

Neural Vectorization and Stroke Dynamics: The 2025 Landscape of Image-to-Plotter Workflows

The Topological Crisis in Digital Fabrication

The intersection of computer vision and digital fabrication has historically been fraught with a fundamental translation error: the dissonance between the pixel and the path. For the better part of three decades, the conversion of raster imagery into vector graphics was dominated by heuristic edge-detection algorithms—most notably Potrace—which interpret visual information strictly as boundaries of contrast. While effective for screen-based graphic design or print media where filled shapes are rendered instantaneously, these methods are fundamentally ill-suited for pen plotting, Computer Numerical Control (CNC) machining, and robotic drawing.

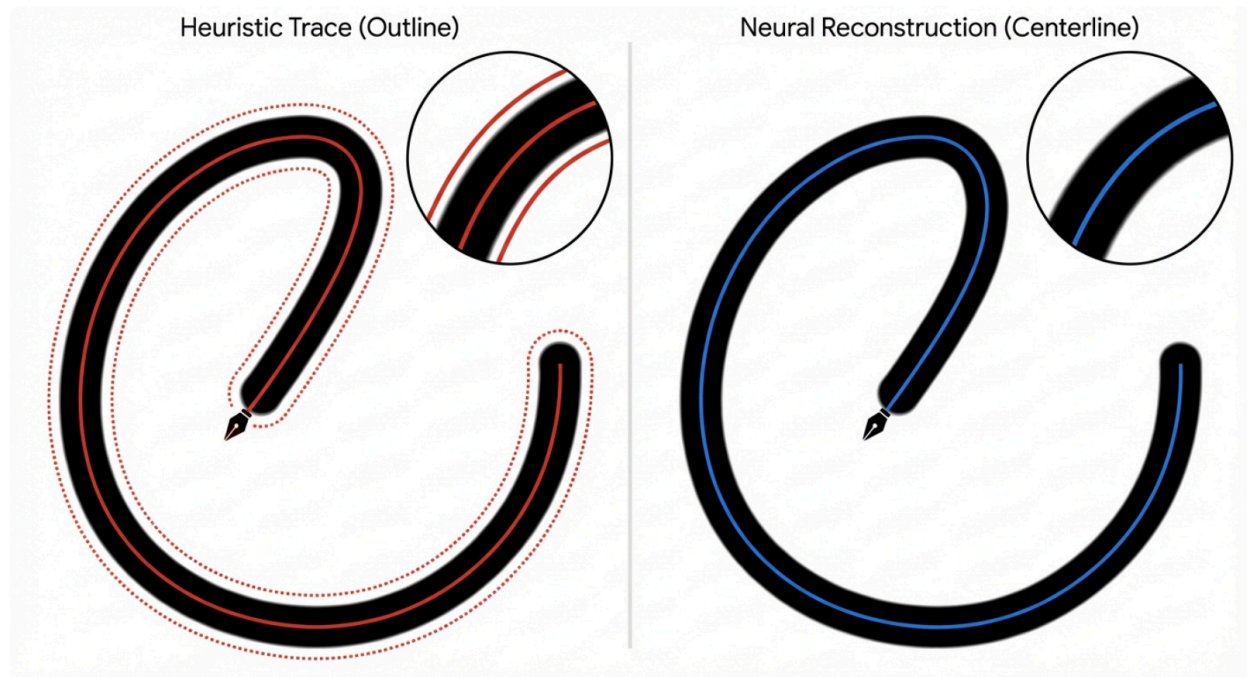
The core of this crisis lies in the distinction between the "outline" and the "centerline." A raster line, no matter how thin, possesses a width in pixel space. Traditional vectorization algorithms, operating on the principle of segmentation, trace the perimeter of this width. When such a path is sent to a pen plotter—a machine that operates with a fixed-width stylus—the result is a "double stroke." The machine traces the left side of the line, turns around at the terminus, and traces the right side. This topological redundancy is not merely an aesthetic failure; it represents a mechanical inefficiency that doubles the travel distance (L), thereby doubling execution time and frequently degrading the physical substrate through excessive ink saturation or material abrasion.¹

By 2023, the limitations of these heuristic approaches had become the primary bottleneck in high-fidelity robotic art. The "uncanny valley" of plotter art was defined by this hollowness—lines that lacked the gestural weight and skeletal integrity of a human stroke. However, the period between 2023 and 2025 has witnessed a paradigm shift. The field has moved from geometric heuristics to **cognitive stroke reconstruction**. Recent advancements presented at premier venues such as NeurIPS 2025, CVPR 2025, and SIGGRAPH have introduced models that do not merely trace contrast but infer the *process* of drawing. These systems, leveraging Vision-Language Models (VLMs), Differentiable Rasterization, and specialized Deep Learning architectures, demonstrate a capacity to "understand" the semantic structure of an image, allowing for the generation of clean, topology-aware vector paths that mimic human motor control.

This report provides an exhaustive analysis of these developments, specifically evaluating their utility for pen plotting. It examines the tension between **visual fidelity** (pixel-perfect reconstruction) and **mechanical fidelity** (efficient, logical stroke order), categorizing new

methodologies into "Generative" (VLM-based) and "Reconstructive" (Stroke-level) approaches. It further explores the "hybrid" workflows that have emerged as the pragmatic standard for artists and engineers in 2025, combining the generative power of diffusion models with the topological rigor of morphological skeletonization.

The Topology Gap: Heuristic Tracing vs. Neural Reconstruction



Comparison of vectorization topologies. (Left) Traditional heuristic tracing (e.g., Potrace) interprets a raster line as a filled shape, resulting in a 'double-stroke' perimeter path when plotted. (Right) Neural stroke reconstruction (e.g., LineDrawer) infers the medial axis, generating a single-stroke path that aligns with the intended drawing gesture, optimizing mechanical efficiency.

The Physics of the Plotter: Why Topology Matters

To fully appreciate the impact of recent neural networks, one must understand the physical constraints of the hardware they are feeding. A pen plotter, such as the ubiquitous AxiDraw or the newer NextDraw, is a Cartesian robot that operates in a continuous vector space. Unlike a raster printer, which deposits ink in a linear sweep, a plotter is a *traveling* machine. Its efficiency is governed by the Traveling Salesperson Problem (TSP)—the optimization of the path to visit a set of points (or lines) with minimal travel cost.³

The "Outline Problem" creates a specific type of inefficiency. A human drawing a line from point A to point B executes a single vector translation. A heuristic tracer converting that line

creates a loop: $A \rightarrow B \rightarrow A$. This not only doubles the draw time but also introduces artifacts. If the pen is a marker, the overlap at the edges creates a darker "railroad track" effect, leaving the center of the line lighter. If the pen is a fine-liner, the result is two distinct lines where one was intended.

Furthermore, heuristic tracers are blind to "occlusion." In a drawing where a foreground object crosses a background object, a pixel-based tracer sees only the visible segments. It breaks the background line into two disjointed shapes. A plotter rendering this will draw the first segment, lift the pen, travel to another part of the page, and eventually return to draw the second segment. This fragmentation increases "pen-up" operations, which introduces servo wear and settling time errors. The holy grail of plotter-oriented vectorization is **Stroke Continuity**—the ability to infer that the two background segments are part of one continuous line that happens to be occluded, and to draw them (or plan them) as a coherent unit.⁵

This is where the Deep Learning revolution of 2024-2025 becomes critical. By moving from "seeing pixels" to "understanding structure," models like **LineDrawer** and **OmniSVG** promise to align the digital file with the physical reality of the drawing process.

The Rise of Generative Vision-Language Models (VLMs)

The most prominent development in the 2025 landscape is the adaptation of Large Language Models (LLMs) and Vision-Language Models (VLMs) to generate SVG code directly. These models treat SVG paths not as visual entities to be rendered, but as a formal language (XML) that can be predicted token-by-token. This approach fundamentally alters the mechanism of vector creation, moving it from image processing to linguistic translation.

OmniSVG: Unified Multimodal Generation

OmniSVG, presented at NeurIPS 2025, represents the current state-of-the-art in VLM-based vector generation. Unlike previous models that relied on intermediate representations or rigid templates, OmniSVG leverages the **Qwen2.5-VL** architecture to tokenize SVG commands and coordinates directly.⁷

Architecture and Tokenization Strategies

The core innovation of OmniSVG lies in its **SVG Tokenizer**. Standard language models struggle with continuous numerical data (like coordinates in a vector space) because they tokenize text into sub-words. OmniSVG addresses this by compressing vector commands into discrete tokens, effectively creating a specialized vocabulary for geometry.

- **Decoupled Logic and Geometry:** The model parameterizes commands (such as MoveTo, LineTo, CubicBezier) separately from their coordinate arguments. This decoupling is crucial. It allows the model to learn the *syntax* of SVG (the structural logic

of opening tags, closing paths, and defining attributes) independently from the *geometry* (the precise spatial location of points). This separation enables efficient training and helps the model maintain the expressiveness required to handle complex structures like intricate anime characters or detailed diagrams.⁷

- **Autoregressive Generation:** OmniSVG generates SVGs autoregressively, meaning it builds the image sequentially from start to finish. In theory, this mimics the drawing process. However, the "order" of generation is determined by the training data's XML structure rather than a semantic drawing stroke order. The model predicts the next token based on the previous context, allowing it to complete partial SVGs or generate new ones from textual descriptions.⁹

Capabilities and Limitations for Plotting

The capabilities of OmniSVG are vast. It can perform Text-to-SVG, Image-to-SVG, and even Character-Reference generation, producing resolution-independent, editable assets.⁷ The introduction of the **MMSVG-2M** dataset—a massive corpus of two million richly annotated SVG assets—ensures that the model has seen a diverse range of vector styles, from simple icons to complex illustrations.⁷

However, for the specific application of pen plotting, OmniSVG presents a mixed bag of utility:

- **Semantic Coherence (Pro):** Unlike pixel tracers, OmniSVG understands that a "cat" is composed of specific shapes (ears, whiskers, tail). It generates these as coherent semantic units. It avoids the "noise" of pixel-tracers, where texture or paper grain might be misinterpreted as geometry.
- **Topological Efficiency (Con):** The generated SVGs are often optimized for visual rendering (browser display) rather than mechanical plotting. The path topology may contain overlapping shapes (e.g., a pupil drawn on top of an eye, rather than a hole cut out of it) or inefficient travel routes typical of standard SVG/XML hierarchies. The autoregressive order is based on file structure, which does not necessarily correlate with an optimal physical drawing path. A plotter might draw the left ear, then the tail, then the right ear, resulting in excessive air-travel.⁷

StarVector: Code Generation vs. Artistic Reconstruction

Parallel to the VLM approach is **StarVector** (2025), which frames vectorization as a pure code-generation task. Utilizing a backbone based on **StarCoder** integrated with a Vision Transformer (ViT), StarVector leverages the massive code-pretraining of LLMs to "write" SVG files.¹²

- **Operational Domain:** StarVector is explicitly optimized for structured graphics: **icons, logotypes, technical diagrams, graphs, and charts**. In these domains, the geometric primitives (perfect circles, aligned rectangles, straight lines) are paramount. The model achieves high "DinoScores" on benchmarks like SVG-Icons and SVG-Fonts, indicating a high degree of visual fidelity to the input.¹²

- **The "Natural Image" Limitation:** Crucially, the model documentation and independent reviews explicitly state that StarVector is **not suitable for natural images or illustrations**.¹² For a pen plotter artist seeking to convert a photograph, a portrait, or a sketch into a drawing, StarVector is likely insufficient. It tends to produce geometric primitives rather than the organic, flowing Bézier curves needed for artistic line art. The model treats the image as a blueprint to be reconstructed with code, rather than a canvas to be drawn with a stroke.

The distinction between OmniSVG and StarVector highlights a bifurcation in the field: **OmniSVG** aims for generalized, multimodal creativity suitable for illustration, while **StarVector** targets precise, structural design. For the pen plotter artist, OmniSVG offers more promise, but both suffer from the "code-first" bias that prioritizes XML validity over stroke connectivity.

Model Capability Matrix: 2025 State-of-the-Art

● Strongest Fit ● Moderate/Fair ● Limitation

MODEL NAME	ARCHITECTURE	TARGET DOMAIN	STROKE CONTINUITY	TOPOLOGY FOR PLOTTING
OmniSVG <small>NeurIPS 2025</small>	VLM (Qwen2.5-VL)	General, Anime Characters, Icons	Medium	Fair
StarVector <small>HuggingFace / StarCoder</small>	VLM (StarCoder + ViT)	Icons, Charts, Technical Diagrams	Low	Fair
LineDrawer <small>Computers & Graphics 2025</small>	Hierarchical (Diffusion + Perception)	Complex Line Art, Plotter Extraction	High	Excellent

Capability comparison of leading 2025 image-to-vector models. 'Stroke Continuity' and 'Topology Quality' are the critical metrics for pen plotting suitability. LineDrawer excels in mechanical fidelity, while OmniSVG dominates in semantic generation.

Data sources: [OmniSVG \(NeurIPS\)](#), [LineDrawer \(Elsevier\)](#), [StarVector \(HuggingFace\)](#), [OmniSVG Paper](#)

Stroke-Level Reconstruction: The "Plotter-Ready" Breakthroughs

While VLMs generate code, a separate lineage of research has focused on **Stroke-Level Process Reconstruction**. These methods are arguably the most significant development for pen plotting in 2025 because they attempt to reverse-engineer the physical drawing process itself. They do not see the image as a grid of pixels or a tree of XML tags, but as a temporal sequence of pen movements.

LineDrawer: Hierarchical Reconstruction of Human Mechanics

LineDrawer, published in *Computers & Graphics* (August 2025), directly addresses the "complex line art" vectorization problem by simulating human perception and motor control. The authors recognize that human artists do not draw outlines; they draw strokes that represent the medial axis of a form. To replicate this, LineDrawer employs a three-level hierarchical framework.⁵

Stage 1: High-Level Semantic Ordering (The "Mind")

The first stage addresses the global planning problem. It utilizes **PaintsUndo**, a pre-trained diffusion model designed to "undo" a painting back to its initial sketch. LineDrawer uses this to generate a sequence of "semantic keyframes." This provides a global roadmap of the drawing process, mimicking how an artist plans a composition: laying down structural lines (the "gesture") before refining local details.⁵

- *Hallucination as a Feature*: By using a diffusion model to hallucinate the *process* (the temporal dimension) from a static image, LineDrawer solves the "occlusion problem." The diffusion model can infer that a line continues behind an object because it "knows" the semantic structure of the object. This allows the system to generate strokes that are topologically complete, even if they are partially hidden in the raster image.⁵

Stage 2: Mechanical Optimization (The "Hand")

Once the global order is established, the second stage models the mechanics of the pen. It employs a **habitual function** that captures human drawing dynamics. This stage merges local sub-strokes into longer, coherent paths. For a plotter, this is critical. A naive vectorizer might break a long curve into ten segments, causing the plotter to lift and settle ten times. LineDrawer's mechanical optimization creates a single, fluid Bézier curve, reducing the number of "pen-up/pen-down" operations—the primary bottleneck in plotting speed.⁵

Stage 3: Perceptual Stroke Rendering (The "Eye")

Finally, a "Cue-Aware" neural renderer extracts variable-width strokes at the pixel level. Unlike centerline tracers that assume a fixed width (skeletonization), LineDrawer preserves the expressive pressure sensitivity of the original art. It identifies "cue points" along the stroke

and infers the local width and taper.⁵ This allows for the generation of "variable-width centerlines," which can be physically rendered using high-quality fountain pens with flexible nibs or by using "hatching" algorithms (like those in vtype or AxiDraw software) to simulate width through multiple passes.

Single-Line Drawing Vectorization: The "Continuous Path"

Research by Magne et al., presented at **Pacific Graphics 2025**, introduces a method specifically for **Single-Line Drawings (SLD)**. This style of art—exemplified by Picasso's continuous line animals—is the theoretical ideal for pen plotting, as it requires zero pen lifts.

The method improves upon traditional skeletonization by refining the "medial axis" (the mathematical skeleton) of the image using a graph-based approach.

- **Neural Intersection Classification:** The core innovation is a specialized Convolutional Neural Network (CNN) trained to classify intersections. In a traditional skeleton, a crossing lines often results in a "junction" node where the path forks. The neural network analyzes the local geometry to determine the correct "flow" of the line, deciding which path is the continuation of the current stroke. This prevents the "junction confusion" common in traditional algorithms where lines inexplicably take 90-degree turns.⁶
- **Application:** This method enables the conversion of clean raster sketches into single, unbroken paths. For robotic drawing, this maximizes efficiency and minimizes the vibration associated with servo acceleration/deceleration during lifts.⁶

Neural Image Abstraction with B-Splines

In a similar vein, Berio et al. (2025) propose a method using **Long Smoothing B-Splines** in a differentiable rendering pipeline.

- **Differentiable Geometric Primitives:** Unlike methods that optimize pixel grids, this approach optimizes the control points of B-splines directly. The B-spline is a curve defined by a set of control points, offering inherent smoothness and continuity.
- **Curvature Minimization:** The loss function used to train this model includes terms for "jerk" and "snap" (the second and third derivatives of position). By minimizing these values, the model effectively enforces smooth, organic curves.
- **Plotting Relevance:** This results in paths that are naturally "fair"—smooth, sweeping curves that minimize machine vibration. In servo-based systems, rapid changes in curvature (high jerk) cause the motors to stutter or overshoot. By mathematically enforcing smoothness, this neural abstraction method produces paths that are physically quieter and cleaner when plotted.¹⁶

Differentiable Rasterization: The Engine of Optimization

The engine driving many of these advancements, particularly in refining the geometry of the strokes, is **Differentiable Rasterization**. This technology bridges the gap between the vector world (parametric curves) and the raster world (pixels), allowing the "raster error" (the difference between the vector output and the target image) to be back-propagated to the vector parameters (control points, stroke width, opacity).

From DiffVG to Bézier Splatting

Since its introduction in 2020, **DiffVG** (Differentiable Vector Graphics) has been the standard differentiable rasterizer. It enabled the first wave of "Vectorization via Optimization" approaches. However, DiffVG suffers from computational bottlenecks, particularly in calculating gradients at shape boundaries, making it slow to converge for complex images.¹³

Bézier Splatting, presented at NeurIPS 2025, introduces a massive leap in efficiency. It adapts the concept of **Gaussian Splatting**—a technique that revolutionized 3D radiance fields—to 2D vector graphics.

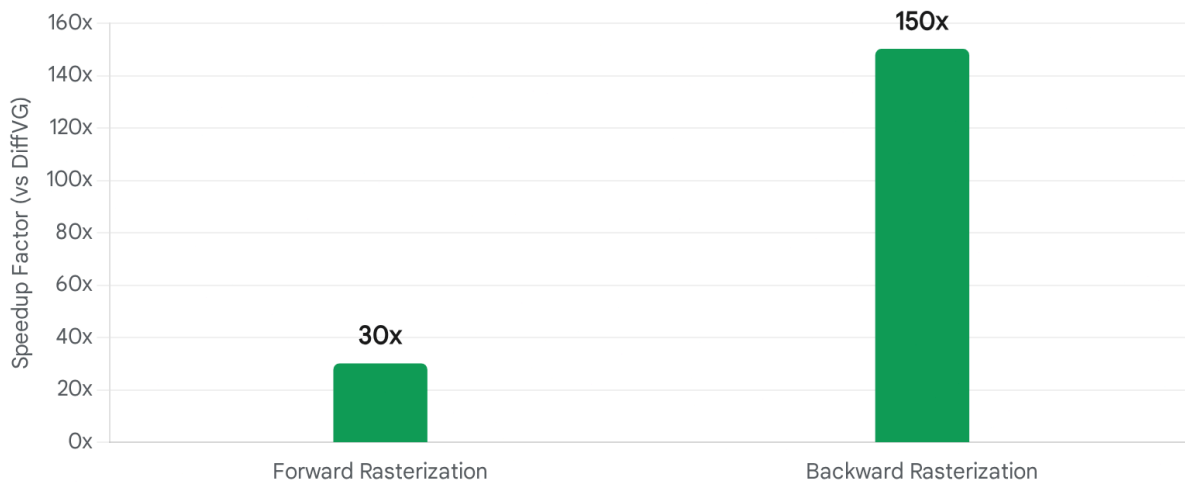
- **Gaussian Splatting Integration:** Instead of performing complex boundary integrals to calculate pixel coverage, Bézier Splatting samples 2D Gaussians along the Bézier curves. The "splatting" process (projecting these Gaussians onto the pixel grid) is computationally trivial for modern GPUs and is fully differentiable.¹⁷
- **Performance:** The result is an order-of-magnitude speedup. Experiments show it achieves up to **150x faster** backward computation (gradient calculation) for open curves compared to DiffVG.¹⁸
- **SVG Output:** Crucially, unlike general Gaussian Splatting which typically outputs a "cloud" of points that is useless for a plotter, Bézier Splatting constrains the Gaussians to the mathematical definition of the curve. This ensures that after optimization, the result can be exported as a standard XML-based SVG path, ready for plotting.¹⁸

This speedup is not just a quality-of-life improvement; it enables "real-time" vectorization applications where a user might adjust a prompt and see the vector path evolve instantly, or where a robot could potentially "see" and vectorise a scene on the fly.

Optimization Speedup: Bézier Splatting vs. DiffVG

Relative Speed Comparison

● Bézier Splatting Speedup Factor



Relative speedup of Bézier Splatting compared to DiffVG. Bézier Splatting achieves up to 150x faster backward computation for open curves, enabling real-time optimization of complex vector graphics.

Data sources: [arXiv \(Bézier Splatting\)](#), [NeurIPS](#)

The Pragmatic "Hybrid" Pipeline: The 2025 Standard

While end-to-end models like LineDrawer and OmniSVG represent the cutting edge of research, they are often difficult to deploy due to closed-source weights or complex dependencies. Consequently, a "hybrid" workflow combining Generative AI with classical morphological algorithms has emerged as the robust "state-of-the-practice" for high-quality plotting in 2025. Discussions within the plotter community (e.g., *PlotterArt*, *Hacker News*) reveal that this pipeline offers the best balance of control and quality.¹⁴

Step 1: Flux + Prompt Engineering for Linearity

The workflow begins with high-fidelity diffusion models, with **Flux.1-dev** and **SDXL** being the preferred engines in 2025. Unlike older models, these can be prompted to generate specific "Technical Drawing" or "Line Art" styles with high geometric coherence.

- **Prompt Strategy:** Users employ prompts that enforce sparsity and high contrast (e.g., "minimalist line art," "technical schematic," "vector illustration"). The goal is to generate a raster image that is visually essentially a vector—clean lines, white background, no shading.¹⁴

Step 2: Morphological Skeletonization (Lee's Method)

Instead of using Canny edge detection (which creates the dreaded double-stroke outline), practitioners employ **Lee's Method** for skeletonization. Available in Python libraries like scikit-image, this algorithm is a morphological erosion process. It iteratively removes pixels from the boundary of binary shapes until only a single-pixel-wide "skeleton" remains.

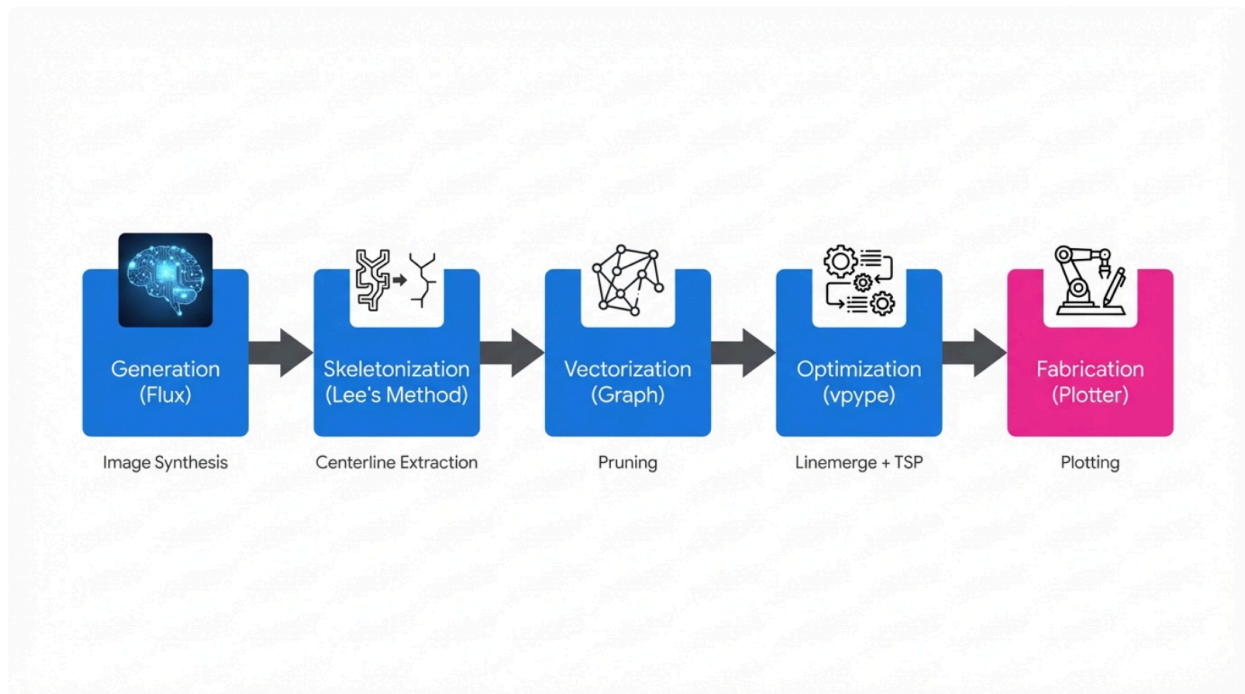
- **Topological Advantage:** This guarantees a centerline representation. The line width of the raster image is effectively discarded, leaving only the connectivity. This is the crucial step that transforms a "shape" into a "path".¹⁴

Step 3: Graph Pruning and Conversion

The raw skeleton produced by Lee's method is often noisy, containing small "spurs" or isolated pixels. The hybrid pipeline converts this pixel skeleton into a **Graph Data Structure** (Nodes and Edges).

- **Pruning:** "Hanging" edges (short branches with no connection at one end) are identified and pruned if they are below a certain length threshold. This cleans up the "fuzz" often seen in skeletonized rasters.
- **Simplification:** Chains of nodes with a degree of 2 (i.e., nodes that just connect two other nodes in a line) are merged into single vector segments. This significantly reduces the data size and complexity before optimization.¹⁴

The Hybrid Plotting Pipeline (2025)



The standard high-fidelity workflow for AI-driven pen plotting. (1) Image Synthesis via Flux/SDXL. (2) Centerline extraction via Morphological Skeletonization (Lee's Method). (3) Vectorization & Graph Pruning. (4) Mechanical Optimization via vpype (TSP sorting, merging).

Step 4: Mechanical Optimization with vpype

Regardless of whether the vector is generated by OmniSVG or a Skeletonization pipeline, the raw output is rarely optimal for a physical machine. 2025 has seen the solidification of **vpype** (Vector Pipeline) as the industry-standard tool for this "mechanical optimization" phase. vpype is a CLI tool that acts as a post-processor, bridging the gap between "Software Vector" (visual) and "Hardware Vector" (motion).³

Key optimizations required for neural outputs include:

- **linemerge:** Neural models and skeletonization algorithms often output fragmented paths (e.g., a continuous line represented as hundreds of tiny, separate segments). linemerge topologically reconnects these segments if their endpoints are within a specified tolerance. This drastically reduces the plotter's "pen-down" frequency.
- **linesort:** This command solves the **Traveling Salesperson Problem (TSP)** for the plotter. It reorders the drawing sequence of the paths to minimize non-drawing travel time (air-time). For complex neural-generated textures, applying linesort can often reduce the total plot time by 40-60%, significantly extending the lifespan of the

machine's motors.³

- **reloop:** This utility randomizes the start/stop point of closed loops. In generative art, if every circle starts at the exact same angle (e.g., 0 degrees), the slight ink bleed at the pen-down point creates a visible "seam" across the artwork. reloop distributes these artifacts, making them invisible to the eye.³

Future Directions: Theoretical Implications and New Benchmarks

Benchmarking and Evaluation: MMSVGBench

A critical maturation in the field is the establishment of standardized benchmarks. The introduction of **MMSVGBench** (part of the OmniSVG project) provides a rigorous framework for evaluating vector generation models.

- **Metrics:** It evaluates models not just on visual similarity, but on **Generation Fidelity** (DinoScore), **Diversity**, and **Editability**.
- **Significance:** The inclusion of "editability" is a proxy for the cleanliness of the vector topology. A highly editable SVG usually has fewer, more logical control points and simpler path structures. For the plotting community, this metric correlates strongly with mechanical viability. A model that scores high on editability is likely to produce paths that are easier for a plotter to execute.¹¹

Path Optimization Algorithms: Beyond Heuristics

While vtype relies on standard greedy or 2-opt algorithms for path sorting, recent academic work has begun to apply **Deep Reinforcement Learning (DRL)** to the path ordering problem.

- **RL for TSP:** New approaches model the plotter's path as a sequential decision-making process. By training an agent on millions of plotting scenarios, these DRL models can find traversal orders that outperform standard heuristics, particularly in multi-color plotting scenarios where pen-swapping introduces a high cost penalty.
- **Graph Neural Networks (GNNs):** GNNs are being applied to encode the vector graph and predict the optimal traversal order directly. These networks learn the latent structure of the drawing, potentially identifying "clusters" of strokes that should be drawn together to maintain ink consistency.²⁴

Conclusions

The landscape of image-to-vector modeling in 2025 has evolved from a reliance on simple geometric tracing to sophisticated, semantically aware reconstruction. The "topological crisis" of the double-stroke outline is being solved by a new generation of models that understand the *process* of drawing.

Semantic Reconstruction is Key: The most promising results for plotting come from models like **LineDrawer** and **Single-Line Drawing Vectorization** (Magne et al.), which reconstruct the stroke rather than just the pixel boundary. These models effectively "hallucinate" the human motor control required to create the image.

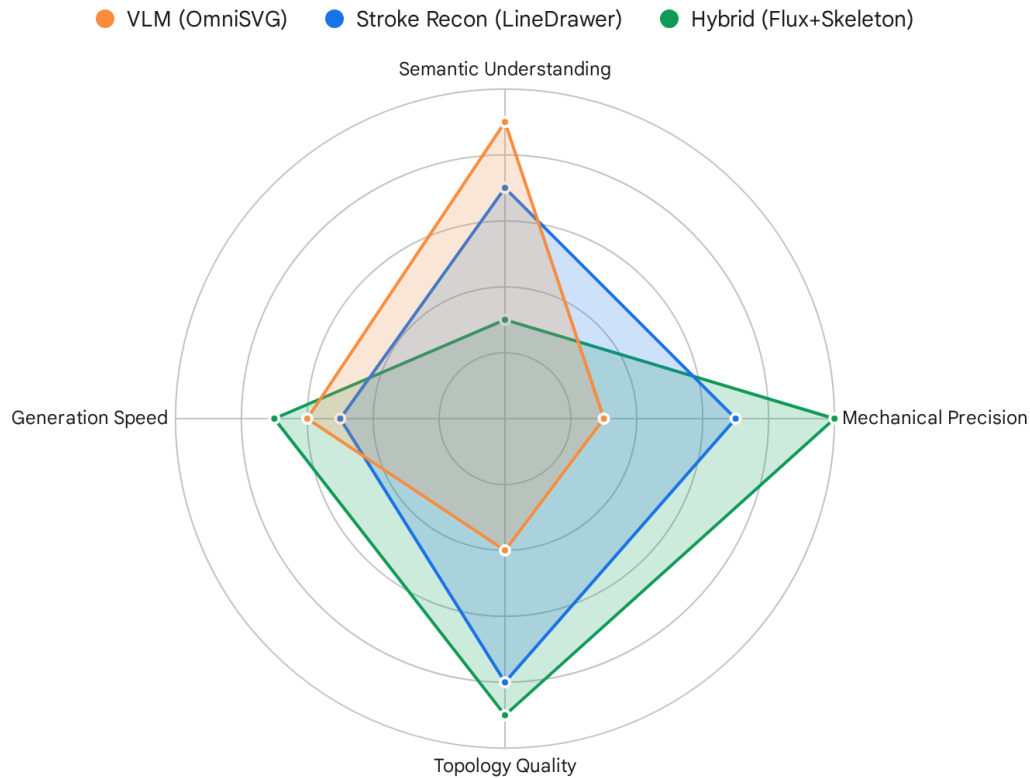
The "Centerline" Gap is Closing: While no single "perfect" end-to-end model exists for all art styles, the combination of **LineDrawer** (for reconstruction) and **Bézier Splatting** (for optimization) offers a robust toolkit for creating plotter-ready art from raster inputs.

Hybrid Workflows Dominate: For immediate, practical application, the **Flux** →

Skeletonization → **vptype** pipeline remains the gold standard. It allows artists to leverage the massive creative range of standard diffusion models while enforcing the rigorous topological constraints required for physical fabrication.

For researchers and practitioners in 2025, the focus has shifted from "how do I trace this image?" to "how do I reconstruct the intent of this image?"—a shift that promises to finally align the digital potential of AI with the physical reality of the pen plotter.

Strategic Selection Framework: Choosing the Right Pipeline



Trade-off analysis of vectorization approaches. VLMs (OmniSVG) lead in creativity but lag in topology. Hybrid workflows (Skeletonization) excel in mechanical precision but lack semantic understanding. Stroke Reconstruction (LineDrawer) offers a balanced middle ground.

Data sources: [OmniSVG \(NeurIPS\)](#), [LineDrawer \(Computers & Graphics\)](#), [HN Discussion](#), [Bézier Splatting \(arXiv\)](#)

Works cited

1. started to use AI to go from photo to line art : r/PlotterArt - Reddit, accessed February 7, 2026, https://www.reddit.com/r/PlotterArt/comments/1l6gnhs/started_to_use_ai_to_go_from_photo_to_line_art/
2. [Trace] Center line option - LightBurn, accessed February 7, 2026, <https://lightburn.fider.io/posts/371/trace-center-line-option>
3. Which version of Processing should I learn if I want to plot my images with an Axidraw plotter? : r/PlotterArt - Reddit, accessed February 7, 2026, https://www.reddit.com/r/PlotterArt/comments/13qh4h5/which_version_of_processing_should_i_learn_if_i/

4. Geometrical Optimal Navigation and Path Planning—Bridging Theory, Algorithms, and Applications - PMC, accessed February 7, 2026, <https://pmc.ncbi.nlm.nih.gov/articles/PMC12656126/>
5. LineDrawer.pdf - Creative Intelligence and Synergy Lab, accessed February 7, 2026, <https://cislabs.hkust-gz.edu.cn/media/documents/LineDrawer.pdf>
6. Single-Line Drawing Vectorization, accessed February 7, 2026, <https://igl.ethz.ch/projects/sld-vectorization/single-line-drawing-vectorization-pacific-graphics-2025-magne-et-al.pdf>
7. NeurIPS Poster OmniSVG: A Unified Scalable Vector Graphics Generation Model, accessed February 7, 2026, <https://neurips.cc/virtual/2025/poster/115696>
8. OmniSVG: A Unified Scalable Vector Graphics Generation Model - arXiv, accessed February 7, 2026, <https://arxiv.org/html/2504.06263v2>
9. OmniSVG: A Unified Scalable Vector Graphics Generation Model - OpenReview, accessed February 7, 2026, <https://openreview.net/pdf/67dc0bc8d01a1a124aeac60a7bb456507f3dff10.pdf>
10. OmniSVG: A Unified Scalable Vector Graphics Generation Model - arXiv, accessed February 7, 2026, <https://arxiv.org/html/2504.06263v1>
11. OmniSVG: A Unified Scalable Vector Graphics Generation Model - arXiv, accessed February 7, 2026, <https://arxiv.org/html/2504.06263v3>
12. starvector/starvector-1b-im2svg · Hugging Face, accessed February 7, 2026, <https://huggingface.co/starvector/starvector-1b-im2svg>
13. RoboSVG: A Unified Framework for Interactive SVG Generation with Multi-modal Guidance, accessed February 7, 2026, <https://arxiv.org/html/2510.22684v1>
14. Show HN: Txt2plotter – True centerline vectors from Flux.2 for pen ..., accessed February 7, 2026, <https://news.ycombinator.com/item?id=46685064>
15. (PDF) Single-Line Drawing Vectorization - ResearchGate, accessed February 7, 2026, https://www.researchgate.net/publication/396238058_Single-Line_Drawing_Vectorization
16. Neural Image Abstraction Using Long ... - Idiap Publications, accessed February 7, 2026, https://publications.idiap.ch/attachments/papers/2025/Berio_TOG_2025.pdf
17. Bézier Splatting for Fast and Differentiable Vector Graphics Rendering - arXiv, accessed February 7, 2026, <https://arxiv.org/html/2503.16424>
18. NeurIPS Poster Bézier Splatting for Fast and Differentiable Vector Graphics Rendering, accessed February 7, 2026, <https://neurips.cc/virtual/2025/poster/117178>
19. Bézier Splatting for Fast and Differentiable Vector Graphics Rendering - arXiv, accessed February 7, 2026, <https://arxiv.org/html/2503.16424v4>
20. I built a Text-to-SVG pipeline that finally gives me clean single-stroke paths (Flux.2 + Skeletonization) : r/PlotterArt - Reddit, accessed February 7, 2026, https://www.reddit.com/r/PlotterArt/comments/1qhhvn4/i_built_a_texttosvg_pipeline_that_finally_gives/
21. Bibliography - Maks Surguy's blog on Technology Innovation, IoT, Design and Code, accessed February 7, 2026, <https://maxoffsky.com/feed/>
22. abey79/vptype: The Swiss-Army-knife command-line tool for plotter vector

- graphics. - GitHub, accessed February 7, 2026, <https://github.com/abey79/vpype>
23. [NeurIPS 2025] OmniSVG is the first family of end-to-end multimodal SVG generators that leverage pre-trained Vision-Language Models (VLMs), capable of generating complex and detailed SVGs, from simple icons to intricate anime characters. - GitHub, accessed February 7, 2026, <https://github.com/OmniSVG/OmniSVG>
24. Neural Combinatorial Optimization for Time-Dependent Traveling Salesman Problem, accessed February 7, 2026, [https://openreview.net/forum?id=UXTR6ZYV1x&referrer=%5Bthe%20profile%20of%20Chuchu%20Fan%5D\(%2Fprofile%3Fid%3D~Chuchu_Fan2\)](https://openreview.net/forum?id=UXTR6ZYV1x&referrer=%5Bthe%20profile%20of%20Chuchu%20Fan%5D(%2Fprofile%3Fid%3D~Chuchu_Fan2))
25. Tackling the Traveling Salesman Problem with Graph Neural Networks - Medium, accessed February 7, 2026, <https://medium.com/stanford-cs224w/tackling-the-traveling-salesman-problem-with-graph-neural-networks-b86ef4300c6e>