use

$$\lambda^{(e)} = (1 - \rho_e)\lambda^0 + \rho_e \lambda^1. \tag{29}$$

Table 2: Quantitative comparison of the impact of geometric regularizers in both contour and detail optimization. All scores are computed between the rasterized sketch and the XDoG edge reference.

Methods	CLIP ViT/16↑	CLIP ViT/32↑	MS-SSIM ↑	DreamSim ↓	LPIPS ↓
W/o $\mathcal{R}_c$	0.9055	0.9118	0.6178	0.2673	0.2419
W/o $\mathcal{R}_d$	0.9035	0.9088	0.5185	0.2782	0.2916
Full model	**0.9083**	**0.9130**	**0.6674**	**0.2370**	**0.1987**

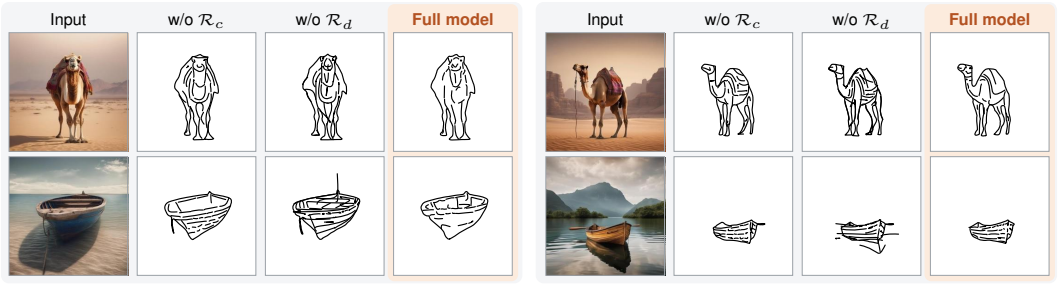


Figure 8: Qualitative ablation of geometric regularizers in contour and detail optimization. Without $\mathcal{R}_c$, the contour becomes less stable. Without $\mathcal{R}_d$, detail strokes become noisier and less spatially coherent. The full model better preserves the global silhouette and internal structure.

This interpolation is applied to the contour coefficient, the scheduled coefficients in $\mathcal{L}_d^{(e)}$, and the SDS guidance weight. It provides a smooth transition from contour-dominant optimization to detail refinement.

The opacity binarization term is delayed by setting $\lambda_\alpha^{(e)} = 0$ before the contour-to-detail transfer is complete. Afterward, $\lambda_\alpha^{(e)}$ is gradually increased. This allows detail strokes to first adapt their geometry before their opacities are strongly pushed toward near-binary values. In this way, AdaSketch prioritizes *where to draw* before making final decision about *which strokes to retain*.

After contour freezing, weight transfer, and opacity warmup are complete, we monitor the late-stage refinement score

$$J_e = \lambda_{\text{sds}}^{(e)} \mathcal{L}_{\text{sds}} + \mathcal{L}_d^{(e)}. \tag{30}$$

The frozen contour objective is excluded because it no longer reflects progress in the trainable detail parameters. We smooth this score using an exponential moving average,

$$\overline{J}_e = \beta_{\text{ema}} \overline{J}_{e-1} + (1 - \beta_{\text{ema}}) J_e. \tag{31}$$

Optimization stops when the smoothed score does not improve for a fixed number of evaluations. The best stored contour and detail states are restored before exporting the final vector sketch.

C Additional Ablation Studies

C.1 Ablation on geometric regularizers in contour and detail optimization

We study the effect of the geometric regularizers by removing the contour regularizer $\mathcal{R}_c$ and the detail regularizer $\mathcal{R}_d$ separately. As shown in Table 2, the full model achieves the best performance on all metrics. Removing $\mathcal{R}_c$ slightly reduces the CLIP scores and causes a clear drop in MS-SSIM, DreamSim, and LPIPS. This suggests that the contour regularizer does not merely improve local smoothness, but also helps maintain a stable and well-parameterized silhouette. Removing $\mathcal{R}_d$ leads to a larger degradation, especially on MS-SSIM and LPIPS. This indicates that detail regularization is important for preserving image-aligned internal structures and avoiding unstable stroke placement. The qualitative results in Figure 8 show a consistent trend. Without $\mathcal{R}_c$, the object silhouette remains recognizable, but the contour can become less balanced and locally less coherent. Without $\mathcal{R}_d$, the detail strokes are more likely to drift, overdraw, or accumulate in incorrect

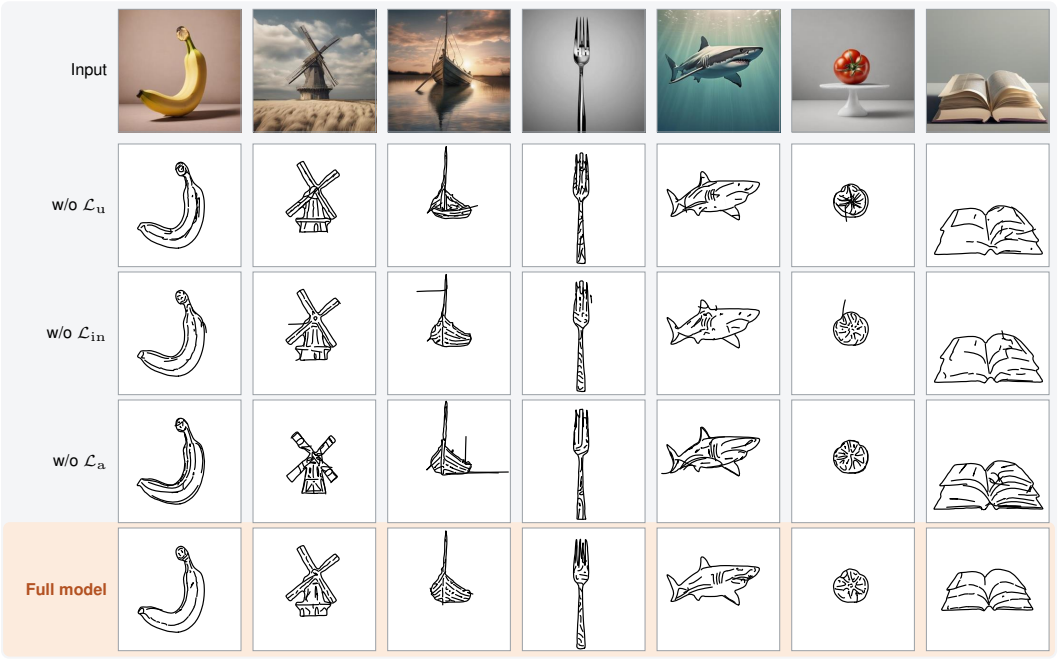


Figure 9: Ablation on detail regularization terms. We compare the full model with variants that remove the uniqueness term $\mathcal{L}_{\mathrm{u}}$, the inside constraint $\mathcal{L}_{\mathrm{in}}$, or the geometry anchor $\mathcal{L}_{\mathrm{a}}$. Each term addresses a different failure mode in detail optimization, including redundant stroke assignment, weak foreground consistency, and unstable geometry refinement.

regions. This is particularly visible in the boat examples, where internal strokes become noisy and less faithful to the object structure. In contrast, our full model produces cleaner silhouettes and more compact internal details. These results show that the two regularizers are complementary. $\mathcal{R}_c$ stabilizes the global contour, while $\mathcal{R}_d$ improves the reliability of detail stroke refinement. We provide an additional ablation on the individual factors inside $\mathcal{R}_d$ in the appendix.

C.2 Ablation on Detail Regularization Terms

We further analyze the contribution of the regularization terms used in detail optimization. Figure 9 compares the full model with variants that remove the uniqueness term $\mathcal{L}_{\mathrm{u}}$, the inside constraint $\mathcal{L}_{\mathrm{in}}$, or the geometry anchor $\mathcal{L}_{\mathrm{a}}$. These terms affect different failure modes. The uniqueness term discourages multiple detail strokes from explaining the same local guidance. The inside constraint restricts visible detail strokes to the valid foreground interior. The geometry anchor preserves the useful structure provided by the attention-based initialization. The visual comparison shows that removing these terms degrades detail organization in different ways. Without $\mathcal{L}_{\mathrm{u}}$, the results often contain repeated or crowded local strokes, as seen in the banana, fork, and tomato examples. Without $\mathcal{L}_{\mathrm{in}}$, some internal strokes become less consistent with the foreground region and the sketch can lose a clear separation between object shape and internal details. Without $\mathcal{L}_{\mathrm{a}}$, detail strokes are less stable and may drift from their initialized semantic locations, which weakens fine structures such as the windmill blades, shark body marks, and book pages. The full model gives more stable detail placement across categories while preserving compact and readable vector sketches.

D Implementation Details

Basic Setup. All sketches are optimized at 512×512 resolution and rendered as black cubic Bézier curves with stroke width 2.5. Unless otherwise stated, we use $N_c = 31$ connected Bézier segments for the contour branch and $N_d = 31$ independent Bézier strokes for the detail branch. The contour branch is rendered as one closed stroke, while detail strokes are open curves with learnable opacity.

Detail opacities are initialized to 0.85. Each Bézier segment is evaluated with $K = 24$ sampled points. For each guidance map, we sample $N_q = 512$ guidance points and use $\sigma_{\mathrm{cov}} = 4.5$ for the coverage kernel. We optimize each sketch for at most 3000 iterations.

Text Prompts and Target Extraction. For single-object inputs, we generate a caption using BLIP-2 and extract the foreground object phrase for Grounded-SAM using a lightweight noun-phrase parser based on spaCy, with a rule-based fallback when the parser is unavailable. For multi-object inputs, the target object name is specified by the user and used directly. The SDS prompt is the BLIP-2 caption for single-object inputs and the user-specified target name for multi-object inputs.

Progressive Optimization Schedule. The contour-dominant stage stops when the contour objective does not improve for 250 iterations. The best contour state is then restored and frozen. The contour-to-detail transfer rate is $r_{\mathrm{tr}} = 0.005$. After transfer, the SDS guidance weight reaches $\lambda_{\mathrm{sds}}^{(e)} = 4.0$, and the contour coefficient reaches $\lambda_{c}^{(e)} = 0.1$. The contour alignment and coverage weights are $\lambda_{\mathcal{A}_c} = \lambda_{\mathcal{K}_c} = 1.0$, and the contour regularizer weights are $\lambda_{\mathrm{s}} = 0.2$, $\lambda_{\mathrm{len}} = 0.05$, and $\lambda_{\mathrm{o}} = 0.8$.

Detail Weights and Stopping. For the detail objective, we report the effective coefficients used in Eq. 27. The effective coverage weight is annealed from $\lambda_{\mathrm{cov}}^{d,(e)} = 1.25$ to $\lambda_{\mathrm{cov}}^{d,(e)} = 0.75$. After transfer, we use $\lambda_{\mathrm{u}}^{(e)} = 0.75$, $w_{\mathrm{keep}}^{(e)} = 2.5$, $\lambda_{\mathrm{in}}^{(e)} = 1.25$, and $\lambda_{\mathrm{a}}^{(e)} = 0.0625$. Unless otherwise specified, the user-specific abstraction parameter is set to $\lambda_{\mathrm{keep}} = 0.3$. The opacity-binarization weight is linearly warmed up to $\lambda_{\alpha}^{(e)} = 0.2$ after the contour-to-detail transfer is complete. We use EMA-based early stopping with $\beta_{\mathrm{ema}} = 0.9$.