Processing Freehand Vector Sketches

by

Chenxi Liu

B.Eng., Beihang University, 2013

M.Sc., Carnegie Mellon University, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

(Computer Science)

The University of British Columbia

(Vancouver)

June 2023

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the thesis entitled:

Processing Freehand Vector Sketches

submitted by **Chenxi Liu** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy** in **Computer Science**.

Examining Committee:

Alla Sheffer, Professor, Computer Science, UBC
Supervisor

Michiel van de Panne, Professor, Computer Science, UBC
Supervisory Committee Member

Robert Rohling, Professor, Electrical and Computer Engineering & Mechanical Engineering, UBC
University Examiner

Jeff Clune, Associate Professor, Computer Science, UBC
University Examiner

Additional Supervisory Committee Members:

Dongwook Yoon, Associate Professor, Computer Science, UBC
Supervisory Committee Member

Abstract

Freehand sketching is a fast and intuitive way for artists to communicate visual ideas, and is often the first step of creating visual content, ranging from industrial design to cartoon production. As drawing tablets and touch displays become increasingly common among professionals, a growing number of sketches are created and stored digitally in vector graphics format. This trend motivates a series of downstream sketch-based applications, performing tasks including drawing colorization, 3D model creation, editing, and posing. Even when stored digitally in vector format, hand-drawn sketches, often containing overdrawn strokes and inaccurate junctions, are different from the clean vector sketches required by these applications, which results in tedious and time-consuming manual cleanup tasks. In this thesis, we analyze the human perceptual cues that influence these two tasks: grouping overdrawn strokes that depict a single intended curve and connecting unintended gaps between strokes. Guided by these cues, we develop three methods for these two tasks. We first introduce *StrokeAggregator*, a method that automatically groups strokes in the input vector sketch and then replaces each group by the best corresponding fitting curve—a procedure we call sketch consolidation. We then present a method that detects and resolves unintended gaps in a consolidated vector line drawing using learned local classifiers and global cues. Finally, we propose *StripMaker*, a consolidation method that jointly considers local perception cues from the first method and connectivities detected by the second method. We further integrate observations about temporal and contextual information present in drawing, resulting in a method with superior consolidation performance and potential for better user interactivity. Together, this work identifies important factors in humans' perception of freehand sketches and provides automatic tools that narrow the gap between the raw freehand vector sketches directly created by artists and the requirements of downstream computational applications.

Lay Summary

Artists often start content creation by freehand sketching. These days, artists frequently sketch digitally using drawing tablets and touch displays and then use software to develop the initial sketch. Unfortunately, these initial sketches often cannot be directly processed by software: sometimes artists use several close, repeated lines to indicate an actual line; and sometimes the lines the artists draw do not meet exactly. This thesis solves these two problems by investigating how humans mentally see sketches without being confused by these busy and inaccurate lines, and building tools so computers can do the same tasks. We compare the result generated by each of our methods against results created by humans and alternative methods. These comparisons show that our methods generate results closer to human results than other methods.

Preface

This thesis presents three collaborative research projects. All conducted user studies were under the approval of the University of British Columbia (UBC BREB Number H16-02320).

A version of Chapter 3 has been published as:

- Chenxi Liu, Enrique Rosales, and Alla Sheffer. "StrokeAggregator: Consolidating raw sketches into artist-intended curve drawings". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2018).

The key ideas formed through discussions between myself and Professor Alla Sheffer. I implemented the method and conducted all experiments. Enrique Rosales implemented and conducted all user studies with guidance from Professor Alla Sheffer, created most figures (except for Figure 3.2, 3.6, 3.8, 3.10, 3.11, 3.12, 3.13, 3.14, and 3.16, which are made by myself), and created the video with the help of Silver Burla. I wrote the initial draft; Nicholas Vining helped with paper editing and proofing; Professor Alla Sheffer wrote the final version of the manuscript. I presented the paper at SIGGRAPH 2018.

A version of Chapter 4 has been published as:

- Jerry Yin[1], Chenxi Liu[1], Rebecca Lin, Nicholas Vining, Helge Rhodin, and Alla Sheffer. "Detecting viewer-perceived intended vector sketch connectivity". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2022).

Most ideas of the binary classifier (Section 4.4) originated from discussions between Jerry Yin and Professor Alla Sheffer; most ideas of the overall method pipeline (Section 4.3, except for "Primary Junctions Classification") were developed by myself and Professor Alla Sheffer. Helge Rhodin was involved in discussions and provided suggestions. Jerry Yin coded most of the underlying facilities and the binary classifier; I coded most of the overall method pipeline. The data was created, collected and curated by myself and Jerry Yin. Jerry Yin and I collaboratively conducted the experiments. Rebecca Lin implemented and conducted the

[1] Joint first authors.

junction perceptual validation study with guidance from Professor Alla Sheffer. I implemented and conducted the comparison study with guidance from Professor Alla Sheffer. The figures were created collaboratively by myself (Figure 4.1, 4.3, 4.5, 4.12, 4.15, 4.16, and 4.17) and Jerry Yin (the remaining figures). Jerry Yin wrote the initial draft for the method related parts; I wrote parts of result related sections; Nicholas Vining helped with paper editing and proofing; Professor Alla Sheffer and Helge Rhodin wrote the final version of the manuscript. I made conference videos with the help of Nicholas Vining and presented the paper at SIGGRAPH 2022.

A version of Chapter 5 has been accepted by SIGGRAPH 2023 as:

- Chenxi Liu, Toshiki Aoki, Mikhail Bessmeltsev, and Alla Sheffer. "StripMaker: Perception-driven Learned Vector Sketch Consolidation". ACM Transactions on Graphics (Proceedings of SIGGRAPH 2023).

The initial ideas originated from discussions between Professor Alla Sheffer and Mikhail Bessmeltsev. Toshiki Aoki created an early prototype and conducted preliminary experiments. The ideas were further developed by myself, Mikhail Bessmeltsev, and Professor Alla Sheffer. I implemented the method using the prototype as a reference and conducted all later experiments. Luciano Silver Burla collected raw data and I collected data annotations using a GUI created by Dave Pagurek van Mossel and modified by Toshiki Aoki. I implemented and conducted all user studies. I created most figures (except for Figure 5.2 made by Matias Bofarull Oddo, Figure 5.5, 5.7, 5.8, 5.10, and 5.11 made by Professor Alla Sheffer), and made the video with the help of Nicholas Vining. I wrote notes for the manuscript; Professor Alla Sheffer and Mikhail Bessmeltsev wrote the final version of the manuscript.

Table of Contents

List of Tables

List of Figures

Acknowledgements

The research of this thesis was made possible by the unwavering support and encouragement of my family, friends, and colleagues, whose constant motivation and guidance inspired me to pursue my academic goals.

First, I would like to express my thanks to my supervisor, Alla Sheffer, for her invaluable guidance and support throughout my doctoral journey. Alla has taught me important lessons on how to verify, demonstrate and present one's research, which I am very grateful for.

I would like to thank my other mentor figures: Mikhail Bessmeltsev, Nicholas Vining, Helge Rhodin, Michiel van de Panne and Dongwook Yoon, for being patient and spending precious time on giving me help and feedback. I would like to thank my lab-mates, Enrique Rosales, Jerry Yin, Rebecca Lin and Dave Pagurek van Mossel for generously co-authoring papers with me, as well as other members of my group for urgent help before many deadlines. I also want to send my thanks and best wishes to Toshiki Aoki for being a mentee of an inexperienced mentor like myself. Special thanks to study participants, artists and reviewers whose names are hidden by anonymity; without their selfless help, my research would not have been possible.

I would like to thank all my friends. Spending time with you folks always helps me recover from hitting obstacles: Matthew Chun for anime watching and organizing group outing; Shayan Hoshyari for interesting discussions and constant friendship since the beginning of my study; Jim McCann for extending his mentorship ; Silver Burla and Jonathan Griffin for being members of our regular Friday gathering; Hanjun Zuo for always being there (virtually).

Last but not least, I would like to thank my parents, Ping Liu and Jia Na, for raising me to be an independent person and for their support, especially during the pandemic years.

I acknowledge that this thesis was completed while I worked at UBC, which is located in the traditional, ancestral, and unceded territory of the xʷməθkʷəy̓əm (Musqueam) people.

Chapter 1

Introduction

(a)

(b)

(c)

Figure 1.1: The evolution of digital drawing tablets. (a) The RAND tablet (1964). (b) Various Wacom drawing tablets and a pen and touch display (photo taken in 2012). (c) An artist draws digitally using a pen and touch display (photo taken in 2019). Source: (a) Gwen Bell, Computer History Museum; (b,c) ©David Revoy under CC BY 4.0.

Freehand sketching is an intuitive way for artists to quickly create and communicate visual content. Traditionally, sketching is done by a single artist with a pen on a piece of paper. Since the advent of computing, devices for digital sketching have evolved rapidly (Figure 1.1), from Davis and Ellis's pioneering digital graphics tablet for natural handwriting, the RAND tablet [26] (1964), to various commercial drawing tablets attached to computers with monitor displays (the iconic Wacom Intuos was released in 1998), to pen and touch displays (the Wacom Cintiq was released in 2001; Apple released the Apple Pencil together with iPad Pro in 2015). As digital drawing devices become user-friendlier and more affordable, digital sketching has become increasingly common among professionals working in

a variety of industries, such as industrial design and paperless cartoons. For these different applications, sketches serve blueprints and need to be further processed by downstream software. Even when stored digitally, freehand sketches, often containing overdrawn strokes and inaccurate junctions, are different from the clean vector sketches required by these applications. To make these applications work, artists carefully clean up input sketches by hand—a tedious and time-consuming task that has not changed since paper-and-pen times.

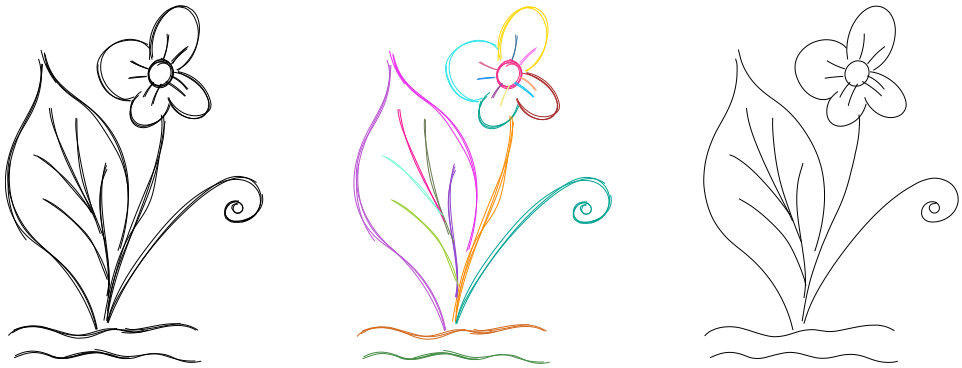
To narrow the gap between what artists sketch and what downstream applications require, we aim to develop automatic methods that tackle the overdrawn strokes and inaccurate junctions that are unavoidable in freehand digital sketches. We refer to these issues as the consolidation and connectivity problems respectively.

Understanding artist intentions is a complex task, as their intentions may be influenced by many factors, such as high-level aesthetic choices and low-level muscle memory. Instead, as strongly suggested in prior literature that artists strive to create drawings that are easily understood by viewers [45, 94, 109], we use these two well-correlated concepts interchangeably in this thesis and aim to develop algorithms that mimic viewers' perception. By taking this approach, we can leverage the rich collection of criteria and principles provided by Gestalt psychology [104], which studies how viewers perceive multiple objects as a whole. Furthermore, we can investigate data-driven methods that learn from viewer-annotated data to tackle these problems.

Before we explore further, we need to specify the format of sketches that we want to study. Digital sketches can be stored in raster and vector format. Sketches in vector format contain more information about inputs than those in raster format, such as the definition of a stroke, local tangents, *etc*., and are feasible to capture using add-ons to common raster drawing software [70]. We thus select vector sketches as our target input and the scope of all methods introduced in this thesis.

1.1 Sketch Consolidation

When creating an initial sketch, artists use overdrawing to correct or refine earlier strokes, emphasize specific curves, and break down hard to draw long and com-



(a) (b) (c)

Figure 1.2: A raw sketch with overdrawn strokes and the corresponding perceived clean sketch. Human observers consistently view the raw sketch (a) as a composition of strips (indicated by different colors) (b). Each strip is seen as depicting a clean aggregate curve and together they compose a clean line drawing (c).

plex curves into shorter, easier to sketch strokes. When presented with such a *raw* sketch, human viewers effortlessly perceive a cluster or a *strip* of strokes as jointly depicting a single artist *intended* curve, called an *aggregated* curve (Figure 1.2). While this mental process is near instantaneous, manually annotating or retracing sketches to communicate this intended mental image to sketch based applications is highly time consuming. We define this perceptual process of grouping strips of strokes and replacing each strip by an aggregated curve as *sketch consolidation*. The computational methods that are designed to mimic human viewers and automate this process are called *algorithmic sketch consolidation*.

Formally, an input sketch consists of n raw strokes, $\mathcal{S} = \{S_1, \cdots, S_n\}$; a sketch consolidation method seeks: (1) a clustering $\mathcal{C} = \{C_1, \cdots, C_m\}$ partitioning the input sketch $\mathcal{S}$ into strips, such that the clustering is as close to the human-perceived result as possible and (2) a clean sketch containing the aggregate curves corresponding to the resulting stroke clusters.

Chapter 3 and 5 rely on perceptual cues for clustering, and this strategy of measuring these cues forms the foundation of both chapters. To give readers a summary, we describe our measurement strategy and summarize the cues as follows.

To compare an arbitrary pair of strokes for measurement, we find correspon-

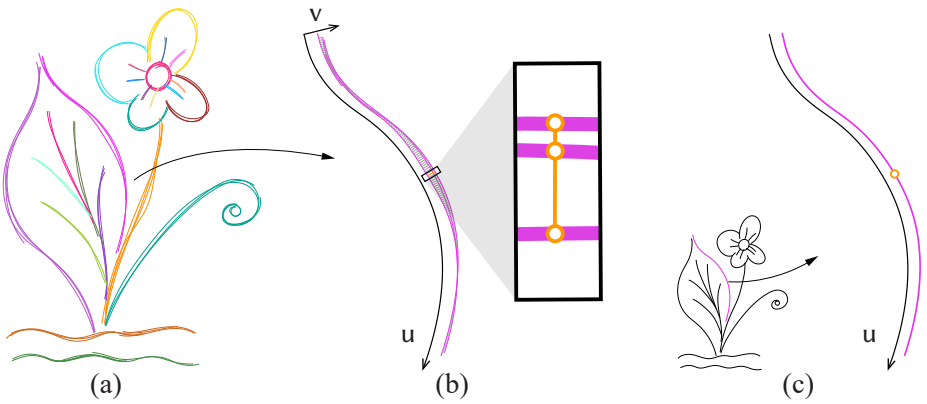


Figure 1.3: Stroke strip parameterization. Each stroke strip in a raw sketch (a) is associated with its corresponding aggregate curve (c). Each point on a raw stroke (orange in zoom-in, b) is associated with a point closest to it on the aggregate curve (orange, c). Points on different raw strokes that are associated with the same aggregate curve point are corresponded (orange in zoom-in, b) and share the same u value.

dences between points from these two strokes via 1D parameterization (Figure 1.3). If two strokes belong to the same strip, the strip would have a corresponding aggregate curve (Figure 1.3b,c). It is trivial to associate any point on these two strokes with a single point on the aggregate curve by finding the closest corresponding point. This corresponding point on the aggregate curve is parameterized because the aggregate curve has a 1D mapping from a parameter u to an arbitrary position on the curve ($\mathbb{R}^2$ in our setting). In this way, we correspond points from different strokes if they are associated with the same aggregate curve point; in other words, if the two points share the same u value in the same isoline with respect to the 1D parameterization. Note that even if these strokes do not belong to the same strip, one is still able to find such correspondences. These correspondences are computed via a simple fit-and-project procedure in Chapter 3 and via a more complicated parameterization method [79] in Chapter 5. These correspondences also help us define the region of interest for stroke-stroke comparison: two strokes are only comparable in the interval where points from both strokes exist, which we refer to as the *side-by-side section*. A value quantifying a cue can be computed by locally measuring pointwise within an isoline and then integrating along the

Table 1.1: Perceptual cues used for vector sketch consolidation.

Category	Cue	Chapter
Local	Absolute distance	Chapter 5
	Density (relatively distance, proximity)	Chapter 3,5
	Angle	Chapter 3,5
	Narrowness	Chapter 3,5
	Evenness	Chapter 5
	Parameterization distortion	Chapter 5
Contextual	Connectivity	Chapter 5
	Global precision	Chapter 5
Temporal	Drawing order of strokes	Chapter 5

side-by-side section. This strategy is extended to measure multiple strokes against multiple strokes and to measure a strip internally by considering all local point combinations.

We apply the same measurement strategy to a set of cues summarized in Table 1.1. Cues are classified into three categories.

Local cues. Cues in this category are measured solely based on the geometry of strokes that are hypothesized to belong to the same strip.

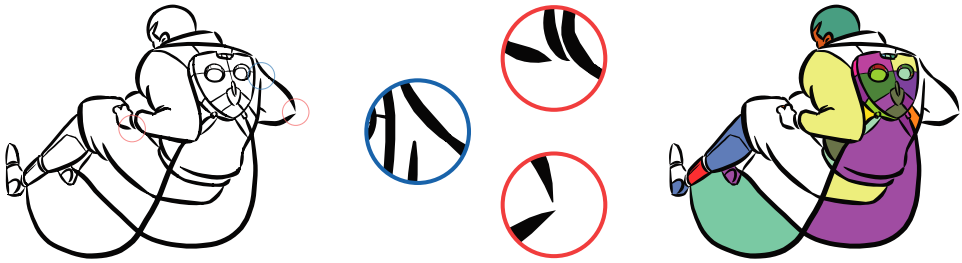
Contextual cues. Cues in this category are measured based on multiple strips. Note that a contextual cue can involve a strip and another strip immediately adjacent or another strip in the same input sketch regardless of positional relation.

Temporal cues. This cue is about the drawing order of strokes independent of the two geometric cues above.

In Chapter 3, we introduce *StrokeAggregator*, a method that automatically consolidates raw vector sketches via grouping and fitting. This method applies purely local cues that are quantified by conducting individual perception studies. In Chapter 5, we expand this basic consolidation method by accounting for more cues that are local, contextual and temporal. Since conducting an individual study per perceptual cue quickly becomes too costly as the number of cues increases, we apply a learning based approach and by integrating local classifier with global refinement based on contextual and temporal cues, we overcome the data sparsity issue. This

new method, *StripMaker*, achieves comparable performance to manual consolidation. In our comparative studies participants preferred our results by a 52% margin over those of StrokeAggregator, the closest algorithmic alternative.

1.2 Sketch Connectivity



(a) (b) (c)

Figure 1.4: Unintended gaps in a typical freehand sketch. A freehand sketch (a) contains numerous unintended gaps (b,blue) that are in the similar scale as intended gap (b,red). Bucket-filling each region a different color indicates that multiple adjacent regions are incorrectly merged (white indicates the background) (c).

When curves meet, they form *junctions*, defining the topological relationship or the *connectivity* of a sketch. The connectivity provides necessary information for plenty of applications; for example, junctions define occlusions for sketch-based 3D modeling [11, 12], serve as a basic editable element in advanced vector graphics editing and modeling [24, 25], and define regions which is essential for drawing colorization [38, 82, 88, 101]. However, artist drawings are inherently imprecise [55], and routinely contain unfinished strokes that artists intend to intersect other strokes, but that stop short of doing so as shown in Figure 1.4. As reported by Yan et al. [112], artists continue to leave *unintended gaps* between strokes even when explicitly asked to draw as precisely as possible. As a result, in practice, connecting unintended gaps is separated from sketching and often treated as a tedious manual preprocessing step of colorization, called flatting, involving repeated zoom-in checks and trial-and-error operations and in the scenario of webtoon production, taking roughly 50% of work time for the whole colorization process [113]. Furthermore, a better understanding of connectivity is beneficial to consolidation as

consolidation and connectivity are interconnected in the sense that viewers' decision of perceptually grouping overdrawn strokes is influenced by junctions formed by potential intended curves [69].

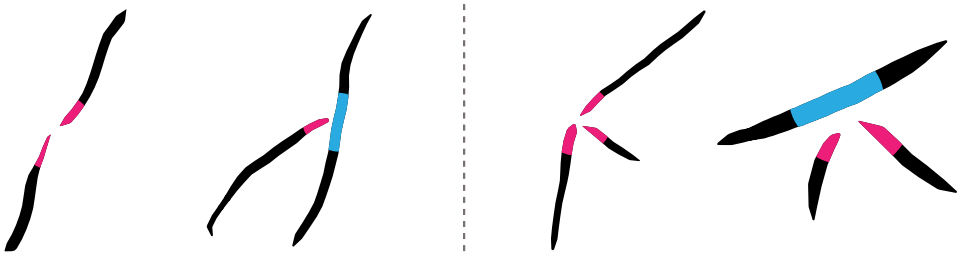


Figure 1.5: The categories and diverse configurations of junctions. Binary junctions are formed by pairs of strokes: end-end junction (a), T-junction (b). High-valence junctions involve multiple dangling endpoints and strokes, in both end-end and T- configurations (c,d).

We categorize junctions into two categories: *binary junctions* and *high-valence junctions* (Figure 1.5). Binary junctions are formed by pairs of strokes. They have two types: end-end junctions that are formed by joining the endpoints of two strokes; and T-junctions that are formed by connecting a dangling stroke endpoint to the middle of another stroke. As the name implies, high-valence junctions involve multiple dangling endpoints and strokes, in both end-end and T- configurations. The existence of high-valence junctions complicates the problem as there are an arbitrary number of configurations of them while each configuration often only occurs occasionally in a sketch.

We discuss this issue in detail in Chapter 4. We further propose a method in Chapter 4 that detects such unintended gaps in consolidated vector line drawings using learned local classifiers in an inference framework that incorporates global cues. We demonstrate our method on a diversity of line drawings in the wild for the colorization application and apply our connectivity detection method to provide contextual cues for consolidation in Chapter 5.

1.3 Contributions

We explore two closely related problems arising in processing freehand vector sketches: consolidation problem and connectivity problem. Our work identifies important factors in humans' perception of freehand sketches, provides, and validates automatic tools that narrow the gap between the raw freehand vector sketch directly created by artists and the requirements of downstream computational applications. Taken together, the automatic tools presented in this work form a pipeline capable of consolidating a rough sketch, then equipping the resulting clean line drawing with connectivity. The final result is then ready for further processing such as colorization (Figure 1.6).

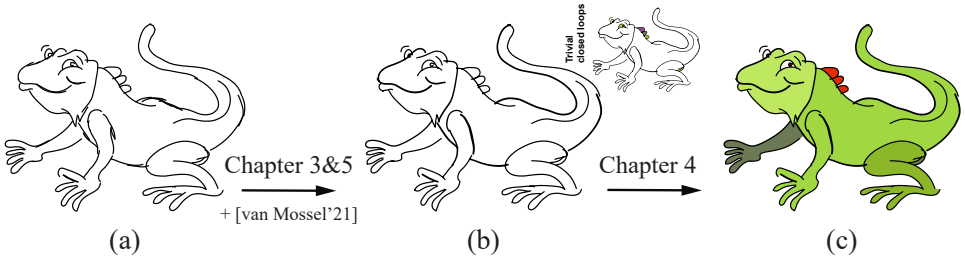


Figure 1.6: A freehand vector sketch processing pipeline. A raw input sketch (a) is consolidated (b) then connected (c) for colorization. Note the closed loops trivially detected on (b) are shown in the inset. Source: © Rami Alsafadi.

Our detailed contributions are as follows.

- In Chapter 3, we present a method that consolidates a raw vector sketch into a clean vector line drawing. The resulting consolidated drawings are validated to be consistent with viewer perception by our comparison studies and evaluated to have path quality most similar to manual results among raster and vector consolidation methods at the time by a recent benchmark [112]. This chapter established a measurement strategy and a set of local cues that serve as foundation for Chapter 5.

- In Chapter 4, we introduce a method that extracts viewer-perceived stroke connectivity from inexact freehand vector drawings. We demonstrate our method on diversely sourced inputs, including actual cartoon movie draw-

ings, doodles, and consolidated drawings. The connectivity studied in this chapter is considered as a contextual cue in Chapter 5.

- In Chapter 5, we propose a more advanced consolidation method that builds upon earlier chapters and utilizes further observations about temporal persistence and global context of drawing. This method exhibits superior consolidation performance and potentials for better user interactivity.

Chapter 2

Related Work

Freehand sketching digitally on a drawing tablet or a pen-and-touch display has been gradually becoming a common practice and motivated research about further processing these digital sketches into refined visual contents, such as colored drawings and 3D models. In this chapter, we first provide background about how an artist sketch and how visual contents are created at industrial scale. We then discuss research topics that share similarities with ours yet have different goals, as well as methods that serve as downstream applications taking vector sketch as input. Finally, we examine prior work about the consolidation problem in raster and vector settings, and the connectivity problem.

2.1 Artist Sketching Practice

Figure 2.1: A typical freehand drawing with overdrawing and unintended gaps. Repeatedly overdrawing existing strokes refines the initial curve (a, b, red). Overdrawing is used to emphasize, in this example, to distinguish blue and cyan strokes are from two strips (a, b, blue, cyan). Overlapping short strokes forms a long and complex curve (a, b, green). Even in this carefully created cleanup line drawing (c), unintended gaps remain (d). Please zoom in to see full details.

Artists frequently apply overdrawing strategies in various scenarios (Figure 2.1): they gradually refine intended imagery by layering new strokes on top of existing scaffolding strokes; they overdraw strokes for emphasis, and they often depict long and complex curves with multiple shorter and simpler strokes that are partially overlapping [2, 32]. Apart from overdrawing, inaccurate junctions are another common feature of freehand sketches. While overdrawing is a strategy which artists consciously apply, unintended gaps are left unintentionally by artists (Figure 2.1). This implies that while a skillful artist may be able to avoid overdrawing when sketching carefully, artists continue to leave unintended gaps between strokes even when explicitly asked to draw as precisely as possible [112]. Although these raw sketches by themselves convey visual contents efficiently, they need to be converted into clean, accurate lines with closed gaps for production in industries, such as paperless cartoon animation and industrial design.

Paperless cartoon production is similar to traditional 2D cartoon production, except that all drawing and colorization processes are performed digitally. A cartoon project is conducted in two top-level stages: preproduction and production; the cleanup is critical for both stages [108]. In the preproduction stage, necessary preparations are made. One crucial asset during preproduction of cartoon animation are storyboards, which provide a first visual interpretation of the script that then guides further developments in production. To serve as a reference for multiple stakeholders, including the main production team, sub-contracting studios, and even the legal department, the final storyboards need to be as fully rendered as possible and commonly require a special role, the cleanup storyboard artist, dedicated to the job. In the production stage, the cleanup is necessary for both foreground characters and background assets to ensure that the lines are consistent and ready for colorization. Since the workload is heavy, this production step is executed by a team of artists, referred to as the 2D cleanup animation team.

Sketches used for industrial design are also developed through multiple steps, changing from concept sketches to presentation sketches [44]. Concept sketches contain construction lines, such as scaffolds, axes, and projection lines, to assist designers building the envisioned 3D shapes using 2D medium. While concept sketches serve as a workspace for designers, presentation sketches are used to demonstract design products to clients and to guide the subsequent 3D modeling

and thus require cleanup. Despite the usage of digital sketching and colorization tools, the common practice in both these examples is to clean up by hands, which is tedious and time-consuming.

2.2 Vector Sketch Processing

Multiple tools target processing of artist sketches, facilitating tasks such as colorization [38, 82, 88, 101], shading [36, 94], 3D editing [52], modeling [66, 109] and manufacture [65]. Similarly to applications in industry, sketch-based methods in academia require clean vector line drawings with accurate junctions.

Before we discuss consolidation in detail, we distinguish two related problem settings: *sketch beautification* and *sketch simplification*. Despite that results from these types of methods could be fed into the same kind of downstream applications, there is a difference: the goal of consolidation is to preserve contents as truthfully as viewers perceive while these beautification and simplification methods alter the visual contents to achieve their respective goals.

Sketch beautification methods assist artists in creating cleaner and more aesthetic drawings by modifying the input curve geometries. Curve fitting based beautification methods target point sequences directly captured by the drawing input device and fit primitive chains as strokes to overcome inaccuracies in capturing and sketching [7, 83, 102]. Pavlidis and Van Wyk [84] proposed a clustering based method designed for diagrams that fits line segments to stroke groups unlike the fidelitous aggregate curves in the consolidation setting. Later methods [18, 37, 51, 74] support more primitive types (e.g., segments, arcs) and enforce spatial relations (e.g., positional, angular) between primitives via constraints, which is still less flexible than consolidation methods.

Sketch simplification methods reduce details in a drawing by representing the original sketch using a subset of the input strokes. This class of methods are applied to reduce visual clutter in detailed artist drawings [41, 75] or to save computation time in non-photorealistic (NPR) rendering [56, 107]. These methods are different from consolidation methods because they do not compensate for the deleted strokes and moreover, the ones for NPR leverage 3D information not available in our problem setting.

2.3 Raster Sketch Vectorization and Consolidation

Line drawing vectorization methods convert a raster sketch into a vector line drawing as accurately as possible, either automatically or semi-interactively. Vectorization methods can be further categorized based on whether the method simultaneously consolidates sketches with overdrawing. For vectorization methods targeting clean raster line drawings, they can acquire the consolidation functionality by preprocessing via a group of methods that consolidate fully in raster space either automatically or interactively.

Clean sketch vectorization methods focus on recovering clean vector curves from pixels in a clean raster sketch. As with vector consolidation, challenges for clean sketch vectorization methods lies in distinguishing near-overlapping strokes and reconstructing junctions accurately. Methods in this category often start by identifying the content pixels by either filtering [20, 28, 29, 77] or simply thresholding [6, 10, 86]; then follow by constructing an initial graph by thinning and connecting [28, 29, 77], tracing a tangential [6, 20] or polyvector field [10], or linking detected keypoints [86]; finalize by optimizing graph topology with several methods focusing on junction topology as well as optimizing the graph geometry [20, 77, 86]. Recent methods rely on neural network with different focuses. Guo et al. [46] propose a method specialized in junction topology. Bhunia et al. [13] explore adapting self-supervised learning strategies. Several neural network based methods target domain specific data including fonts [60], technical drawings [31], and CAD sketches [71].

To handle raster inputs containing overdrawing, clean sketch vectorization methods need to be applied following raster-to-raster consolidation methods. Raster-to-raster consolidation methods transform a raster sketch with overdrawing to a clean raster line drawing. Early raster-to-raster consolidation methods [19] applies flow oriented filter with a fixed kernel size which may not apply to different levels of details within a sketch. Recent raster-to-raster consolidation methods based on convolutional neural network work either automatically [96, 97, 111] or interactively [98]. Apart from overdrawing, these methods also handle raster sketches captured in poor conditions, such as, non-uniform canvas textures, uneven lighting, and faint stroke colors, thanks to the image processing nature. For more deep

learning based methods, see the survey by Xu et al. [110] for details.

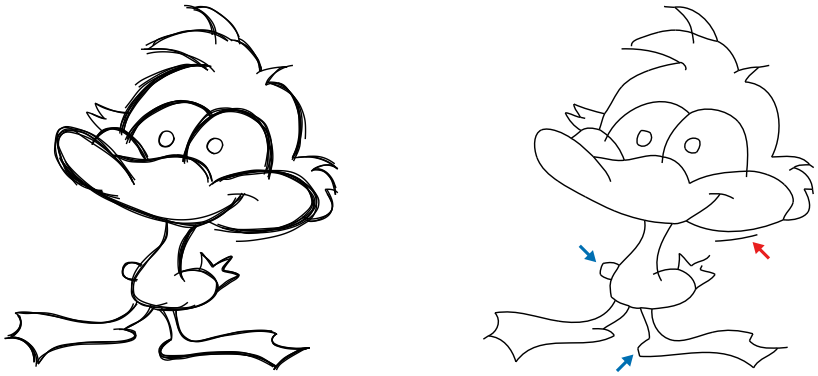
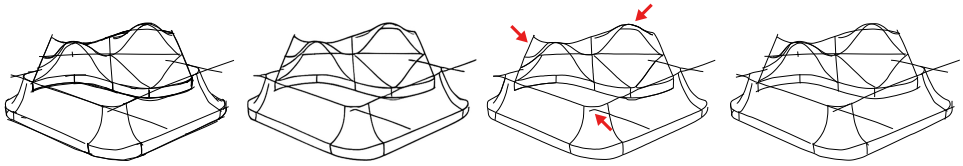


Figure 2.2: A single level of detail used by Stanko et al. [100] does not fit all local contents (red arrow). Their integer grids cause artifacts when fit to high-curvature smooth curves (blue arrows).

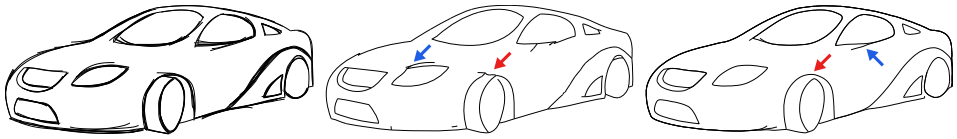
The raw sketch vectorization methods consolidate and vectorize at the same time. Most methods [21, 35, 80, 117] detect and iteratively merge regions, then extract region boundaries as an initial graph. Following these steps, these methods simplify the graph topology and optimize the geometry into the final vector result. Parakkat et al. [82] use a semi-automatic approach that require user to merge initially detected regions. These region based methods tend to over-simplify the inputs and cannot naturally process open curves. Note that these region based methods recover junctions between regions along side consolidation and the details of region detection strategies are further discussed in Section 2.5. Stanko et al. [100] introduce a method that formulate target clean curves as integer grids—a concept borrowed from quad meshing. This integer grid method still heavily depends on a global kernel size, which may not suit all detail levels locally in a sketch, and the outputs often contain artifacts when a grid corner is associated with a high-curvature smooth curve (Figure 2.2). Mo et al. [73] propose a deep learning based method that trace raw raster sketch by clean vector curves. However, their traced outputs contain almost exclusively short, broken curves rather than long, consistent curves as intended that are meaningful for the downstream applications.

The vector consolidation problem can be converted into the raster consolidation



(a) Input (b) [Xu et al. 2019] (c) [Xu et al. 2019]→ [Puhachov et al. 2021] (d) StripMaker

Figure 2.3: The raster-to-raster consolidation methods, e.g., Xu et al. [111], have difficulties handling multi-way junctions (b,c) that are trivial for vector space consoldiation method (d). Note that the consolidated raster output (b) needs to be further processed by a clean sketch vectorization method, e.g., Parakkat et al. [82], to obtain the final vector result (c).



(a) Input (b) [Stanko et al. 2020] (c) [Xu et al. 2019]→ [Puhachov et al. 2021]

Figure 2.4: Raster-space consolidation methods are significantly affected by input resolutions and due to diverse levels of details within a single sketch, these methods can destroy fine details (red arrows) and leave overdrawing unattended (blue arrows) at the same time.

problem by rasterizing the input sketch. However, solving the problem in this way faces major challenges and remains open as acknowledged by Parakkat et al. [82] and Yan et al. [112]. Vectorization and raster consolidation in general are more challenging than vector consolidation due to reduced information. For instance (Figure 2.3), though the combination of state-of-art raster-to-raster consolidation [111] and clean sketch vectorization [86] generates viewer perceived results on simple inputs, it has issues handling multi-way junctions when the sketch complexity increases; while for vector consolidation, the connectivity is clearly indicated by underneath stroke topologies. In addition, when consolidating vector sketches in raster space the choice of rasterization resolution can significantly impact output quality, and there exists no principled way of choosing the best resolution [112]. As Figure 2.4 shows, while raster-space methods often destroy fine details or leave

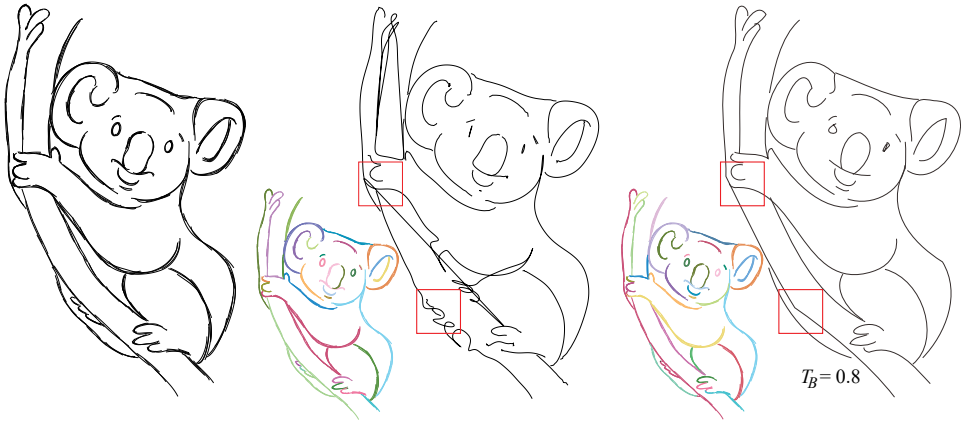
overdrawing unattended, our vector-space method is consistently better at preserving fine details. With little to no cost for digital drawing software to capture strokes in vector format, we believe vector-space method is a more promising direction for solving consolidation problem.

2.4 Vector Sketch Consolidation

Vector sketch consolidation has two subproblems: identifying strips, or groups of strokes perceived as depicting single curves, and fitting the aggregate curve to each strip. Compared to the grouping subproblem, the fitting is less challenging. Early methods first order point samples on a stroke by projecting to the dominant axis [58], or by computing Laplacian spectral embedding [78], then fit a curve to fully ordered points. We introduce a moving least squares based fitting in Chapter 3 that integrate a tangent fitting term in addition to the sole position fitting term used by the earlier methods. Pagurek van Mossel et al. [79] propose a method dedicated to fitting stroke strips that orients strokes, explicitly compute a 1D parameterization using isoline as the basic element, and fit the final curve using position, tangent and an extra curvature smoothing term. Since this state-of-art method robustly provides desired 1D parameterization and fitting curve even given incorrect stroke strips, we use this method for fitting and focus on the grouping subproblem in Chapter 5.

The grouping subproblem remains an open problem [112]. Early sketch analysis and consolidation methods [8, 90, 95] use cues such as absolute proximity, degree of parallelism, and continuation to group strokes. These methods rely on user specified thresholds to determine which strokes to group, requiring per-model tuning. Moreover, the majority of methods [8, 90] measure cues based on one or few pairs of corresponded point samples, e.g., closest points, endpoints and midpoints, which is prone to sensitive to noise unlike the integral based measurement we apply; Shesh and Chen [95] measure cues on all combinations of sufficiently close points, which is computationally more expensive and comparing faraway points is not as meaningful.

Orbay and Kara [78] proposed a neural-network-based method which is the first work using the word "consolidation" and identifying branching as a challenge. Their model learns a pairwise probability function depending on features of angles



(a) Input (b) [Orbay&Karaet al. 2011] (c) [Liu et al. 2015]

Figure 2.5: Vector sketch consolidation remains an open problem. The method by Orbay and Kara [78] does not generalize well (b). The result of Liu et al. [69] is noticeably influenced by the choice of method parameters and is imperfect even after tuning parameter (c).

and distances. Their method then builds initial clusters by thresholding the probabilities and subsequently refines these clusters using an algorithmic branch separation step. In their clustering framework, every stroke pair is examined based on their shortest path which fails to capture the correct positional relationship when two strokes are in two different branches (should be in two clusters) sharing the same stem (thus, the shortest path cannot capture this separation). The method works well when trained and tested on drawings produced by the same artist; but, as the authors acknowledge and shown in Figure 2.5, this method fails to generalize to drawings from multiple sources and usually over-merges significantly.

Liu et al. [69] introduce contextual angle and proximity metrics defined relative to the size of empty spaces, or regions, enclosed by the input strokes. They construct the regions by iteratively merging small regions based on a threshold and in a similar fashion, they merge the raw strokes iteratively with a threshold adjusted based on nearby region sizes. As a result of the usage of regions, their method assumes the input sketches are closed planar maps and thus may have unexpected behaviors when the sketch contains multiple layers or open curves are involved. Additionally, as demonstrated in Figure 2.5, their method requires users to man-

ually adjust two thresholds, without clear semantic meanings, to produce optimal results, which diminishes the method's effectiveness and makes it scale-dependent even though the input vector sketches are scale-free.

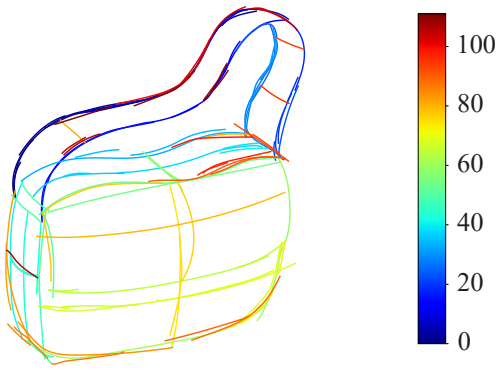


Figure 2.6: Sketch drawing order in color-coding. The drawing order of strokes in this typical sketch shows that strokes depicting the same local content are often drawn consecutively shown in similar colors. Yet it is not always the case as there are few strokes put down out of order indicated by orange and red, perhaps to fulfill the refinement purpose of overdrawing.

Temporal cues are taken into consideration by several incremental drawing systems that consolidate sketches on-the-fly [4, 9, 42] and an interactive stroke grouping system [76]. These incremental drawing systems [4, 9, 42] are designed for 3D sketching, and due to the usage of simple cues similar to those of early consolidation methods, heavily depend on user gestures or click inputs as guidance. Noris et al. [76] take user-drawn guiding strokes as cores and cluster raw input strokes based on their perceptual similarities to the cores and their drawing orders. Unlike the consolidation setting where raw strokes are grouped into strips, this interactive stroke grouping system aims to divide strokes into semantic components, which is higher level than our target use case. Although the grouping strategies of these systems are far from error-proof, they inspire us to utilize temporal cue. Our analysis of manually consolidated inputs shows that around 30% of viewer perceived multi-stroke strips contain strokes which were not drawn consecutively (an example is in Figure 2.6). Validated by this analysis, we design the consolidation workflow to use drawing order as guidance in Chapter 5.

2.5 Sketch Connectivity

The connectivity of a sketch is defined by junctions. As described in Section 2.1, inaccurate junctions are inevitable in freehand sketches. We focus on the inaccuracy caused by unintended gaps in this thesis. Existing methods for gap closure are mostly designed to be used in a semi-automatic setup and can be roughly grouped into two categories: methods that collect user inputs to guide the following automatic gap connection algorithm; and methods that generates an initial gap closed result and then optionally interact with user for corrections. Note that due to the interconnected nature of consolidation and connectivity [69], several consolidation methods mentioned above also tackle the connectivity either in an integrated raster-to-raster framework [96, 97] or as a by-product from the region based strategy [35, 82].

Within the first category, vector graphics creation systems [1, 3, 39] first ask user to specify a gap size threshold then automatically close all gaps below the given value; other methods in vector [76] and raster space [88, 101] fully rely on user to provide gap closure instructions using stroke-based interactions. The first type is prone to errors since the gap size alone is insufficient to correctly close all unintended gaps in a typical sketch with different levels of details. The second type, as discussed by Parakkat et al. [81], tend to be highly sensitive to properties of the initial inputs, requiring significant amount of trials-and-errors to produce a desired outcome. Our method in Chapter 4 is fully automatic and account for a set of local cues apart from the gap size.

The second category of methods attempt to first automatically close gaps and only resort to user correction when necessary. Yet since the initial results often contain artifacts, most methods identify as semi-automatic and consider user correction as default.

Gap closure is studied for the industrial design drawing. Gryaditskaya et al. [45] conducts gap closure as a preprocessing step using a distance threshold defined as a function of stroke width. This heuristic is an improvement over the ones used by vector graphics creation systems [1, 3, 39] but it is still far from sufficient to handle all scenarios. We similarly normalize distances by stroke width but our extended cue set is adequate for the gap closure problem. Other methods [23,

105, 106] restrict to a smaller target data type: wireframe drawings of polyhedra, under the assumption that input drawings contain only straight lines and end-to-end junctions without intentionally dangling endpoints. While these methods also make use of multiple cues, their problem setup is much more limited compared to ours and our method can be applied as-is to drawings of polyhedra.

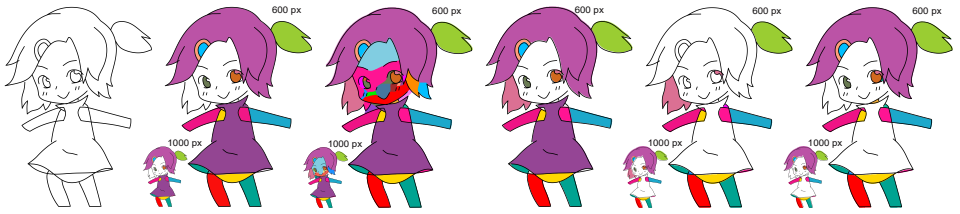
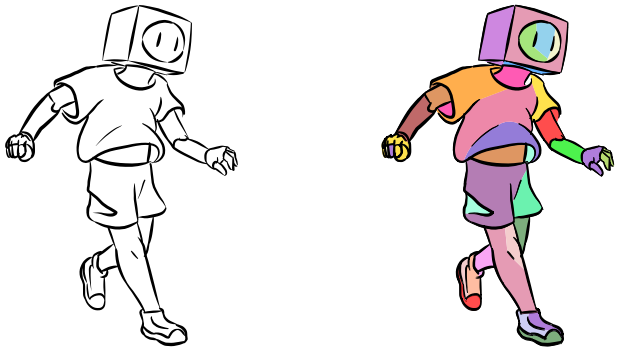


Figure 2.7: Recent raster-space gap closure methods produces sub-par outputs with both unintended junctions and unresolved dangling endpoints on a relatively simple input. Each region is assigned a different color and white indicates the background.

Sasaki et al. [93] propose a raster-to-raster gap closure method. They employ a deep learning based structure similar to the raster-to-raster consolidation methods [96, 97] with the learning objective specified to gap closure. As mentioned earlier, these raster-to-raster consolidation methods can implicitly close gaps while fulfilling the consolidation task. Zhang et al. [116] collect Danbooregion, a large dataset of regions in 5377 raster cartoon drawings annotated by real artists. To achieve this amount of data, they assist artists with a neural network trained on a small subset in a human-in-the-loop annotation fashion. Building upon Danbooregion, Zhang et al. [116] develop a neural network based flat-filling method to color each region in an input raster line drawing with a single color based on a user color scribble. Similarly based on neural networks, Yan et al. [113] create a flat colorization method with a different interaction design. Based on their interviews of professional webtoon creators, they introduce automatically generated neural lines that connect gaps to define initial regions, then their system allows artists to rewrite neural lines in the following fine colorization stage. All these raster-to-raster methods suffer from the resolution dependency issue similarly to the consolidation problem setup.

Determining intended junctions via region detection on raster input is a com-

mon strategy. Several methods apply a combination of trapped-ball initialization and subsequent diffusion locate closed regions in raster line drawings [35, 117]. Although this is a fully automatic procedure, the authors acknowledge that when using their default parameters the method may often fail to produce results aligned with user expectations and thus require per-input tuning or detailed user correction. Alternatively, other methods detect regions via Delaunay triangulation [81, 82], or via extending curves with dangling endpoints [38]. These methods also tend to produce redundant regions and since the target application is colorization, they rely on users to duplicate colors for redundant regions. All these raster based methods, despite the support of a fully automatic mode, depend on user corrections for desired results and frequently mishandle open curves within a region like in the consolidation setup. We leverage information provided by vector data to address a more general problem of locating and closing unintended gaps, including both gaps along region boundaries and between strokes internal to such regions.



(a) (b)

Figure 2.8: Jiang et al. [54] connects most of dangling endpoints in the input (a) yielding a great number of redundant regions while still mishandling detailed areas, e.g., fingers in this example (b).

Jiang et al. [54] propose a vector-space gap closure method. They hold the assumption that the vast majority of dangling endpoints are unintended, leading their method to often close gaps that viewers perceive as intentional (Figure 2.8). Therefore, like the region detection based methods above, they propose a semi-manual interface that enables users to correct such undesirable connections.

Chapter 3

StrokeAggregator: Artist-Intended Vector Sketch Consolidation

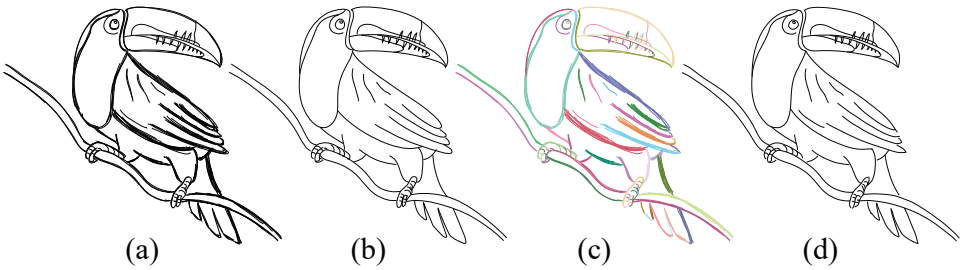


Figure 3.1: Stroke consolidation: (a) a raw, vector format, sketch; (b) manually consolidated clean curve drawing; (c) algorithmically clustered strokes and (d) consolidated curves. Our output curve set (d) is of similar quality to the manually generated one (b). Please zoom in online to see image details. Raw sketch: © Enrique Rosales. Manual consolidation: © Elinor Palomares.

Freehand line drawing provides a natural avenue for artists to quickly communicate shapes, ideas and images. When creating line drawings from scratch, artists often employ oversketching, using groups of multiple *raw* strokes to depict their intended, *aggregate*, curves (Figure 3.1a). Human observers can easily visually parse, or *consolidate*, these drawings by mentally replacing clusters of raw strokes with their corresponding aggregate curves. To create more refined, colorized, or shaded drawings, or to use these sketches as inputs to editing or modeling software, artists typically perform manual stroke consolidation by retracing the drawing and replacing raw stroke clusters with carefully drawn corresponding aggregate curves (Figure 3.1b) [2, 32]. In this chapter, we present *StrokeAggregator*, an algorithm that generates consolidated drawings of comparable quality to those generated by artists (Figure 3.1d) and lays the foundation for further improvements in Chapter 5.

3.1 Introduction

Given the prevalence of tablets and other pen-sensitive displays, artists can easily create line drawings within a computer program and have the strokes recorded in vector form. These vector drawings contain more information about artist intent than their raster counterparts, motivating us to use vector format sketches as input. Algorithmic consolidation of both raster and vector drawings remains an open challenge: existing methods require parameter tuning and frequently fail to produce satisfactory results. Manually generating consolidated drawings from either raster or vector sketches requires expertise and time. An artist required nearly thirty minutes to create the consolidated drawing in Figure 3.1b; our algorithm generated a comparable quality consolidated drawing in five minutes.

We identify and describe the core factors that lead viewers to mentally consolidate raw strokes in line drawings in Section 3.2.1. Intuitively, we expect aggregate curves to correspond to distinct, narrow clusters of roughly evenly spaced strokes (Figure 3.2). We expect strokes within the same cluster to be *angle compatible*, or to be roughly parallel along their nearby side-by-side sections (Figure 3.2a), and expect strokes within the same cluster to be roughly evenly spaced; we expect this internal spacing to be significantly smaller than the closest distance from strokes within the cluster to all partially parallel strokes outside it (Figure 3.2b). Perception literature refers to this spacing-based property as *relative proximity* or relative distance [104]. Our challenge is to algorithmically account for these properties. Relative proximity assessment is complicated by the fact that pairwise distances between strokes can vary at different points along them, resulting in different spacing along different side-by-side stroke sections (Figure 3.2d). Human observers mentally separate stroke *branches* once the spacing between them becomes visibly uneven. Algorithmically replicating branch separation requires local analysis of spacing between side-by-side strokes.

Our algorithm is based on two key insights. We note that for nearby strokes, angular compatibility provides a strong negative cue: nearby strokes with sharply varying tangent directions are unlikely to describe the same aggregate curve. We also note that given a group of angle compatible strokes, we can successfully assess if these strokes form an internally consistent cluster with respect to the principles

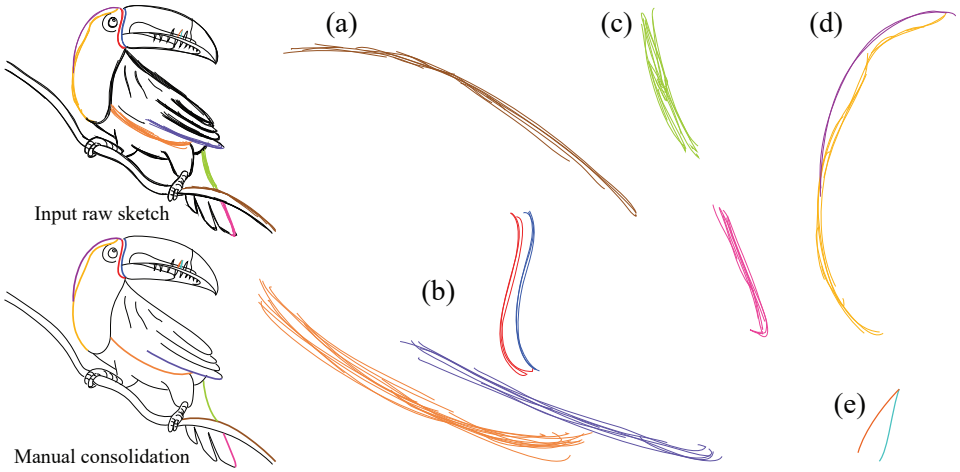


Figure 3.2: Manual consolidation examples (color shows stroke grouping): (a) a typical cluster consists of strokes which are angle compatible, or roughly parallel along their side-by-side portions; (b) within each cluster, strokes are roughly evenly spaced and the internal distance is much smaller than the inter-cluster distance. Note that the absolute distance between the top red and blue clusters is roughly the same as the internal absolute distance of the orange cluster; however, humans treat the two differently based on relative proximity rather than absolute distance; (c) disjoint Gestalt continuous clusters define separate aggregate curves; (d) connected branches with uneven internal density define separate curves; (e) width to length ratio, or cluster narrowness impacts viewer choices. Here, strokes are viewed as separate despite satisfying all other grouping cues. Raw sketch: © Enrique Rosales. Manual consolidation: © Elinor Palomares.

above. We use these observations as the basis for a coarse-to-fine gradual clustering framework (Section 3.3). We form initial coarse clusters based on angular compatibility between strokes and refine those based on average pairwise distance between them, to form clusters of roughly evenly spaced strokes (Section 3.3.1). We then perform local analysis of intra-cluster stroke spacing to detect and separate stroke branches (Section 3.3.2). In the presence of perceptual ambiguities in both stages, we separate groups of strokes absent clear evidence that the combined cluster satisfies all necessary perceptual criteria. Our final step (Section 3.3.3) relies on the internal consistency of the computed clusters to resolve ambiguities and to merge clusters which are both angle and spacing compatible. Finally, we fit a

shape preserving aggregate curve to each resulting cluster (Section 3.4). We rely on the same set of perception driven parameters across all inputs throughout the entire process; we derive their values from perception literature, and customize them to our setting via targeted human perception studies (Appendix B.1).

Key to our approach is the ability to consistently assess perceptual compatibility between, and within, groups of strokes; and to use the same metrics across different configurations of overdrawn strokes that artists may draw (Figure 2.1, see surrounding text for description). We provide this unified framework by computing a common parameterization for each group of assessed strokes based on their corresponding aggregate curve.

In summary, our overall contribution is the first sketch consolidation method that reliably generates output curve networks that are consistent with viewer expectations without the need for any parameter tuning. We achieve this goal by leveraging a combination of perceptual criteria and insights about artistic practices, which guide our clustering framework and help resolve data ambiguities.

We present a gallery of results generated using our algorithm on a diverse set of 36 raw line-drawings, created by multiple artists (Section 3.6). We validate our observations and algorithm via a series of user studies, and extensive comparisons to prior art and manually consolidated drawings. These experiments jointly confirm that our method outperforms the state of the art, and provides results consistent with viewer expectations. We plan to release our data and code to facilitate further research.

3.2 Overview

3.2.1 Perception of Oversketched Strokes

To mimic the mental process viewers apply to consolidate the drawing, we rely on the following observations about human perception of sketches derived from perception literature and sketching tutorials.

Angular compatibility. Studies indicate that viewers rely on *angular compatibility*, or the degree of similarity between stoke tangents, when grouping nearby

side-by-side strokes [8, 90] (Figure 3.2a). While viewers mentally group strokes that serve as visible continuation of one another [11] they do not hallucinate curves absent from the drawing, and thus employ separate corresponding aggregate curves for such strokes (Figure 3.2c).

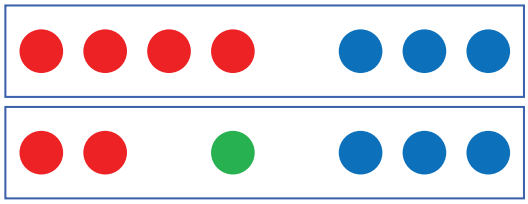


Figure 3.3: Humans group visual objects based on relative distance. The clustering below (indicated by coloring) has the inner-cluster distance that is the same as the inter-cluster distance above.

Relative proximity. Perceptual literature strongly suggests that humans group objects based on *relative proximity*, or *relative* distance: given a set of shapes, we visually group objects if the spacing between them is much smaller than the space between them and other objects (see Figure 3.3) [104]. Proximity can also be interpreted as a function of *density*: the perceived groups have near-constant internal object density, while incorporating any other object into the group would result in highly uneven density. Note that this grouping is contextual—similarly spaced objects (Figure 3.3, top versus bottom) are seen as belonging to the same, or different, groups based on the position of other objects. Also note that proximity based grouping is scale independent, scaling all distances in Figure 3.3 by the same amount will not change the grouping. Using proximity as a criterion for stroke grouping poses several challenges. First, it requires context, since one cannot assess the *relative* proximity of any individual pair of strokes. Second, relative proximity is a negative rather than positive property: it indicates when objects do **not** belong together—when both or one of them have much more close by objects—not when they do. For roughly evenly spaced strokes, relative proximity alone provides no cue as to whether these strokes should, or should not, belong together. Lastly, distances between side-by-side strokes vary at different points along them, raising a question of how to assess proximity *locally*.

Narrowness. We speculate that humans intuitively understand curves as being narrow, namely having a small width to length ratio. We believe they use this intuition to distinguish between equally spaces strokes that jointly depict aggregate curves and those that do not (Figure 3.2e). We incorporate this *narrowness* criterion into our clustering algorithm, and use a narrowness threshold estimated via a perception study that validates our hypothesis (Appendix B.1).

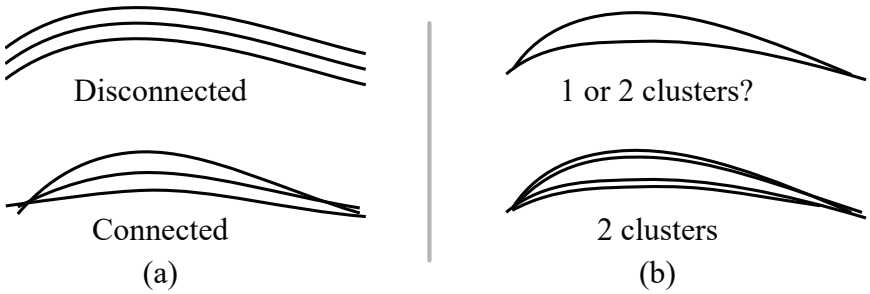


Figure 3.4: Connectedness and strength in numbers cues.

Connectedness. The *connectedness* principle highlighted by perception research [104] suggests that humans group objects that are inter-connected, such as points connected by lines. For strokes, this principle argues for grouping intersecting or near intersecting strokes when doing so does not contradict other cues (see Figure 3.4a).

Strength in numbers. Even with these cues in place, we theoretically can have stroke configurations which, from a purely perception driven perspective, are ambiguous (see Figure 3.4b). To address this type of configurations, we leverage artist intent. Specifically, we recall that our inputs are generated by artists who intend for viewers to assemble a clear mental image of the drawn content. Design literature [32] suggests that artists rely on thicker, overdrawn, lines to enhance drawing clarity and eliminate ambiguities. This suggestion confirms our observation that artists use tight multi-stroke clusters to highlight intended aggregate curves that may be ambiguous when drawn with a single stroke (Figure 3.4b). We refer to this principle as *strength in numbers* and use it to resolve ambiguous configurations,

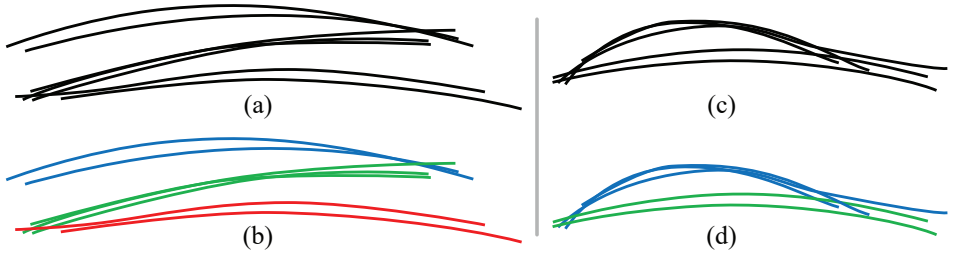


Figure 3.5: Local versus global proximity: (a,c) on-average evenly spaced (and connected) strokes may depict multiple aggregate curve branches; (b,d) perceived narrow clusters.

by using stroke number within a cluster as a factor in the final decision making (Figure 3.6d, Section 3.3.3).

3.2.2 Algorithm

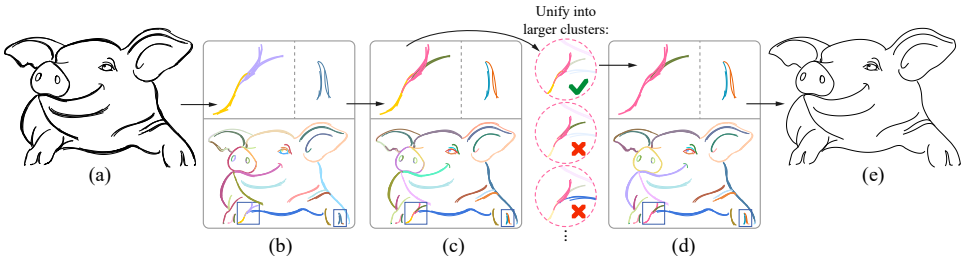


Figure 3.6: Given a raw vector sketch (a), our method first clusters based on pairwise compatibilities of angle and relative proximity, resulting in clusters consisting of connected parallel strokes (b, Section 3.3.1). Our method then analyzes relative proximity within each cluster to separate branches (c, Section 3.3.2). Given these reliable clusters, our method assesses all pairs of nearby clusters and merges them following the visual grouping rules (d, Section 3.3.3). Finally, the clusters are consolidated into the cleaned-up sketch (e, Section 3.4). Raw sketch: © Enrique Rosales.

The observations above provide cues for judging the likelihood that a given group of strokes depicts a single aggregate curve; however, these cues cannot be easily translated into any standard clustering framework. While relative proximity plays a major role in clustering decisions, assessing it requires context and thus

cannot be reliably performed on stand-alone stroke pairs. Moreover, distances between strokes may vary at different points along them, requiring fine-grained local analysis to separate connected stroke branches that depict different curves (Figure 3.5), which in turn requires a meaningful dense inter-stroke correspondence. Our method overcomes these challenges by employing a targeted clustering framework that refines clusters by gradually incorporating new and more localized perceptual cues into their assessment (Figure 3.6). We first coarsely cluster strokes based on average, or global, compatibility between their strokes (Section 3.3.1, Figure 3.6b). We first assess the angular compatibility of each pair of strokes independently. While this metric may become fuzzy for far away strokes and borderline cases, we successfully use it to provide initial, rough stroke segmentation (Section 3.3.1). We refine the obtained segmentation by assessing the relative proximity between strokes within each angle-compatible cluster, breaking clusters into sub-clusters, each of which has roughly uniform average inter-stroke spacing (Section 3.3.1). We then assess the width of the resulting clusters, as well as their local spacing uniformity. We use both cues to detect and refine clusters which are only weakly connected, namely those that have multiple distinct curve branches (Section 3.3.2, Figure 3.5). The output of this stage is a set of stroke clusters that satisfy all our perceptual criteria, and their corresponding aggregate curves (Figures 3.6c, 3.5bd).

Across all these clustering stages, we opt for a conservative interpretation of ambiguous and borderline cases, keeping strokes apart absent clear evidence of compatibility. The last merging stage of our algorithm resolves these ambiguous cases by relying on the fact that, at this point, most of the clusters already contain multiple strokes; we can therefore use intra-cluster stroke proximity to more reliably assess inter-cluster relative proximity. We merge pairs of clusters if the combined cluster satisfies our key perceptual criteria: angle, relative proximity, and narrowness. Since most of the processed clusters contain multiple strokes, we also employ the strength in numbers principle by including cluster size in our consideration of borderline cases. This process is repeated until no more cluster pairs can be merged (Section 3.3.3, Figure 3.6d).

To assess the different properties of the considered clusters throughout the algorithm, we compute their corresponding aggregate curves (Section 3.4) and use

those as a common reference frame, or parameter domain, for perceptual properties assessment. We use the same computation to generate the final consolidated drawing (Figure 3.6e).

Input and Output. The input to our algorithm is a line drawing in vector format, generated using a standard stylus and tablet interface, and where each stroke is represented by a polyline. We expect each stroke to have an associated width value, generated via tablet pressure or other UI specification, and we assume a stroke width of one if such a value is not available. We replace clusters of strokes that jointly depict individual artist intended curves by their corresponding aggregate curves. We represent the aggregate curves using the same format as the input strokes: as polylines with an associated width. We leave it to the user to decide if they want to fit these polylines with smooth curves later on (e.g. using the methods in [7, 72]).

Raw strokes captured via a stylus-on-tablet interface are often noisy due to a combination of involuntary hand movement and capture software inaccuracy [7, 72]. We pre-process the raw data as described in Appendix A.1. We do not use this process on previously cleaned data, such as the examples provided by [69]. Our figures include both raw and pre-processed strokes: input renders show the raw strokes and clustering output images show pre-processed ones for comparison.

3.3 Stroke Clustering

3.3.1 Coarse Clustering

Clustering Based on Angular Compatibility

The angular compatibility between a stroke pair provides the first cue about whether these strokes depict a common aggregate curve. Two nearby strokes S_i and S_j are more likely to depict the same aggregate curve when they are fully or partially parallel and are less likely to belong together when they are orthogonal to one another. We define an angular compatibility score $ComA(S_i, S_j)$ that addresses all these scenarios (Equation 3.1). This score is set to be positive for strokes that are

angle compatible, and negative for those which are not. The value of the score reflects the degree of (in)compatibility. Since angle provides a confident estimator of compatibility only for nearby side-by-side strokes, we set the score to a small negative value for all other stroke pairs, allowing their clustering to emerge from the interaction of more adjacent strokes.

Given the angular compatibility scores, we wish to group stroke pairs with positive scores, to separate strokes with negative scores, and to resolve ambiguities by considering the magnitude of the scores. This set of requirements naturally fits into a correlation clustering framework [5]. The advantage of using correlation clustering over other clustering formulations is that the number of clusters emerges directly from the input scores and does not need to be estimated as *a priori*. We formulate our clustering goal as maximizing $\sum_{ij} ComA(S_i, S_j) Y_{ij}$, where $Y_{ij} = 1$ if the two strokes are in the same cluster and $Y_{ij} = 0$ otherwise. Obtaining an optimal correlation clustering is proven to be NP-complete; we use the method of Keuper et al. [59], which provides an efficient approximation of the optimum.

Pairwise Angular Compatibility Score

We require an angular compatibility score that is robust to noise and accounts for the different adjacency relationships between stroke pairs: strokes that are fully and partially side-by-side. In previous work, this problem is handled by crafting multiple special cases [8, 69] or by considering only angles at closest points [78]. Purely local angle computation is clearly unreliable, as point-wise normals can be noisy, but an average or integral measure requires a meaningful reference frame or correspondence between the two strokes. We provide a unified, integral angular compatibility score by first fitting a common aggregate curve $\mathcal{S}_{ij}^A$ to the pair of strokes $\mathcal{S}_i$ and $\mathcal{S}_j$ (Section 3.4), and then assessing the angles between the tangents of this common curve and each of its originating strokes. Specifically, we define $D_a(\mathcal{S}_i, \mathcal{S}_j)$ (Equation 3.2) as the *angular distance* between each stroke and the aggregate curve and set $D_a(\mathcal{S}_i, \mathcal{S}_j) = \max(D_a(\mathcal{S}_i, \mathcal{S}_{ij}^A), D_a(\mathcal{S}_j, \mathcal{S}_{ij}^A))$. Since each point on the input strokes has a corresponding point on the fitted curve, this formulation addresses all possible stroke configurations, providing a unified measure. We convert this angular distance value $\phi = D_a(\mathcal{S}_i, \mathcal{S}_j)$ into a compatibility score as

follows:

$$ComA(\mathcal{S}_i,\mathcal{S}_j) = \begin{cases} 1, & \phi < 8^\circ \\ \exp(-\frac{(\phi-8^\circ)^2}{2\sigma_1^2}), & 8^\circ \le \phi < 17^\circ \\ 0, & 17^\circ \le \phi < 23^\circ \\ -1.5\exp(-\frac{(\phi-30^\circ)^2}{2\sigma_2^2}), & 23^\circ \le \phi < 30^\circ \\ -1.5, & 30^\circ \le \phi \end{cases}, \tag{3.1}$$

The parameters of this function reflect cues from perception research. Literature indicates that viewers use approximately 20° as the threshold distinguishing between perceived similar and dissimilar tangents [49]. We therefore center our compatibility function around this value, and use an angular compatibility threshold $T_a = 20$ through the rest of our computations. We set the size of the Gaussians to create smooth dropoff: $\sigma_1 = 9^\circ/3.5, \sigma_2 = 7^\circ/3.5$. At this stage we are seeking for conservative clusters, and therefore we use a higher negative than positive maximal correlation score (1 v.s. -1.5).

Angular compatibility only impacts clustering decisions for nearby curves. We expect far away curves to end up in the same final cluster only if they are interconnected via series of intermediate proximate and angle compatible strokes (Figure 3.2ac). We therefore set the overall score to a minuscule negative number -10^{-6} for strokes that are far from one another (farther than twenty times the stroke widths, $20W_s$ away at their nearest points) or have no side-by-side sections (Figure 3.7). This value is small enough to allow strokes to be grouped together if they share angle compatible intermediate strokes, but pushes them apart otherwise.

Angular Distance

We compute the angular distance between a stroke $\mathcal{S}_i$ and a corresponding aggregate curve $\mathcal{S}_{ij}^A$ as follows. For a point $\mathbf{p} \in \mathcal{S}_i$, we define the corresponding point $\mathbf{p}' \in \mathcal{S}_{ij}^A$ as its closest point on the aggregate curve. Given this correspondence mapping $\mathbf{p}' = M_i(\mathbf{p})$, we compute the pointwise angular difference at $\mathbf{p}'$ as $A_i(\mathbf{p}') = \arccos(\mathbf{t} \cdot \mathbf{t}')$. Here, $\mathbf{t}$ and $\mathbf{t}'$ are unit tangents to $\mathcal{S}_i$ and $\mathcal{S}_{ij}^A$ at $\mathbf{p}$ and $\mathbf{p}'$ respectively.

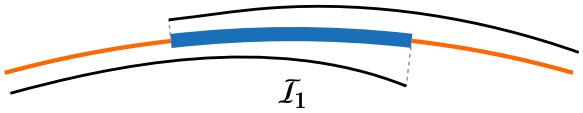


Figure 3.7: Stroke pair layouts.

We intend to use the stroke to curve angular distance to evaluate whether two strokes are roughly parallel. Therefore, instead of integrating angular distances along the entire curve, we narrow the computation to *sections of interest* where points on the aggregate curve have corresponding points on both input strokes $\mathcal{I}_1$ (Figure 3.7, blue). We evenly sample the points $\mathbf{p}$ along $\mathcal{S}^A_{i,j}$ and define the angular distance as

$$D_a(\mathcal{S}_i, \mathcal{S}^A_{i,j}) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} A_i(\mathbf{p}'), \tag{3.2}$$

where $|\mathcal{I}_1|$ is the number of samples along the section $\mathcal{I}_1$.

Average Proximity Based Clustering

Our first step of the coarse clustering stage focuses on angular compatibility, and thus often groups side-by-side strokes which are visibly disjoint (Figure 3.8a). We separate such strokes by breaking angle compatible clusters into sub-clusters with no sudden internal proximity changes based on *average* inter-stroke proximity. This process results in clusters which are narrow enough to be effectively parameterized via a shared aggregate curve based correspondence (Figure 3.8b). We use the computed correspondences to further refine these clusters using *local* proximity analysis (Section 3.3.2, Figure 3.8c).

To measure the proximity, or distance, between two strokes $\mathcal{S}_i$ and $\mathcal{S}_j$ we fit them using an aggregate curve $\mathcal{S}^A_{ij}$ which provides us with their common parameterization. We define the correspondence mapping $\mathbf{q} = M_{ij}(\mathbf{p})$ where $M_i(\mathbf{p}) = \mathbf{p}' = M_j(\mathbf{q})$ are the mappings from the strokes to the curve $\mathcal{S}^A_{ij}$. Note that by construction the points $\mathbf{p}', \mathbf{q}, \mathbf{p}$ are colinear and the line connecting them is orthogonal to

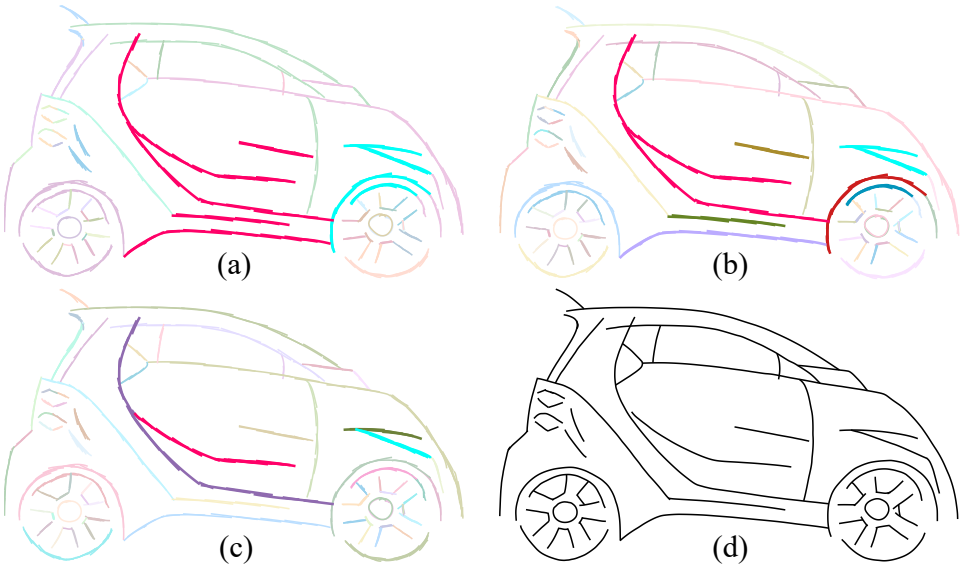


Figure 3.8: Clustering stages: (a) angle based clustering output with two clusters (pink and cyan) highlighted; (b) average proximity based clustering breaks these two clusters into roughly evenly spaced distinct components; (c) local refinement separates branches producing uniformly narrow clusters; (d) consolidated output. Input sketch is from [78].

the aggregate curve. The average distance is then defined as

$$D_{i,j}(\mathcal{I}_1) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} ||\mathbf{p} - \mathbf{q}||. \tag{3.3}$$

If the side-by-side section $|\mathcal{I}_1|$ is empty we set the inter-stroke distance $D_{i,j} = +\infty$. Our computation directly employs the mapping between the stroke points, since at this point in the computation, the side-by-side portions of the strokes we consider are roughly parallel, ensuring reliable correspondences. This was not the case for the angle difference computation (Equation 3.2), where to obtain reliable values we had to map strokes to the aggregate curve instead of to one another.

To measure proximity within a cluster $\mathcal{C}$, for each stroke, we locate its nearest neighbor based on the inter-stroke distance. We define the internal cluster proxim-

ity as the maximum of these distances:

$$D_c = \max_{i \in c}(\min_{j \in c,\ j \neq i} (D_{i,j})).$$

Intuitively, this value measures the size of the largest gap between strokes in the cluster. We measure the distance between two distinct clusters by finding the closest distance between any two strokes where each stroke belongs to a different cluster:

$$D_{c,c'} = \min_{i \in c, j \in c'} D_{i,j}$$

Following the relative proximity principle, we merge clusters $\mathcal{C}$ and $\mathcal{C}'$ if both of the following conditions are true.

$$D_{c,c'} < T'_d \cdot \max(D_c, D_{c'}),$$
$$\max(D_c, D_{c'}) < T'_d \cdot \min(D_c, D_{c'}).$$

We set T'_d as follows. Our proximity study (Appendix B.1) indicates that humans separate lines when the ratio of intra-cluster to inter cluster distances reaches approximately $T_d = 2.1$. The distances we employ at this stage are averaged along the full length of the strokes, and are thus only approximating closest inter- and intra- cluster distances. We perform more fine grained-analysis during subsequent local separation; therefore, to avoid over-segmentation at this stage, we use $T'_d = 1.25 \cdot T_d$. Increasing the multiplicative factor from 1.25 to 1.3, or even 1.4, leads to no visible changes in our outputs.

We merge clusters incrementally, using the merging criterion above. We speed up computation by using the HDBSCAN algorithm [16].

Initialization. Our clustering criterion uses intra-cluster distances D_c. However, these are only meaningful for clusters with at least two strokes. We generate initial clusters by leveraging the connectedness principle. We recall that intersecting or near-intersecting strokes are likely to be seen as grouped together. We therefore generate initial clusters by forming (near-)connected stroke components. We consider two partially side-by-side strokes as nearly-intersecting if they have pairs of

points at a distance less or equal to twice the stroke widths, $2W_s$. We group intersecting pair of strokes only if the resulting clusters conform to our perceptual cues: we check that the two strokes are angle compatible and that their joint aggregate curve is narrow. We measure angular compatibility as

$$A_{i,j}(\mathcal{I}_1) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} \arccos(\mathbf{t}(\mathbf{p}) \cdot \mathbf{t}(\mathbf{q})), \tag{3.4}$$

where $\mathbf{p}' = M_i(\mathbf{p}) = M_j(\mathbf{q})$, and $t(p), t(q)$ are their respective tangents. If the angle average $A_{i,j}$ exceeds the threshold T_a, we keep the strokes separate. To assess narrowness, we compute the width W_c of their joint aggregate curve (Equation 3.5) and compare the curve's length to width ratio against the threshold $T_n = 8.5$ established via our study (Appendix B.1).

Aggregate Stroke Width To compute the width of an aggregate curve, we first shoot left and right orthogonal rays from densely sampled point $\mathbf{p} \in \mathcal{I}_1$ on the curve and locate the farthest left and right intersections $i_l(\mathbf{p})$ and $i_r(\mathbf{p})$ with cluster strokes along each ray. We set the width as

$$W_c = \max(W_s, \underset{\mathbf{p} \in \mathcal{I}_1}{\text{median}}(||i_l(\mathbf{p}) - i_r(\mathbf{p})||)). \tag{3.5}$$

3.3.2 Local Cluster Refinement

The clusters obtained via this bottom-up clustering are visually connected but may depict multiple connected curve branches instead of a single aggregate curve (Figures 3.5, 3.8, 3.9). We detect such multi-branch clusters and separate them into branches that correspond to individual aggregate curves using a top-down recursive process (Algorithm 1). At each level of the algorithm we consider two criteria: evenness of the internal spacing between cluster strokes (Section 3.3.2), and cluster narrowness. We assess narrowness as described in Section 3.3.1. For any cluster that fails one of these tests, we perform the split that maximally reduces spacing unevenness (Section 3.3.2). Once a cluster is split into left and right branches, we recursively apply the refinement algorithms to these branches.

In assessing spacing evenness, we seek to detect contiguous branches, or sub-clusters, that have significantly larger intra-cluster spacing along a significant portion of their length, compared to the internal spacing within each branch (Section 3.3.2). We generate candidate sub-clusters based on local inter-stroke spacing (Section 3.3.2) and then compare their internal spacing to the inter-cluster one to determine if they indeed need to be separated (Section 3.3.2).

ALGORITHM 1: Recursive Branch Separation

Input: A set of strokes to separate, $\mathcal{C}$.
Output: *ResultBranches*.
ResultBranches $\leftarrow \{\mathcal{C}\}$;
PotentialSeparations $\leftarrow$ FindPotentialSeparations($\mathcal{C}$) (Sec. 3.3.2);
$R_{max} \leftarrow 0$; $\{\mathcal{C}_L^*, \mathcal{C}_R^*\} \leftarrow \{\mathcal{C}, \emptyset\}$;
for *each separation* $\{\mathcal{C}_L, \mathcal{C}_R\}$ *in PotentialSeparations* **do**
    $R \leftarrow$ ComputeGapRatio($\{\mathcal{C}_L, \mathcal{C}_R\}$) (Sec. 3.3.2);
    if $R > R_{max}$ **then**
        $R_{max} \leftarrow R$; $\{\mathcal{C}_L^*, \mathcal{C}_R^*\} \leftarrow \{\mathcal{C}_L, \mathcal{C}_R\}$;
    end
end
if $R_{max} > T_d$ *or* $\mathcal{C}$ *violates* Narrowness **then**
    LeftBranches $\leftarrow$ RecursiveBranchSeparation($\mathcal{C}_L^*$);
    RightBranches $\leftarrow$ RecursiveBranchSeparation($\mathcal{C}_L^*$);
    ResultBranches $\leftarrow$ *LeftBranches* $\cup$ *RightBranches*;
end

Potential Clusters

We compute potential sub-clusters by analyzing local spacing between strokes (Figure 3.9). We parameterize each cluster by shooting orthogonal rays from densely sampled aggregate curve points and compute the intersections of these rays with the cluster strokes (Figure 3.9, inset). We order the intersections from leftmost to rightmost (with left and right defined with respect to the aggregate curve direction). The lengths of the segments, or gaps, between consecutive intersections along individual rays, provide a local measurement of relative proximity. If all these gaps are of equal size, then visibly the intersection points and their corresponding strokes are grouped together. If a gap g is much larger than the gaps to the left $g_L \in G_L$ and right $g_R \in G_R$ of it, then the intersections to the left and to

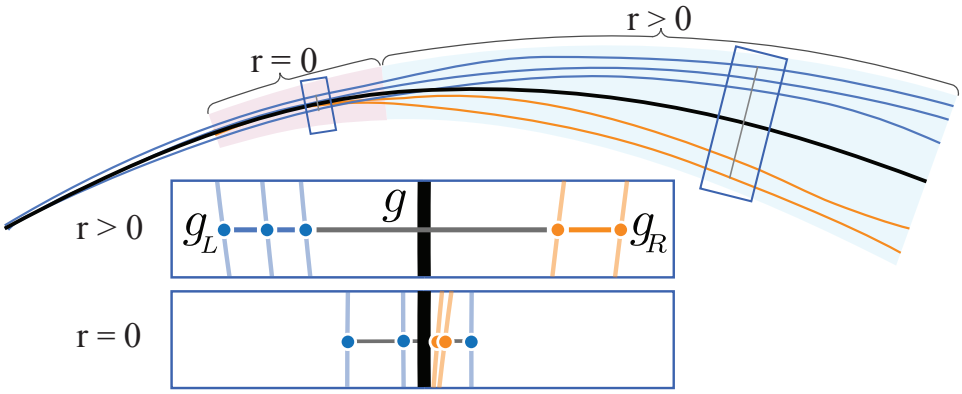


Figure 3.9: Local cluster refinement: Pointwise stroke correspondences are defined using intersections between strokes and orthogonal rays emanating from the cluster's aggregate stroke (black). The spacing between lowest top (blue) and highest bottom (orange) intersection points is significantly larger than the internal spacing within the top (blue) and bottom (orange) branches. We measure this uneven distribution of intersection points by comparing the inter-cluster gap g (gray, upper inset) and the left, right gaps g_L, g_R (blue, orange, upper inset). The measured gap ratio r is positive when the two clusters are clearly separate (upper inset, blue shadow section) and zero when they overlap (lower inset, red shadow section).

the right and the strokes they lie on are locally visibly separate (Figure 3.9). We first detect candidate gaps g which indicate possible cluster separation using the ratio between the length of this gap and that of those left and right to it as a cue Specifically, we mark a gap g as a candidate if

$$g > T_d(g_L + g_R)/2.$$

If g is the leftmost or rightmost gap, we only compare its size against that of the gaps to the right, or left, respectively. If there is only one gap, i.e. only two participating strokes, we set $g_L = g_R = 2W_s$, the same lower bound on gap size as in the initialization of Section 3.3.1.

Given a candidate gap, we assign the strokes to the left and right of it into separate left and right clusters, $\mathcal{C}_L$ and $\mathcal{C}_R$, respectively. We then assign the remaining strokes to these clusters as follows. We first advance left and right along the aggregate curve as long as all currently marked strokes remain on correct sides. At each

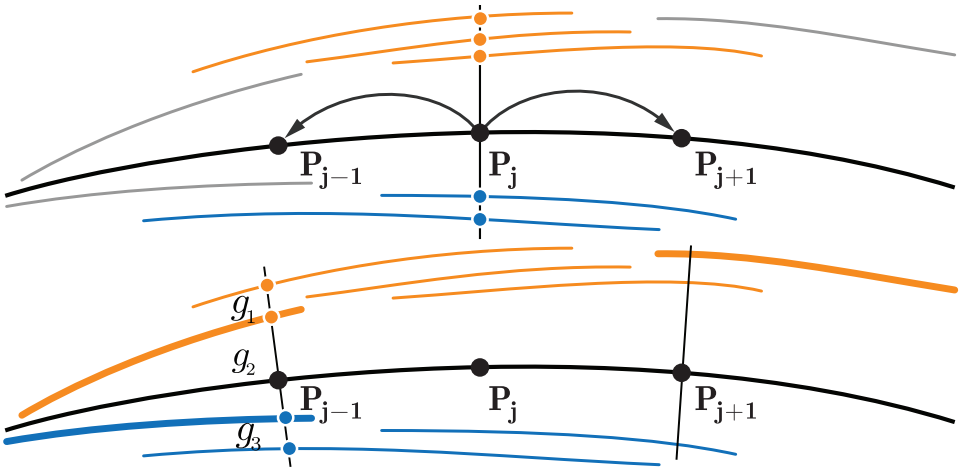


Figure 3.10: The growing step for potential separation generation. At position $\mathbf{p_j}$, given the gap across the aggregate curve, the intersection points are labeled into blue and orange, and the strokes are labeled correspondingly. The assignment at $\mathbf{p_j}$ is propagated into $\mathbf{p_{j-1}}$ and $\mathbf{p_{j+1}}$. There are three possible separations at $\mathbf{p_{j-1}}$ defined respectively by g_1 to g_3. Our method chooses the largest gap g_2 greedily. There is only one possible assignment at $\mathbf{p_{j+1}}$.

encountered aggregate curve point, we split the unmarked strokes locally based on the largest gap between the previously marked strokes. Intuitively, the optimal assignment of the remaining strokes is one that maximizes the average gap between the left and right clusters. To make this assignment, we assess three alternatives and choose the separation that produces the largest average gap ratio. The three alternatives we test are assigning each stroke to the nearest, left or right, cluster based on shortest distance, assigning all remaining strokes to $\mathcal{C}_L$, or assigning all remaining strokes to $\mathcal{C}_R$.

Separation assessment

Given a pair of clusters, we analyze the gap ratio to determine whether they should be separated. We iterate over all rays that intersect both clusters and, for each ray, locate the leftmost intersection with the right cluster and the rightmost intersection with the left cluster. If these intersections are immediately next to one another, we compute the ratio between the size of middle gap g and the size of the average left

and right gaps as above

$$r = g/((g_L + g_R)/2).$$

If the intersection order is flipped, the clusters are locally connected. In this case, we set $r = 0$.

The left and right gap values are ill-defined if the one of these clusters consists of a single stroke. They can also be arbitrarily small at a location where two or more strokes intersect; a division by a value close to zero would result in an arbitrarily large ratio value which would drastically change the average ratio. We resolve both cases by rounding $(g_L + g_R)/2$ up to a lower bound. To determine the bound we compute the average inter-stroke distances d_l and d_r within the left and right clusters. If the larger of these is above the baseline value of $2W_s$ that we use throughout (Section 3.3.1, 3.3.2), we use $2W_s$ as the lower bound.

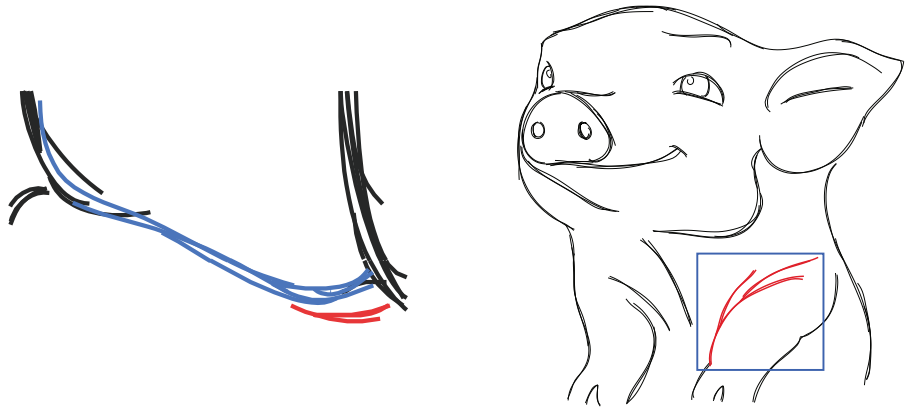


Figure 3.11: Special cases of separation assessment. Wide cluster is separatable even when inter-stroke distance is small (a); An intermediate cluster may contain more than one branch (b). Input sketches: © Enrique Rosales.

Otherwise, we examine if the cluster is sufficiently wide to potentially merit separation. We classify a cluster as wide if the ratio of its length l to its maximal gap g_m inside is close to the narrowness threshold $l/g_m < 2T_n$. We use the default bound for non-wide clusters. For clusters which are wide, yet have very small left and right inter-stroke distances (see Figure 3.11a), we follow the strength in numbers principle and use the smaller bound $\max(d_l, d_r, W_s)$. This choice facil-

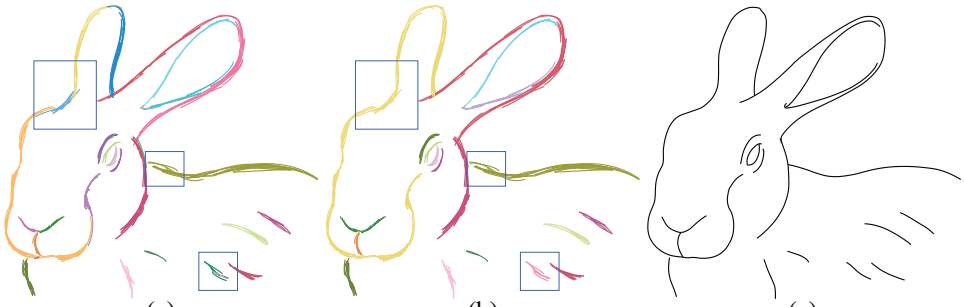


Figure 3.12: Final unification: (a) before; (b) after, (c) consolidated result. Input sketch: © Enrique Rosales.

itates separation of small clusters with small inter-cluster gaps but even smaller intra-cluster gaps.

We use these computed ratios to determine if the left and right clusters are separable. In theory, if each of the left and right clusters had uniform internal spacing, we could directly compare the average of local ratios r to our proximity threshold T_d to determine if the two clusters need to be separated. However, our original cluster could have multiple branches (Figure 3.5ab). Thus, either of the left or right clusters may consist of more than one branch (see Figure 3.11b) and, as a result, may have large internal gaps; this makes gap ratio assessment less reliable. To nevertheless separate such right and left clusters, we use a more lax gap ratio assessment, setting R to the average of the 90% largest ratio values and compare this number to the threshold as a separation criterion. While this approach may occasionally lead to over-segmentation, the resulting split clusters are merged back by our final unification step. If multiple cluster pairs pass the splitting test, we select the one with the largest R.

3.3.3 Cluster Unification

We finalize the consolidation process by assessing each pair of clusters and merging them if the joint cluster satisfies our compatibility criteria (angle, proximity, and narrowness). We can now perform this task reliably as most clusters now contain multiple strokes, allowing for reliable relative proximity assessment and aggregate curve width estimation. Conceptually this step mirrors our branch separation

step (Section 3.3.2) by using similar criteria and principles. As an optional step, we further consolidate the output drawing by detecting and enforcing T-junctions and shared end points between aggregate curves.

Pairwise Assessment We determine if a pair of clusters $\mathcal{C}_l$ and $\mathcal{C}_r$ should be merged based on narrowness, local angular compatibility, and relative proximity. We assess narrowness as before: we compute a common aggregate curve $\mathcal{S}^A_{lr}$ that corresponds to the union of the two clusters. If the length to width ratio of the curve $\mathcal{S}^A_{lr}$ is smaller than T_n, we keep the two clusters separate.

We assess angular compatibility within the region where the two clusters are side-by-side. Given the aggregate curve $\mathcal{S}^A_{lr}$ and the left and right aggregate curves, $\mathcal{S}_l$ and $\mathcal{S}_r$, we compute the average angle difference as described in Equation 3.4 by averaging pointwise angle differences between $\mathcal{S}_l$ and $\mathcal{S}_r$ with respect to $\mathcal{S}^A_{lr}$. If the angle average exceeds the threshold T_a, we keep the clusters separate.

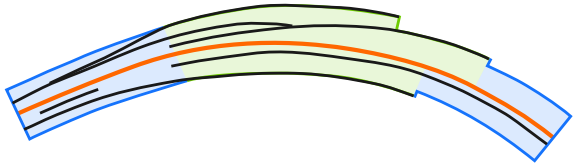


Figure 3.13: The outermost strokes in a cluster form a cluster envelope that is used to assess proximity between clusters.

Proximity Assessment. In assessing proximity between clusters, we try to overcome local noise by computing distances between clusters that account for their average rather than pointwise width. We wish to use this average width when computing distances between clusters in regions where the pointwise width is smaller than the average. To this end, we introduce the notion of a cluster envelope (see Figure 3.13) computed based on the cluster's width. This envelope is designed to reflect the average width of the cluster and contain all cluster strokes. We fit every cluster with an aggregate curve $\mathcal{S}^A$ and compute the widths of these curves W_c (Equation 3.5). We define the cluster envelope using the cluster's width as follows. We shoot orthogonal rays left and right from dense ordered samples on the clus-

ter's aggregate curve. If the distance from the curve to the outermost intersection with a cluster stroke is larger than half the curve's width, we use this intersection (Figure 3.13, green) as an envelope vertex, otherwise, we use a point along the ray at a half width distance as a vertex (Figure 3.13, blue). We connect vertices corresponding to adjacent samples on the left and right of the curve, forming two envelope boundaries. We connect the last left and right vertices on both ends of the cluster to form a closed envelope polygon.

For each cluster we compute the median gap g within it. To compute it, we consider all gaps between adjacent intersections along orthogonal rays emanating from aggregate curve samples. For median computation we ignore rays that intersect only a single stroke, as well as intersections which are less than a stroke width apart.

We merge clusters if the distance between their envelopes is less than $T_d \cdot (g_l + g_r)/2$ everywhere along their side-by-side sections. We measure the local distances along the rays computed for each cluster and compare those to our threshold. To account for noise in the computation, we ignore sequences of gaps larger than this threshold if the length of this sequence (measured as distance between the originating samples of the rays) is less than $\min(5W_s, 0.1 \cdot L)$ where L is the length of the aggregate curve $\mathcal{S}^A_{lr}$.

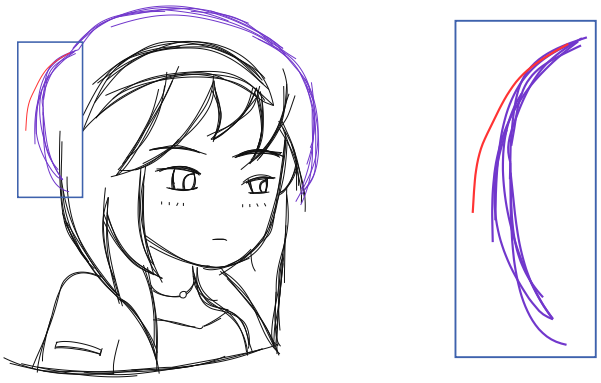


Figure 3.14: An example of an outlier stroke visually separate from the other strokes in their intended cluster, sourced from Liu et al. [69].

Single-stroke clusters. As noted earlier, relative proximity assessment requires at least three strokes to be meaningful, making assessment of proximity for single-stroke clusters problematic. Moreover, when drawing free-hand, artists do occasionally draw outlier strokes—ones that are intended to depict a target aggregate curve but are sufficiently inaccurate to be visually separate from the other strokes in their intended cluster (see Figure 3.14).

We handle such ambiguous configuration by leveraging the strength in numbers principle. For pairs of clusters where one cluster has multiple strokes and the other has only one stroke, we use the angle and narrowness tests as above, but apply a relaxed version of the proximity test as follows. We keep the clusters apart if the shortest distance between the single stroke and the envelope of the multi-stroke cluster is larger than the median gap g computed on the multi-stroke cluster m. Otherwise, as before, we measure the gaps between the cluster's envelope and the stroke and compare those to $T_d \cdot g$. We relax the strict proximity requirement above and merge the stroke into the cluster if half the gaps within the side-by-side region are below the threshold.

We finally consider pairs of single-stroke clusters. We use exactly the same process as for the single stroke test above, but use the stroke width W_s in lieu of the gap size g.

Outliers. Finally, we addresses a common artifact present in raw artist drawings. When artists draw clearly erroneous strokes, instead of deleting them, they sometimes simply hide them underneath wide clusters of overdrawn strokes. To detect such outliers, for each pair of single-stroke and multi-stroke clusters we assess containment as follows. We intersect the single stroke $\mathcal{S}$ with the cluster's envelope and measure the portion of $\mathcal{S}$ which is outside the envelope. We classify the stroke $\mathcal{S}$ as an outlier and remove it from the consolidated output, if this portion is less than 10% of its length.

Enforcing curve connections. We locate and enforce coincident aggregate curve end-points and T-junctions as an optional post-processing step. We consider two endpoints of aggregate curves $\mathcal{S}_i$ and $\mathcal{S}_j$ with width W_i and W_j respectively as co-

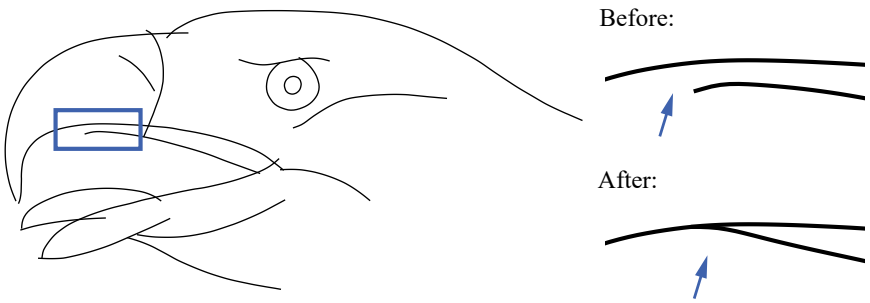


Figure 3.15: T-junctions are optionally enforced as a post-processing step.

incident, if they are within a distance of $W_i + W_j$ from one another. We consider an end-point of a curve $\mathcal{S}_i$ as forming a T-junction with the curve $\mathcal{S}_j$ if it is similarly within distance $W_i + W_j$ from its closest point on $\mathcal{S}_j$. To enforce these detected connections, we project the stem end-points at T-junctions to the top curves of the T, and place the shared end-points at their average locations. We propagate the connection constraints along the curves by using standard Laplacian deformation [99]; we use the current positions and tangents of curve vertices as reference and trigger the deformation by constraining the curve end-points to their new locations. We show a comparison in Figure 3.15. All final results shown in this paper have this optional step turned on.

3.4 Fitting

The goal of this stage is to fit an aggregate polyline curve to a cluster of polyline strokes. In computing the curve we seek to capture its artist intended shape, and to explicitly preserve the slopes, or tangents, of the input strokes (Figure 3.16). Our main challenge is that while our input points are ordered along each given stroke, we have no order between points on different strokes. Standard fitting frameworks are not well designed for such data: traditional polyline or parametric curve fitting techniques for unordered data typically do not account for tangents, while implicit frameworks that use normals or tangents are typically designed for closed curves.

We compute the desired curve using a modified Moving-Least-Squares (MLS) fitting algorithm [63, 64]. The standard MLS formulation does not support tangent optimization, since tangent processing requires point order information which is

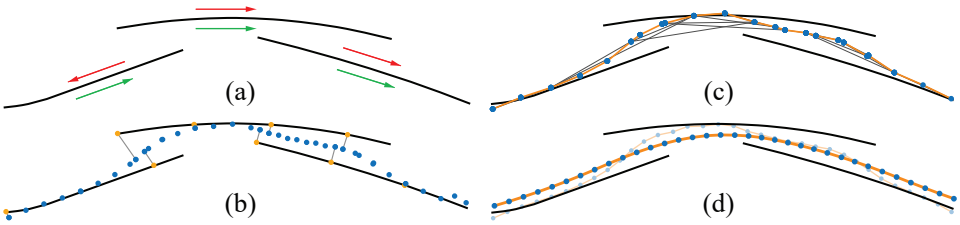


Figure 3.16: Aggregate curve fitting: (a) input stroke with original (red) and consistent (green) orientations; (b) MLS fitting output; (c) proximity graph and extracted polyline (e) resampled (thin) and final (thick) optimized polyline curve.

not available in the MLS setting. To provide an ordering, we split the fitting into three stages: we first perform an initial MLS optimization, where we solve for positions and tangents separately; we then use these positions and tangents to compute an initial aggregate polyline; finally, we align the edges of this polyline with the desired tangent directions. As an alternative to our first step, one could obtain tangential information by constructing a non-oriented gradient field [19]; however, this still requires consistently orienting the resulting tangents.

Stroke Orientation To perform any meaningful operations on point tangents, we require their orientations to be consistent (Figure 3.16a). More specifically, we want point tangents along parallel or near-parallel strokes to have similar directions.

We achieve this goal using a simple pair-based orientation method. We pick the longest stroke in the group, and set its orientation as *defined*; we set the orientations of all other strokes to be *undefined*. We then repeatedly select the closest pair of one *defined* and one *undefined* strokes based on a distance computed as described below. We assign an orientation to the *undefined* stroke such that $\mathbf{t}(\mathbf{p}) \cdot \mathbf{t}(\mathbf{p}') > 0$ using their respective representative points $(\mathbf{p}, \mathbf{p}')$. We assign a distance value to each pair of strokes as follows. If the mid-point tangents of the two strokes are near perpendicular (larger than 60° in our implementation), their orientation with respect to one another is not well defined. We therefore set the distance between them to ∞. This choice delegates the orientation decision to other more reliable pairs if these exist. Otherwise, we locate close and representative pairs of points on

the two strokes. To avoid points with unreliable normals, we only consider points on each stroke whose tangents are within 60° to the mid-point tangent. We then select the closest pair of such sample points $(\mathbf{p}, \mathbf{p}')$ and use the distance between them as the pairwise stroke distance. This process works well for the data we tested and requires less computation than more complex alternatives such as eigenspace analysis [78].

MLS fitting Our initial fitting step uses Moving-Least-Squares (MLS) with adaptive neighborhood size [63, 64]. We adapt the basic MLS framework to simultaneously compute both position and tangent values. MLS takes a point cloud as the input and projects these points to the position-error-minimized manifold (the position stroke $\mathcal{S}^P$ in our case) [64]. To conduct the MLS projection step, each point needs to be associated with a local neighborhood. Following the method in Lee [63], we construct the neighborhood by adaptively increasing the radius of a disk centered at each point. The radius is increased until all points in this disk are adequately co-linear; that is, until the correlation reaches a minimum value ρ. We use an initial neighborhood size of $h_0 = 10W_s$ and set the minimum correlation to $\rho = 0.7$. We obtain the point positions on $\mathcal{S}^P$ using the standard MLS projection (Figure 3.16b).

We compute the corresponding tangents as follows. Let $\mathbf{p}$ be the position of a input sample and $\mathbf{t}$ be its corresponding tangent. With the final neighborhood size h, we now define the averaging kernel for a position $\mathbf{p}^0$ with tangent $\mathbf{t}^0$ as

$$K(\mathbf{p}^0, T) = \frac{\sum_{p \in N(\mathbf{p^0})} \mathbf{t} \cdot \theta(||\mathbf{p} - \mathbf{p}^0||)}{\| \sum_{p \in N(\mathbf{p^0})} \mathbf{t} \cdot \theta(||\mathbf{p} - \mathbf{p}^0||) \|}. \tag{3.6}$$

We define the neighborhood $N(\mathbf{p^0})$ to include all the points p that satisfy $||\mathbf{p} - \mathbf{p}^0|| < \alpha h$ and $\mathbf{t} \cdot \mathbf{t}^0 > \beta$. Here, we scale the neighborhood size h by $\alpha = 0.6$ to avoid tangent over-smoothing, since tangents are more sensitive than positions. We set $\beta = \cos(\frac{\pi}{3})$ to avoid averaging outlier tangents. $\theta(d) = \exp(-d^2/(\alpha h)^2)$ is a Gaussian function, similar to the position Gaussian of the MLS projection.

Polyline extraction. After computing the positions and tangents on for points on $\mathcal{S}^P$, we extract an ordered sequence of such points that will form the base for our output polyline (Figure 3.16c). We compute this sequence as a path in a directed graph as follows. We construct an Euclidean proximity graph where each point is connected to all neighbors within the distance of h. Each edge in this graph is assigned a direction that aligns with the averaged tangent of its two endpoints. When the dot product of the two tangents is negative, it suggests that one of them is an outlier and the edge is thus deleted. We then compute the minimum spanning directed tree using Edmonds algorithm [22, 30] and trim the tree down to its largest path. We determine if the path is closed by searching for a path from its end to its beginning. If such a path exists, and its length is below a small value ($5W_s$ in our implementation), we label $\mathcal{S}^P$ as closed. An artist may not precisely line up the start and end of a closed loop, and may accidentally extend the end of a closed loop past its starting point. In order to address this case in addition to the start to end path, we test paths between all vertices within 10% away from the start and end points.

Tangent optimization. We now seek to optimize the polyline $\mathcal{S} = \{\mathbf{p}_i\}(i = 1, \ldots, n)$ by aligning its edges $(\mathbf{p}_i, \mathbf{p}_i + 1)$ with the corresponding neighborhood tangents. Our objective function is defined as

$$d(\mathcal{S}, \mathcal{S}^A) = \sum_{i=1}^{n} \| \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{||\mathbf{p}_{i+1} - \mathbf{p}_i||} - K(\mathbf{p}_i, T)||^2 + \lambda ||\mathbf{p}_i - \mathbf{p}_i^0||^2, \tag{3.7}$$

$\mathbf{p}_i^0$ is the initial position of point $\mathbf{p}_i$ on the aggregate polyline curve. Here, the first term enforces tangent alignment and the second term reflects the expectation that the polyline stays close to its original position. We set $\lambda = 10^{-3}$ to prioritize tangent alignment.

We minimize Equation 3.7 using iterated least squares. We define the kth round

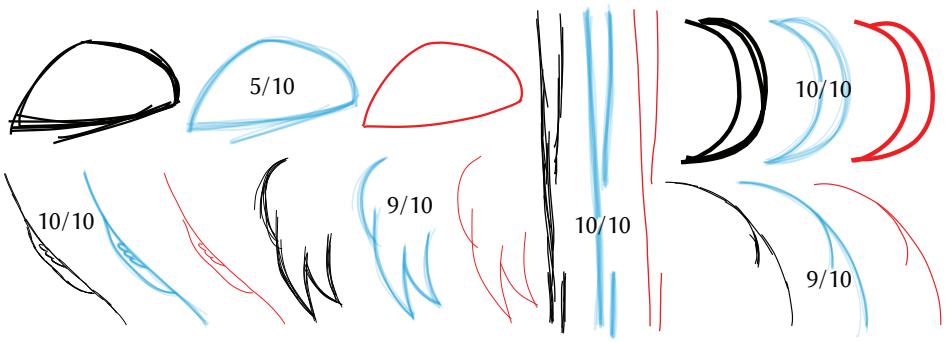


Figure 3.17: Examples of manually (blue) and algorithmically (red) traced aggregate curves of different stroke configurations (black). Manual results form multiple participants are overlaid over one another. The ratio shows the number of participants whose results agreed with the plurality consolidated result in terms of output curve number and approximate location. In all cases our result aligns with the plurality response.

objective as

$$d(\mathcal{S}^k, \mathcal{S}^{k-1}) \approx \sum_{i=1}^{n} || \frac{\mathbf{p}_{i+1}^k - \mathbf{p}_i^k}{||\mathbf{p}_{i+1}^{k-1} - \mathbf{p}_i^{k-1}||} - K(\mathbf{p}_i^{k-1}, T)||^2 + \lambda ||\mathbf{p}_i^k - \mathbf{p}_i^0||^2 \tag{3.8}$$

Here, we replace the varying polyline edge length term in the denominator with the known corresponding length in $\mathcal{S}^{k-1}$; $K(\mathbf{p}_i^{k-1}, T)$ is the average kernel centered at position $\mathbf{p}_i^{k-1}$ and T is the input tangent set. The aggregate tangent update helps center the curve and diminish the impact of outlier stoke tangents.

We solve this least-squares problem using standard Cholesky decomposition. For smooth input data a single tangent update step is typically sufficient. However, solving the problem for multiple rounds gives better results for highly noisy cases. We find three iterations to be sufficient for all experiments.

3.5 Validation

We validate the key aspects of our method in a number of ways: comparisons to manual consolidation, comparison against prior art, and qualitative evaluation.

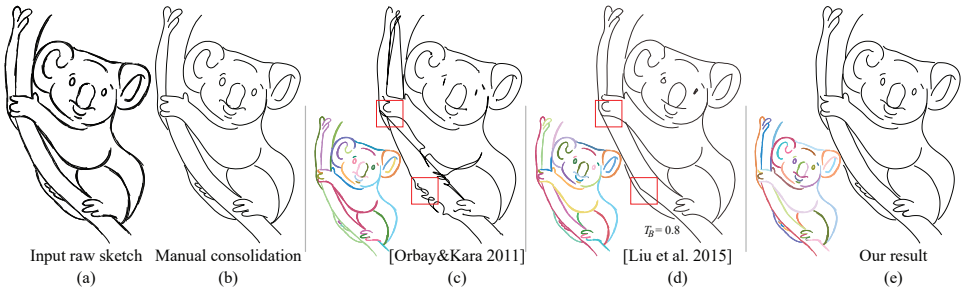


Figure 3.18: Consolidation comparison (clusters and fitted curves): (a) input; (b) manually consolidated drawing; (c) Orbay and Kara [78]; (d) Liu et al. [69]; (e) our result. While the results of prior methods exhibit a range of artifacts, our result (e) is consistent with the manual consolidation output (b). Raw sketch: © Enrique Rosales. Manual consolidation: © Elinor Palomares.

Comparison to Manual Consolidation. Our method aims to recover the viewer-perceived consolidated curve set from the input drawings; therefore the key criterion for assessing it is via a comparison to manually consolidated results. We perform two separate comparative studies.

The first study has two goals—to asses how consistent humans are in their consolidation choices given a collection of strokes, and to compare human consolidation choices to our algorithmic ones. We picked 28 samples of different stroke configurations selected from a diverse set of 14 drawings. We then asked 10 participants (4 artists and 6 non-artists) to draw the curves they perceive these strokes to represent: "You will examine different images in which you will have to trace a clean version of the strokes you see." The combined results for a subset of the inputs (blue) superimposed with our output (red) are shown in (Figure 3.17). The results show that human observers are generally consistent in their consolidation choices. For 80% of the inputs, at least 8 out of 10 participants provided the same curve configuration. On only 2 out of 28 inputs (including one in Figure 3.17) the participant configuration choices were evenly split. In all cases, StrokeAggregator's result was similar to the plurality response.

In our second study, we selected seven complete input drawings and asked an artist to consolidate them. Figures 3.1 and 3.18 show two such artist results side-by-side with our outputs. As these comparisons show, our results are well-

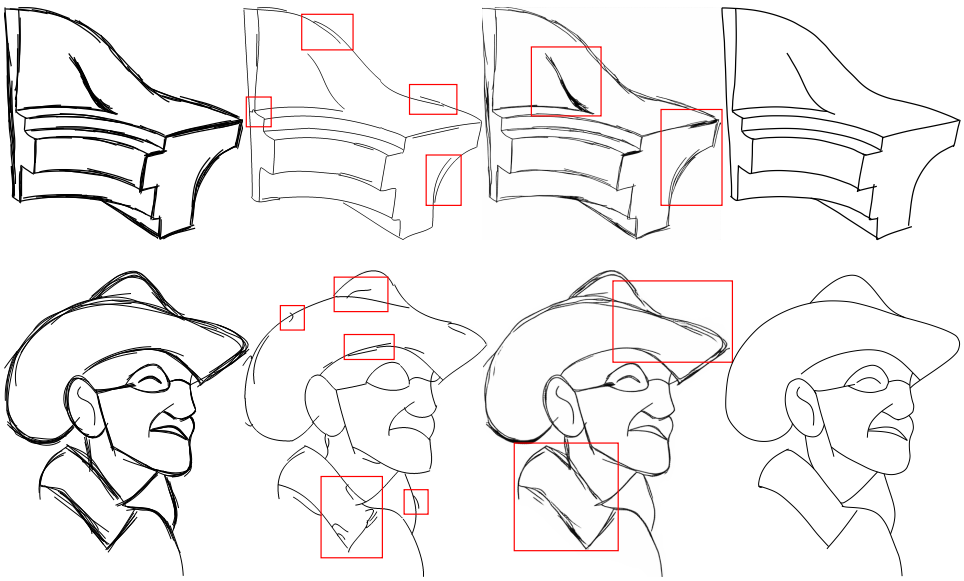


Figure 3.19: Comparison with raster cleanup and vectorization methods. The top input is from Orbay and Kara [78]; the bottom input is from Liu et al. [69].

aligned with the artist outputs. It took the artist 10 to 30 minutes to create each consolidated output, significantly larger than our automatic consolidation times of 1 to 8 minutes.

Comparison to Prior Art. We compare our framework against the most recent alternatives. Figure 3.19 compares our output against two raster-space methods for vectorization [35] and cleanup [97]. To perform the comparison, we rasterized the drawings at their original resolution using standard software and ran the executable kindly provided by the authors. As shown by the results, both raster methods fail to fully consolidate the strokes when presented with drawings containing thick stroke clusters. Our method successfully consolidates these inputs. The failure of these methods is unsurprising, as raster-space methods rely on less information. It also suggests that directional information, which is increasingly available due to the wide usage of tablet displays, benefits our consolidation task.

Figures 3.18, 3.20, and 3.21 show comparisons to vector consolidation meth-

Input raw sketch (a) | [Orbay and Kara 2011] (b) | Our result (c)

Figure 3.20: Comparison (clusters and fitting) with Orbay and Kara [78]. Inputs sourced from Liu et al. [69]. In column two the examples of wrongly clustered strokes are highlighted.

ods [69, 78]. The results in these figures were provided by the authors. As shown in Figures 3.18 and 3.20, the method of Orbay et al. frequently fails to separate connected clusters resulting in poor consolidation outputs, on a range of inputs on which our framework produces the expected results. Figures 3.18 and 3.21 compare our results with those of Liu et al. Our method achieves better fine input feature preservation, while still correctly consolidating wide large-scale clusters. The results of the method of Liu et al depend on a user provided parameter T_B (listed in the figures), and those of Orbay et al. depend on the choice of the input training sketches. Our outputs are produced with no additions input or parameter adjustment.

Qualitative Evaluation. We also conducted a study to compare our outputs to artist outputs and previous work. We asked 20 participants to compare our outputs to consolidated drawings generated by alternative methods [35, 69, 78, 97] and artists (5 each of [35, 78, 97], 6 [69], and 5 artist). Each query in this study included an input drawing ("Original", top) and two consolidated outputs ("(a)" and "(b)", bottom), arranged in random order and presented side-by-side: one gener-

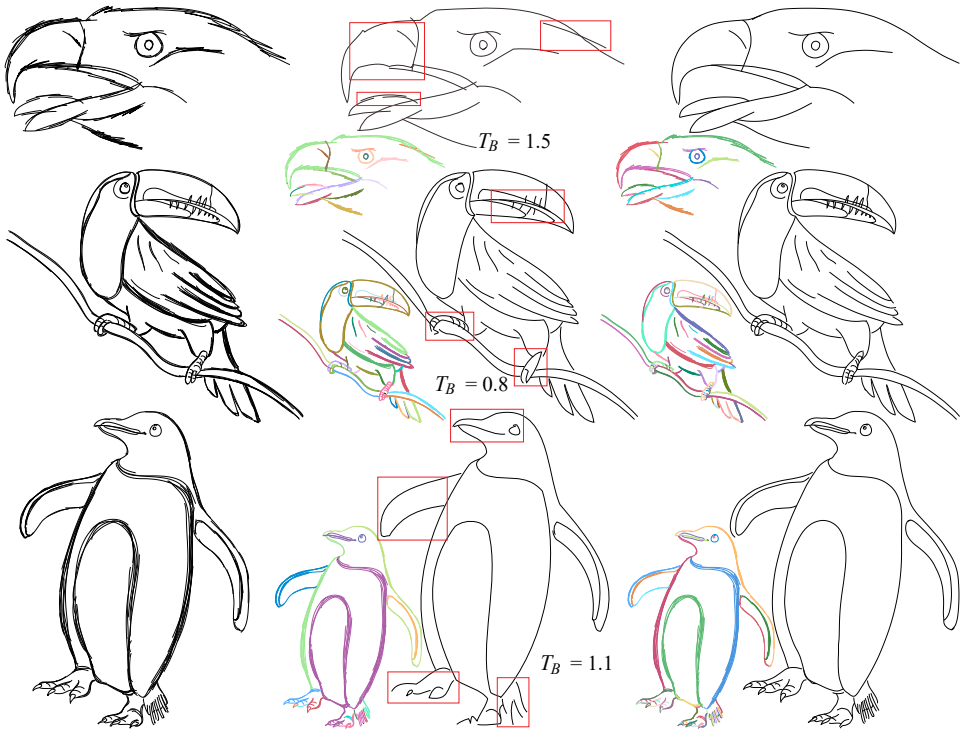


Figure 3.21: Comparison (clusters and fitting) with Liu et al. [69]. Note the differences in the consolidation of feet and other fine features. The eagle input is sourced from Orbay and Kara [78]. Toucan, penguin: © Enrique Rosales.

ated by our algorithm, and one generated by an alternative method or by an artist. We asked "Which of the drawings below, '(a)' or '(b)' is a cleaner and accurate version of the drawing on top 'Original'? If both are, please select 'both' if neither select 'neither"'. Our results were judged on par with those created by artists; in comparisons with artist results, viewers selected "both" 50% of the time, and preferred our result 25% of the time. In a comparison with prior work, our results were overall judged as superior 92% of the time. In comparisons to the method of Liu et al [69], our results were preferred 80% of the time and ranked on par 17% of the time. In comparisons to other methods, our results were judged as superior 97% of the time. These numbers validate that our methods performance is on par

Figure 3.22: Additional diversely sourced results. The duck input is sourced from Liu et al. [69], the architectural model and man are sourced from Orbay and Kara [78], shark and triceratops: © Cristina Arciniega, flower and bow-tie: © Enrique Rosales. Please zoom in online to see image details.

with manual consolidation and is far superior to prior art.

3.6 Results

We tested our method on 36 inputs with sizes (measured in pixels) ranging from approximately 300x400 to 1000x800, 20 of which are shown throughout this chapter. Our inputs include examples sourced from prior work, e.g. fandisk (Figure 3.19), eagle (Figure 3.21), man and opera (Figure 3.22) are from [78], and grandpa (Figure 3.19), duck (Figure 3.22), and fairy and car (Figure 3.20) are from [69]. They also include new inputs created by two different artists (e.g. Figures 3.1, 3.18, shark, bowtie and triceratops in Figures 3.22). Our inputs include relatively clean drawings with few overdraws (bowtie, man) and very sketchy drawings with large clusters of overdrawn strokes (penguin, toucan, grandpa). We include both drawings of organic shapes (toucan, pig, penguin), as well as design drawings of free-form and regular shapes (opera, fandisk, car). Our framework produces results consistent with viewer perception on all these inputs.

Impact of Design Choices. Figures 3.6, 3.8, 3.12 show the stages of our progressive cluster refinement process, highlighting the contribution of each stage. The local analysis stage (Section 3.3.2) is critical for processing clusters with branch-

ing structures (Figures 3.6b, 3.8b). The narrowness cue is critical in processing features such as the tail of the penguin (Figure 3.21), the moon shaped windows on the building (Figure 3.23), or the stripes on the side of the shark (Figure 3.22).

Runtimes. Our method takes on average 2.5 minutes to consolidate a drawing. Approximately 50% of the time is spent in the final unification, given we exhaustively assess pairs of nearby clusters in this stage. The rest of the runtime is split between the angular compatibility stage and the proximity refinement stage with a ratio of $1:4$. In our inputs, the numbers of strokes range from a couple of dozens to 300 and the number of clusters range from 15 to 140 (toucan).

Limitations While human observers likely base some of their mental consolidation decisions on content recognition (Section 3.5), our method relies only on local stroke context. Thus, it may fail in situations where stroke level cues become unreliable. In particular, our method targets inputs where overdrawing is used for the purposes identified in Chapter 2, and is not directly applicable to stylized line drawings (Figure 3.23ab). In such drawings strokes are used as expressive paint-brushes and their tangents no longer reflect the tangent of their corresponding aggregate curves. In this setting, our core cue of angular compatibility between cluster strokes fails. Figure 3.23c shows another example where local context and global image recognition may result in different consolidation outputs. While our result is consistent with human grouping given local context only (left), one may argue that in the global context (right) humans would group the highlighted vertical strokes together to form a building corner.

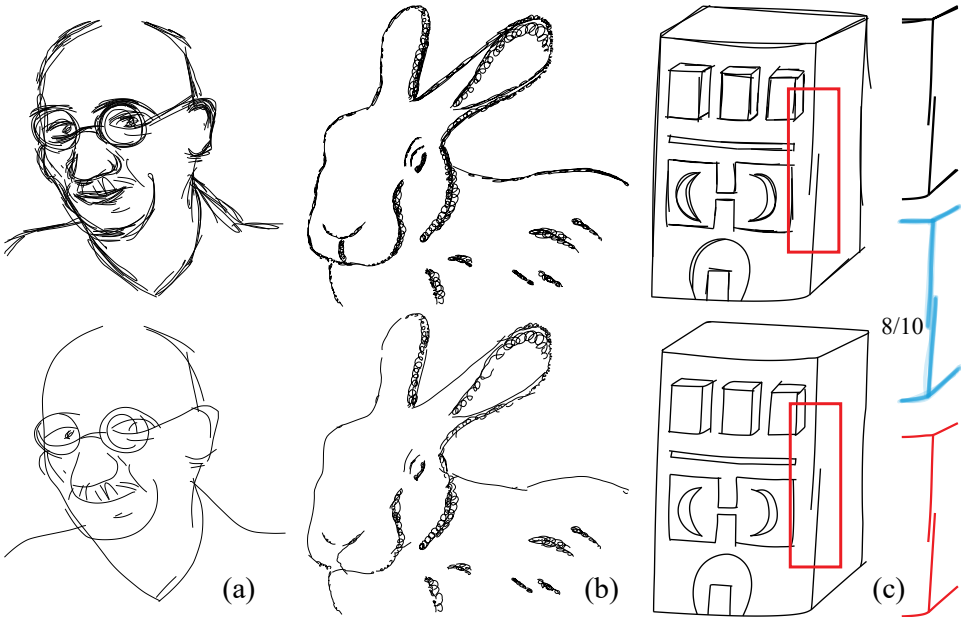


Figure 3.23: Our framework relies on local context rather than recognition. Thus its ability to process intentionally sketchy (a) or stylized inputs with unreliable stroke tangents (b) is limited. (c) Our clustering choices on this input are consistent with local human ones (8 out of 10 viewers keep the strokes separate given purely local context (left)), and do not account for global context which humans rely on given the complete image (right). Bunny: © Elinor Palomares.

Chapter 4

Detecting Viewer-Perceived Intended Vector Sketch Connectivity

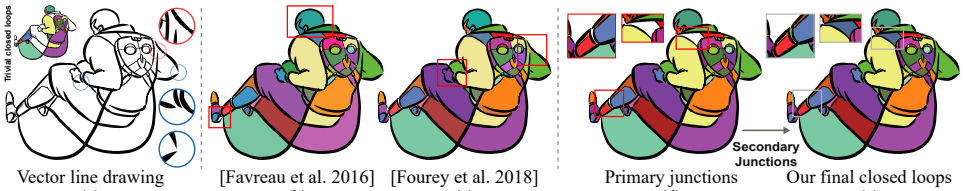


Figure 4.1: A typical freehand vector line drawing (a) and the connectivity indicated by intersections only (a, top left), by two previous work [35, 38] (b, c) and by our method (d). Each closed loop interior colorized with a different color, with the background left white. Please zoom in to see image details throughout the paper. Input image ©The "Hero" artist Team under CC BY 4.0.

Freehand vector line drawings are often imprecise with strokes intended to intersect stopping short of doing so. Because of this inaccuracy, even after being cleaned up by consolidation method, e.g., StrokeAggregator in Chapter 3, consolidated sketches need to be further prepared for downstream processing, such as colorization (Figure 4.1(a)). While human observers easily perceive the artist intended stroke connectivity, manually, or even semi-manually, correcting drawings to generate correctly connected outputs is tedious and highly time consuming. In this chapter, we propose a novel, robust algorithm that successfully extract viewer perceived intended stroke connectivity distinguishing between intended junctions (e.g., circled in blue in Figure 4.1(a)) and intended gaps (e.g., circled in red in Figure 4.1(a)) outperforming prior art. We arrive at this solution by leveraging observations about local and global factors that impact human perception of inter-stroke connectivity.

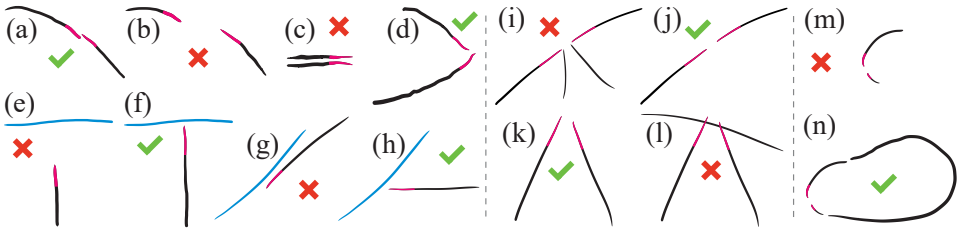


Figure 4.2: Human observers employ local and global cues to determine if a dangling endpoint (red) is intentional, or is intended to be part of a junction. As highlighed in (ab) and (ef), distance is a major factor in distinguishing between intended junctions (af) and intended gaps (be). Different tangent directions can impact the perception of junction intent for endpoint (cd) or endpoint and stroke pairs (gh) at the same distance from one another. (i-n) The presence of other strokes can change the perception of whether strokes do or do not form junctions.

4.1 Introduction

Free-form line drawings are ubiquitous, both as an art form and as inputs to different computer applications. Thanks to the broad availability of touch pen and stylus devices, such drawings, especially those that artists intend to process later with different algorithmic tools, are increasingly recorded and stored in vector format [15, 33, 45, 79]. Drawing processing applications typically require inputs with precisely identified stroke intersections (Fig. 4.1e). Unfortunately, artist drawings are inherently imprecise [55], and routinely contain *unfinished* strokes that artists intend to intersect other strokes, but that stop short of doing so (Fig. 4.1a) [82, 101, 112, 114]. While human observers easily mentally complete unfinished strokes [57, 101] and identify the intersections they are *intended* to form, extracting this intended connectivity algorithmically remains an open challenge (Sec. 2.5). We propose a new perception-driven algorithm for identifying intended intersections that produces outputs well aligned with human perception and significantly outperforms the state of the art.

Locating intended junctions algorithmically is highly challenging, as little research exists on the cues that humans employ when mentally separating intended intersections from intended *gaps*. Observations of manual annotations of intended junctions and intentionally dangling stroke *endpoints* (Fig. 4.2), suggest that viewers leverage both local and contextual cues when making such decisions (e.g. the

relation between the strokes in Fig. 4.2m is not evident, but these same strokes are seen as clearly forming an intended end-to-end junction given the extra stroke in Fig. 4.2n). We use a combination of perception literature review, observation of artist drawings, and manually annotated junctions to identify the factors that determine whether humans perceive dangling stroke endpoints as parts of intended junctions or not (Sec. 4.2). We hypothesize that these decisions are impacted by both local and global cues, and thus depend both on the geometry of the strokes in question (Fig. 4.2a-h) , their immediate surroundings (Fig. 4.2i-l) and on the more global drawing context (Fig. 4.2mn).

A possible approach for addressing perception motivated tasks is to learn the desired outcomes from human annotated data. Applying this approach in our context raises two conflicting challenges. First, the global nature of the human decisions noted above means that a purely learning-based method would require a very large body of fully annotated sketches to adequately perform the task at hand. At the same time, typical drawings contain many dozens of strokes, and manually separating all intended junctions from intended gaps in a single drawing takes 20 minutes or more on even moderately complex sketches using interfaces optimized for this task (Sec 4.5); manually annotating large collections of drawings is thus impractical. We overcome these difficulties by developing a hybrid approach which combines data-driven and perception-driven components, and allows us to robustly compute outputs well aligned with human perception from just 31 drawings with partial manual annotations.

We leverage the collected annotations to design two local classifiers. Our first classifier uses the local geometry and context of a pair of dangling endpoints to predict how likely they are to form an end-to-end junction. Our second classifier uses similarly local information to predict the likelihood that a dangling endpoint and a stroke form a T-junction. Both classifiers utilize geometric features we expect to strongly correlate with human perception of junctions, and are trained on our collected manual annotations. Our classifiers achieve an accuracy of 99% in leave-one-drawing-out cross-validation. We embed these classifiers in an incremental decision making process that combines purely local considerations with global properties. It first performs basic pairwise classification across all pairs of valence two end-end and simple T-junction candidates, identifying *primary* junc-

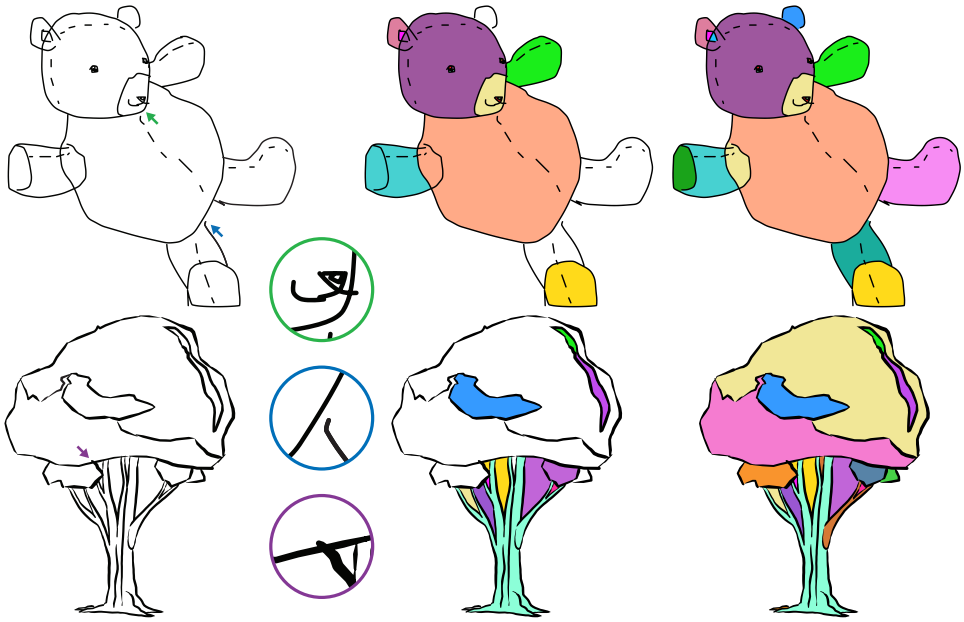


Figure 4.3: Typical free-hand drawings (a) contain jaggy, fragmented, and over-drawn strokes (pointed and circled in green), unintentionally dangling endpoints (pointed and circled in blue) and strokes that extend past their intended end-junctions (pointed and circled in purple). Directly extracting closed stroke loops from such drawings (b) produces heavily under-segmented outputs. By identifying unintentionally dangling endpoints and forming intended junctions we form loops consistent with viewer expectations. Top input image ©Mathias Eitz, James Hays and Marc Alexa under CC BY 4.0. Bottom input image ©The "Hero" artist Team under CC BY 4.0.

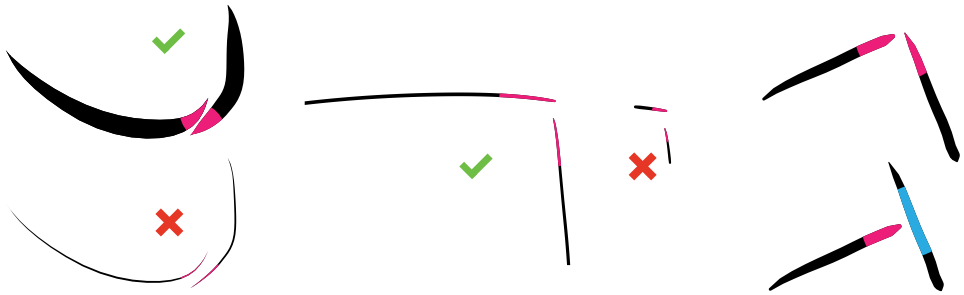
tions (Fig. 4.1d); it then leverages these decisions to form high probability, more complex, *secondary* junctions, and finally uses all previously made decisions to analyze global context and identify junction candidates whose likelihood of forming junctions is strongly boosted by global perceptual cues (Fig. 4.1e).

We design and test our method to operate on free-hand artist drawings which exhibit a range of inaccuracies and drawing artifacts (Fig. 4.3). We evaluate our method on 95 diversely sourced inputs, and validate it via comparisons to manual annotations and a perceptual study comparing our outputs against prior art. In our

annotation comparison our method achieves comparable accuracy to human annotators (92% versus 94%). Comparative study participants preferred our results by a factor of 9 to 1 over the best performing competitor (Sec. 4.6). These advancements are made possible by leveraging novel insights about human perception of intended junctions and converting those into an actionable robust algorithm for distinguishing intended intersections form intended gaps.

4.2 Perception of Intended Sketch Connectivity

The goal of our algorithm is to identify intended junctions and gaps as perceived by human observers. Like prior work [94, 109] we operate under the assumption that artists aim for their drawings to be well understood, and that viewer understanding of the drawings is in general consistent with artist intent. As in many similar problem settings, one of the big challenges in detecting the intended connectivity is that the exact mechanism observers use to mentally perform this task is unknown. At the same time, our review of related literature points to a number of cues observers are likely to use when identifying intended junctions (Fig. 4.2).



(a) (b) (c)

Figure 4.4: Stroke-pair properties. The inter-stroke distance is relative rather than absolute (ab). The relative location of the projection of the endpoints affects the perceived junction type (c).

Stroke-Pair Properties Prior research [23, 38, 81, 82] convincingly demonstrates that *distances* between potentially connected strokes play a large role in the perception of intended junctions. Notably, the perception of inter-stroke distance is

relative rather than absolute: wider strokes are perceived as closer to one another than thin strokes located at the same distance (Fig. 4.4a), and same distance longer strokes are seen as closer than shorter ones (Fig. 4.4b). Prior research [23] further suggests that stroke *directions* at the evaluated potential junction locations serve as a similarly strong cue (e.g. the distance between the pairs of strokes in in Fig. 4.2cd is identical but due to different endpoint directions we view the strokes in Fig. 4.2d as intended to connect, and the ones in Fig. 4.2c as not). Lastly we note that strokes are perceived as more likely to form an end to end junction if their nearest points are at or near their endpoints and a T-junction if the endpoint of one is close to the *middle* of the other (Fig. 4.4c); notably this distinction suggests that the type of intended junction formed by the strokes depends on the *relative location of the projection* of the endpoints of one stroke onto the other.

Local Context We note that perception of intended junctions is impacted not just by the geometry of the participating strokes but by both local and global context. Specifically the presence of other strokes in the immediate vicinity of the assessed pair can impact the perception of junctions (Fig. 4.2i-l), providing *local context*. In particular the presence of *nearby intersections* between the assessed and other strokes impacts the perception of whether an endpoint is dangling or not, and thus the expectation of its stroke being part of any additional junctions.

Closure. Lastly, and critically, we note that the *closure* principle of Gestalt psychology [61, 104] is highly relevant when analyzing perceived sketch connectivity, as it suggest that viewers are highly likely to mentally close gaps between strokes if doing so results in formation of closed loops, or regions. Liu et al. [69] utilize this principle for sketch cleanup, merging together overdrawn strokes perceived to bound the same region. They suggest that to evaluate whether a sequence of strokes (or stroke segments) is forming a perceived loop one can consider the ratio between the diameter D of the biggest circle inscribed inside the loop and the length of the largest gap L between consecutive strokes in the sequence as illustrated in Figure 4.6,

$$R_C = \frac{D}{L}. \tag{4.1}$$

Figure 4.5: Method Overview. Given a vector line drawing (a), we first detect trivial stroke-wise intersections forming closed stroke loops (b, right). We then identify likely end-to-end (red) and T- (blue) junctions (b, left zoom-ins). With these pairs and their predictions, we constructs primary junctions, supporting arbitrary valence (c, see left zoom-ins for examples). We proceed to identify *secondary* T-junctions formed by the remaining dangling endpoints and composite strokes (d, see left zoom-ins for example connections). In the final closure integrated step, we close remaining undesirable gaps by jointly evaluating classifier predictions and gap ratios along the boundaries of potential cycles (e, see the zoom-in for a connection classified as marginally negative in our primary step and accepted in this step). Input image ©The "Hero" artist Team under CC BY 4.0.

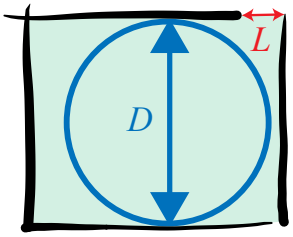


Figure 4.6: The gap ratio $R_C = D/L$ quantifies the gap size relative to the adjacent region reflecting the closure property of Gestalt psychology.

The larger this *gap ratio* is the more likely viewers are to perceive the strokes as forming a loop. We observe that conversely, when the ratio is sufficiently small, viewers are unlikely to perceive strokes as forming a loop even if other cues suggest otherwise. In particular our analysis of manually annotated sketches (Sec. 4.6) suggests that viewers do not mentally close gaps between strokes when doing so results in loops with the ratio R_C being below 1. We refer to this property as *minimal cycle ratio*. This property is connected to topological persistence in the curve and surface reconstruction literature [27, 91].

4.3 Algorithm

Our method takes as input raw, free-hand sketches collected in the wild (Sec. 4.6). We process drawings with both constant and variable width strokes. We preprocess these drawings removing hooks, merging overdrawn strokes, and detecting all *trivial* stroke intersections (locations where pairs of strokes overlap) as discussed in Appendix A.2. The output of our method is a set of intended junctions between pairs of locations on these strokes, where these strokes do not overlap a priori in the drawings. We specifically focus on junctions formed by connecting stroke endpoints with either other endpoints (end-to-end junctions) or with locations along other strokes (T-junctions). Our last closure-aware step (Sec. 4.5) considers both junctions involving endpoints and those involving pairs of nearest mid-stroke locations on adjacent strokes.

We identify these intended junctions by leveraging a combination of the perceptual cues listed above and ground-truth data. While access to ground truth data is highly beneficial for producing results consistent with human perception, a core challenge we face is data scarcity: as noted in Sec. 4.1 annotating intended junctions is a time consuming and mentally non-trivial task. Typical artist drawings (e.g. Fig. 4.1) have over a hundred strokes ; and take 20 minutes or more to annotate. Thus we require an approach capable of correctly detecting intended intersections using limited data.

We achieve these goals using a method that relies on a combination of four key elements.

Pairwise Junction Classifiers. We utilize the collected annotations to train two classifiers, one for predicting how likely a pair of stroke endpoints is to form an end-to-end junction and one for predicting how likely an endpoint and a stroke are to form a T-junction (Sec. 4.4). Our classifiers utilize a compact set of features encoding the properties of the evaluated pairs and their local context.

Filters. We utilize the perceptual cues above to narrow down the set of endpoint and endpoint-stroke pairs that the classifiers are applied to by automatically discarding *impossible pairs*, i.e. ones that humans are virtually guaranteed to *not*

see as forming intended junctions. This filtering allows us to dramatically reduce the number of negative (perceived as not forming a junction) training examples we need to collect (naive classifier would otherwise need to evaluate all pairs of strokes and endpoints in a drawing against one another). It also allows us to reduce the number of features that the classifiers operate on and thus further reduce the amount of negative and positive training data they require to achieve robust performance. Our pre-filtering considers both the properties and the local context of the classified pairs.

(a) (b)

Figure 4.7: High-valence junctions in two example configurations.

Intended High-Valence Junctions Freehand sketches can contain junctions with arbitrary high valence; including both complex end-to-end junctions and ones connecting multiple endpoints to a shared T-junction point (see Figure 4.7). Notably, while many drawings contain such junctions, they each contain only a handful of them. This sparsity means that while in theory one could train separate classifiers for different high-valence junction topologies, collecting enough data to train such classifiers would require annotating a very large number of drawings. We overcome this challenge by developing a junction processing workflow where we successfully utilize the pairwise junction classifiers to predict the likelihood of higher valence junctions (Sec. 4.3, 4.3).

Closure-Aware Classification. While the perception of closure plays an important role in human perception of stroke connectivity, closure evaluation is inherently global in that it involves considering multiple strokes and multiple potential

intended junctions at once. As such it is hard to account for using only pairwise or other purely local classifiers; learning closure based choices directly would therefore require a dramatic increase in the amount of training data. We address closure in a data efficient manner by using a delayed decision process where in the first rounds of our computation, closure, and specifically, minimal cycle ratio, is only used as a hard negative constraint preventing us from forming junctions that would create loops violating this constraint. Once all local decisions are made we efficiently compute potential loops induced by these decisions and revisit all prior negative classification decisions, incorporating closure as a positive cue (Sec. 4.3).

Armed with these tools we formulate our intended intersection detection as the following gradual decision process (Fig. 4.5).

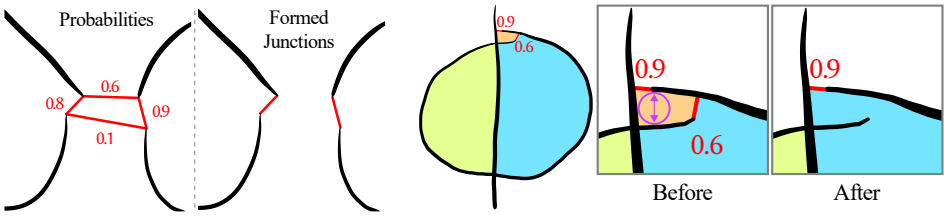
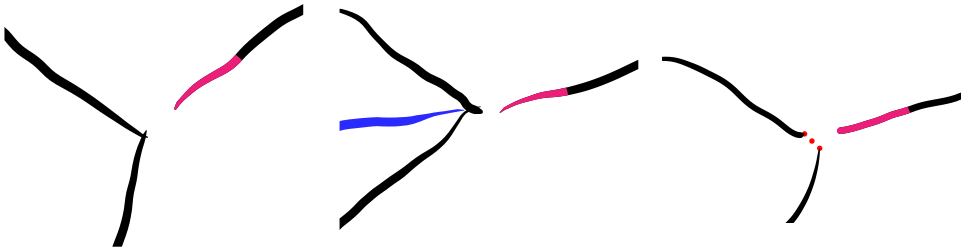


Figure 4.8: When solving for primary junctions, the method picks the best high-valence junction configuration (a) while avoiding creating small cycles.

Primary Junctions Classification

Our primary junctions classification step first identifies pairs of endpoints or endpoints and strokes that have the potential to form intended junctions and then uses our two classifiers to compute the probability of each pair forming such a junction (Sec. 4.4). It uses this information to form likely binary and high-valence junctions when doing so does not violate our minimal cycle ratio constraint. Specifically, when the classifier deems two or more pairs containing one or more same endpoints as each being likely to form an intended junction we form the subset of these interconnected junctions that maximizes the joint probability across them (see Figure 4.8a). After making these decisions, we evaluate the gap ratio (Eq. 4.1) for each newly formed cycle; if a cycle violates the minimal cycle ratio constraint ($R_C < 1$),

we break it by removing the lowest probability junction along its perimeter (see Figure 4.8b).



(a) (b) (c)

Figure 4.9: When solving for secondary junctions, strokes that are connected by trivial junctions (a) and previously detected intended junctions (bc) are considered to be a single stroke.

Secondary Junctions Classification

Our *secondary* classification step addresses the formation of intended high-valence junctions. Specifically, it analyzes remaining dangling endpoints and nearby previously identified junctions and evaluated whether the dangling strokes should be extended to connect to these junctions. The previous junctions considered include both trivial junctions (Figure 4.9ab) and previously detected intended junctions (Figure 4.9c). Our evaluation leverages two observations: we first note that two strokes whose endpoints are part of a common junction can be conceptually seen as a single *composite* stroke (pairs of black strokes on the left of each inset in Figure 4.9), thus for our purposes we can evaluate if a dangling endpoint should be connected to such a junction utilizing our T-junction classifier and applying it to the dangling endpoints and such *composite* strokes. While at a high-valence junctions one can potentially composite multiple strokes (Figure 4.9b), given a dangling endpoint, humans are likely to only consider the composite stroke (black in Figure 4.9) formed by the two strokes that are immediately next to the assessed endpoint (with respect to a circular ordering around the junction location) and ignore those occluded by these two (blue in Figure 4.9). This observation again suggests that we can evaluate if a dangling endpoint should be connected to such a high-valence

junction utilizing our T-junction classifier by applying it to the dangling endpoint and temporary *composite* strokes formed by the strokes (or portions of strokes) that are part of the junction and are immediately next to the endpoint. We process all classifier decisions utilizing such composite strokes using a process identical to the one in Sec. 4.3.

Global Closure-Aware Classification

Our final, *closure integrated step* forms potential stroke cycles containing remaining dangling endpoints or pairs of nearest points on adjacent strokes and uses a combination of previously computed classifier probabilities at these endpoints and the gap ratios along these cycles to determine whether the remaining gaps along these cycles should be closed or remain open (Sec. 4.5).

4.4 Junction classifier

End-end classifier. Given endpoints p_1 and p_2 on strokes S_1 and S_2, we construct four sets of features, motivated by the perceptual cues in Sec. 4.2. In the description below, each asymmetric feature, reported for p_1, is computed for both endpoints; the minimum and maximum over both values are used to make the classifier commutative by design.

Distance. We use three features to encode the distance between the stroke endpoints. We measure the distance between the stroke envelopes $d^E = \|\mathrm{p}_1 - \mathrm{p}_2\| - \frac{1}{2}(w_1 + w_2)$ (see Figure 4.10a). We convert this distance into three viewer-perceived scale-invariant features by normalizing it by (1) the mean of the maximum width of each stroke (W_1, W_2), and by the (2) min and (3) max of the stroke lengths (L_1, L_2).

Directions. We use four types of features to encode the interaction between the directions of the two strokes at the endpoints of interest overcoming drawing inaccuracies. We codify the *type* of the junction, characterizing it as belonging to one of the three categories in Figure 4.10b, by counting whether two, one, or zero of the endpoints project onto the opposing stroke endpoints. We include the angles θ_1, θ_2 between the stroke tangents and the line connecting the two endpoints (see Figure 4.10c); we compute the tangent $\vec{t}_1$ by stepping back from the endpoint along

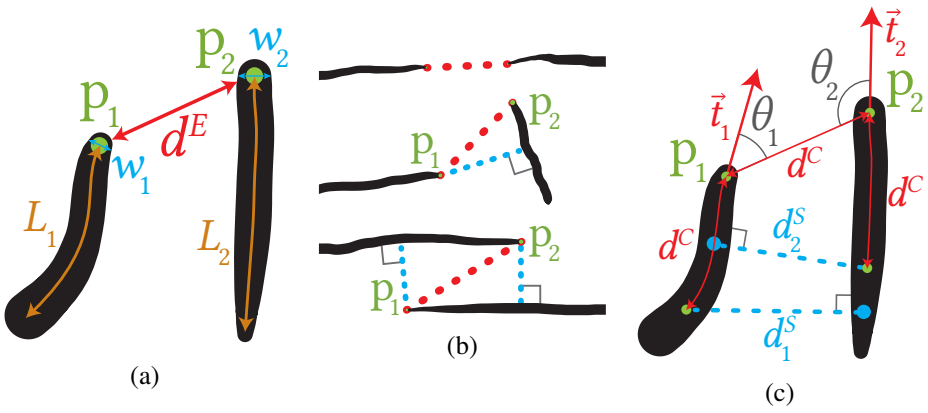


Figure 4.10: Measurements of distance and direction.

the stroke by the distance $d^C = \|p_1 - p_2\|$. We also include two ratios that are even less susceptible to noise than the tangents: the *step-away ratio*, measured as the distance d_2^S from the step-away point to S_2 divided by d^C, and the *projection ratio*, the distance from p_1 to S_2 divided by d^C.

Relative Location. We encode the distance between the projection of p_1 on S_2 and its closest endpoint along S_2, normalized by L_2.

Local Context. We encode local context as the distance from p_1 to the closest stroke in the drawing other than S_1 and S_2, normalized by gap size d^C.

T-junction classifier. We use similar features for the T-junction classifier, modifying them to account for its asymmetric nature. To this end, we compute distance measures between endpoint p_1 on S_1 and the closest point p_2 on stroke S_2, the directional features are computed only from the endpoint to the other stroke, and we skip the projection ratio since the projection and p_2 are the same for T-junctions. When computing local context, points that are occluded by S_2 are excluded. We incorporate larger context for T-junction decisions than for end-end ones by using an additional *endpoint density* feature, defined as a function of the distances from this endpoint to all endpoints,

$$b = \sum_{p_e \in \{\text{endpoints}\}} e^{-\frac{1}{2}\left(\frac{1}{\sigma}\frac{\|p_1 - p_e\|}{w_e}\right)^2}.$$

Table 4.1: Gini importances of junction classifier features.

Endpoint-endpoint features	Gini importance
Envelope distance $d^E/(0.5(W_1+W_2))$	0.246
Envelope distance $d^E/\max(L_1,L_2)$	0.171
Envelope distance $d^E/\min(L_1,L_2)$	0.160
Junction type	0.001
$\max(\theta_1,\theta_2)$	0.015
$\min(\theta_1,\theta_2)$	0.011
Larger step-away ratio $\max(d_1^S/d^C, d_2^S/d^C)$	0.031
Smaller step-away ratio $\min(d_1^S/d^C, d_2^S/d^C)$	0.027
Larger projection ratio	0.001
Smaller projection ratio	0.009
Larger relative location	0.009
Smaller relative location	0.0004
Larger distance to nearest other	0.190
Smaller distance to nearest other	0.128
T-junction feature	**Gini importance**
Envelope distance $d^E/(0.5(W_1+W_2))$	0.250
Envelope distance d^E/L_1	0.174
Envelope distance d^E/L_2	0.270
θ_1	0.067
Step-away ratio d_1^S/d^C	0.077
Relative location	0.012
Distance to nearest other	0.136
Endpoint density b	0.015

The contribution of each endpoint is a Gaussian of the distance to it normalized by w_e, the average of the widths along its stroke. We use $\sigma = 1$, which ensures that endpoints fall to a negligible contribution when they are 3 stroke widths away.

Training. We implement both the endpoint-endpoint and T-junction classifiers as random forests that are trained on either endpoint or endpoint and stroke pairs that are labelled as intended or unintended junctions.

List of Features. A list of the classifier features we use and their Gini importance values are listed in Table 4.1 where important features have larger values. All features contribute to the decision making. The top five important features of end-end classifier are envelope distance $d^E/(0.5(W_1+W_2))$, larger distance to nearest other, envelope distances $d^E/\max(L_1,L_2)$, $d^E/\min(L_1,L_2)$, smaller distance to nearest other. They corresponds distance and local context cues. Similarly, the top five important features of T-junction classifier are envelope distance d^E/L_2,

$d^E/(0.5(W_1+W_2))$, d^E/L_1, distance to nearest other, step-away ratio d_1^S/d^C. They corresponds distance and local context cues, followed by the direction cue. This suggests the perceptual cues influencing both end-end and T-junction classifiers have similar importance order: distance, local context, and direction are the top three most important features in descending order. The top two cues suggest that both local and contextual cues are important to a similar degree.

4.5 Algorithm Details

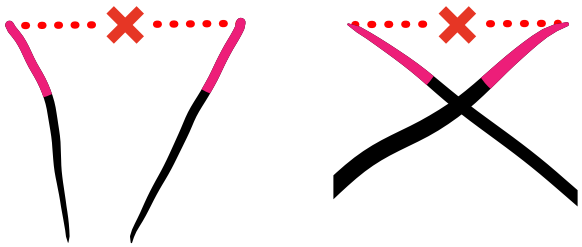


Figure 4.11: End-end pairs with diverging directions are filtered.

Filtering Junction Candidates The number of possible junctions grows quadratically with the number of endpoints, and the vast majority of end-end or end-stroke pairs are not intended to connect. Our filtering avoids a massive imbalance between positive and negative examples during training, reduces the runtime cost of our method, and drastically reduces the number of negative annotations we need to collect for training. Following the observation about the impact of local context, for each endpoint, we only consider connections with the closest three endpoints and strokes, respectively. We further filter those that are too far, lack a line of sight, or have diverging directions (Figure 4.11). We similarly prevent end-to-end junctions between parallel strokes. The closest-three filter reduces the number of pairs from quadratic to linear, and the later filters further reduce the number of pairs passed to the classifier by a factor of 2 or more on a typical input.

Global Closure-Aware Classification At this point of the process, we expect the vast majority of intended intersections to be appropriately classified, enabling us to

compute meaningful stroke cycles containing only a handful of unintended gaps.

We first locate all pairs of closest points on immediately adjacent strokes; for each pair we compute the cycles formed by connecting the pair; we mark the pair as forming an intended junction if the distance between its points is smaller than the length of the largest previously closed gap along these cycles and $R_C > 20$ for both cycles.

Our closure-aware step then leverages the probabilities computed by the classifier and combines those with the evaluation of the closure ratio R_C (Eq. 4.1) to identify pairs of strokes that are likely to form junctions once closure is accounted for. Our analysis of training data suggests that the closure ratio can be viewed as a boost signal, increasing the likelihood of a gap being intended by approximately a linear factor. In other words, given a ground truth probability P of the end-points of a gap forming a junction, the gap ratio boosts this probability to $P' = P + C(R_C - 1)$. In practice, the probabilities provided by our classifier are approximate. We thus use a conservative step-function approximation of the formula above with $C = 0.025$. Using a step size of 0.05 and starting at $P = 0.45$ for each gap with classifier probability of P and above we close the gap if $P' > 0.5$.

Specifically, at each step value of P we order all junction candidate pairs with probability P or more; for each pair we compute the cycles formed by connecting the pair. We mark the junction as intended if $P' > 0.5$ for both cycles. We repeat this process considering two candidate pairs at once. We have not encountered cases for which evaluating three or more pairs was necessary.

Random Forest. Our random forest classifiers have 100 trees each. We limit the maximum depth to 10 for the endpoint-endpoint classifier and 12 for the T-junction classifier. We use the scikit-learn library [85] for all training.

Training Set. We trained our method using 31 sparsely annotated drawings. In assembling this set we aimed for a diverse set of sources spanning different styles, content and levels of expertise. Our training set consists of 13 drawings from the Blender Art Gallery [14], 5 from *Quick, Draw!* [47], 2 design sketches from OpenSketch [44] and 11 original drawings. In total, we have 290 positive endpoint-

endpoint examples, 1778 negative endpoint-endpoint examples, 460 positive T-junction examples, and 2817 negative T-junction examples. The annotatins were created using an in-house interface that incrementally colorizes regions based on user annotations.

Closing gaps. We visualize closed gaps by using shortest straight lines connecting participating endpoints and strokes. To close gaps in a geometrically-pleasing manner, one can use the method of [54] or modify a curve-fitting method (e.g. [79]) to enforce junctions.

4.6 Results and Validation

We tested our method on 95 previously unseen inputs from a diverse set of sources spanning different styles, content and levels of expertise. To this end we include 30 professional drawings of characters and organic shapes created using the Blender Grease Pencil Tool and provided in the Blender Art Gallery [14]; rough amateur sketches, including five each from [47], [33] and Ge et al. [40], and 10 each from [92] and [87]. We also included 11 drawings of polyhedra from [23], and one input from Jiang et al. [54]. In addition to these raw inputs, we applied our methods to pre-consolidated sketches: 8 from StrokeStrip [79] and 3 from OpenSketch (using the ground truth consolidations for the former, and the StrokeAggregator (Chapter 3) consolidations for the latter), as well as algorithmic vectorizations of 7 raster drawings from Parakkat et al. [81, 82]. Representative examples are shown in the paper.

We validate the key aspects of our method in a number of ways: we evaluate our classifiers using leave-one-drawing out cross-validation, evaluate our methods final classification decisions by comparing them against manual annotation, and compare our method to algorithmic alternatives via a comparative user study.

Classifier Cross-Validation. We evaluate our classifiers using a round-robin cross-validation process where we leave one drawing out, train on the remaining drawings, and then test on the ground truth labels in the left-out drawing. Under cross-validation, we achieve an accuracy of 99%, a precision of 97%, and a recall of

96%. As expected, visual analysis of the few failure cases points to global cues discussed above as the main reason for failure.

Perceptual Validation. While assessing artist intent requires direct access to the artist, sketch processing literature [45, 94, 109] strongly suggests that artist intent is well correlated with viewer perception. We thus focus our evaluations on compariosn against viewer expectations. In addition to the ground truth labels used for training the classifiers, we collected manual annotations of 91 potential end-to-end and T-junctions across 10 drawings from the test set, with each potential junction annotated by 8 non-expert study participants. Across all junctions, participants agreed with the majority response 94% of the time, and were evenly split on 1 junction. The *final* classification decisions made by our algorithm agree with the majority response 92% of the time, nearly identical to the human agreement level—the most we can expect from an algorithm. This agreement number suggests that for a typical drawing with approximately 100 dangling tips after running our method users are unlikely to require more than 2-3 corrections to obtain an output consistent with their expectations.

Comparisons Against Prior Art We compare our method against prior interactive and automatic methods. When comparing against the former, we seek to assess the time it takes a user to generate a desired drawing connectivity using ours versus alternative approached. We focus this comparison on the LazyBrush [101] method, as it has been implemented in a popular professional software package [62]. On a representative input (Fig. 4.13a) it took an artist 31 minutes to achieve the desired output (Fig. 4.13c). To achieve this result, they used 70 scribble of different widths (including one erased in the process), Fig. 4.13b. Starting from our automatically generated output (Fig. 4.13c) the user required 2 minutes to generate the same output, using 7 corrections.

We also compare our method to five state of the art automatic gap closure methods, whose code we were able to access [35, 38, 82, 93, 97]. As discussed in Sec. 2.5 these methods detect closed cycles in raster data. To compare against these methods we rasterize our inputs and run them on the raster data, We use

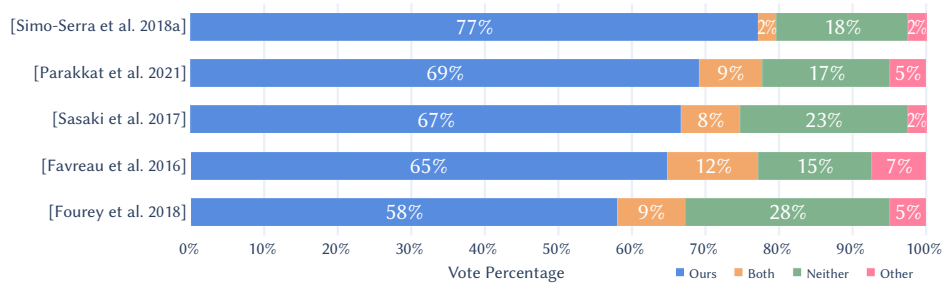


Figure 4.12: Study summary: participants preferred our method over all alternatives by a factor of 9 to 1 or more.

Figure 4.13: Comparison against interactive region detection. Given an input (a), the interactive LazyBrush tool [101] required 31 minutes (70 strokes, one erased) (b); starting from our automatically computed output (c) users required 2 minutes (7 corrections) to obtain the same final output (d). Input image ©The "Hero" artist Team under CC BY 4.0.

the output colorizations of [35, 38, 82]. We colorize the outputs of [93, 97] using flood-fill. We render the vector strokes on top of both colorizations and use a consistent raster resolution for all drawings (600 px for shorter image side); based on our experiments, this resolution produces the best result on average across these methods.

Fig. 4.15 and Fig. 4.16 compare our results against those generated by these methods. To compare the perceptual accuracy of our method against these prior approaches we conducted a comparative study (Appendix B.2). Participants were shown an input line drawing, and colorizations of this drawing obtained using our method and an alternative method. They were asked to "envision which strokes in

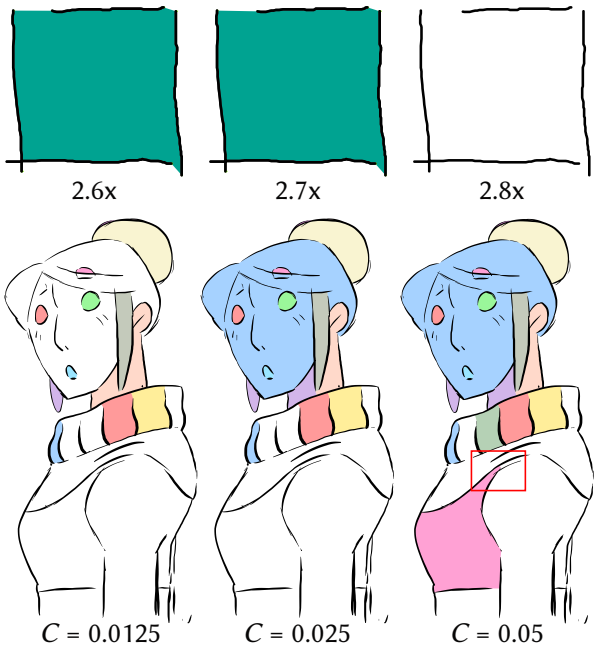
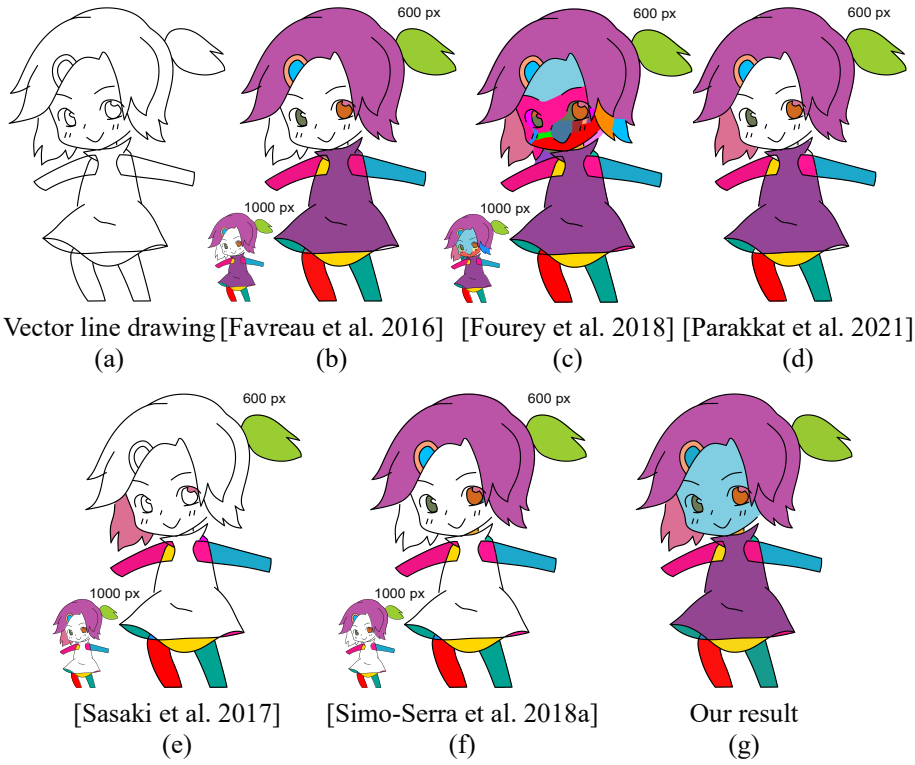


Figure 4.14: Impact of increasing the top left gap size (top) and the closure factor C (bottom) during our final, global closure-aware classification step. Top input image ©Company et al. [23]. Bottom input image ©The "Hero" artist Team under CC BY 4.0.

[the input] drawings are intended by the artist to form closed loops," to "Identify the differences between the two [shown] colorings (ignore small color bleedings)," and then answer "Which of the images on the bottom. (B) or (C), better corresponds to the partition you envisioned?" Overall we had collected answers to 135 comparative questions, 6 answers per question. The study findings are summarized in Fig. 4.12. Participant debriefings suggest that viewers looked for both under- and over-segmentations when evaluating alternative colorizations. When both colorizations were imperfect, they preferred the colorization with fewer or less visually disruptive errors. In a comparison with the best alternative [35] our method was preferred 65% of the time, and judged equally good 12% of the time; the method of Favreau et al. [35] was preferred just 8% of the time, and neither result was judged as corresponding to the participant envisioned one 15% of the time.



Vector line drawing (a) | [Favreau et al. 2016] (b) | [Fourey et al. 2018] (c) | [Parakkat et al. 2021] (d)

[Sasaki et al. 2017] (e) | [Simo-Serra et al. 2018a] (f) | Our result (g)

Figure 4.15: Rasterizing vector sketches and then applying the methods of [35] (b), [38] (c), [82] (d), [93] (e), and [97] (f) to compute closed stroke loops produces subpar outputs with both unintended junctions (e.g. Fourey et al. [38] over-segments character's face) and unresolved dangling endpoints (e.g. none of [35, 82, 93, 97] separates character's face from the background). Our outputs (g) correctly identify both intended junctions and intended dangling endpoints. We show both high and low resolutions (600 px and 1000 px) for (b, c, e, f); and the authors' automatically selected resolution (600 px) for (d). Light gray spots in the output of [82] correspond to pixels unassigned by their method. Input image ©Jiang et al. [53].

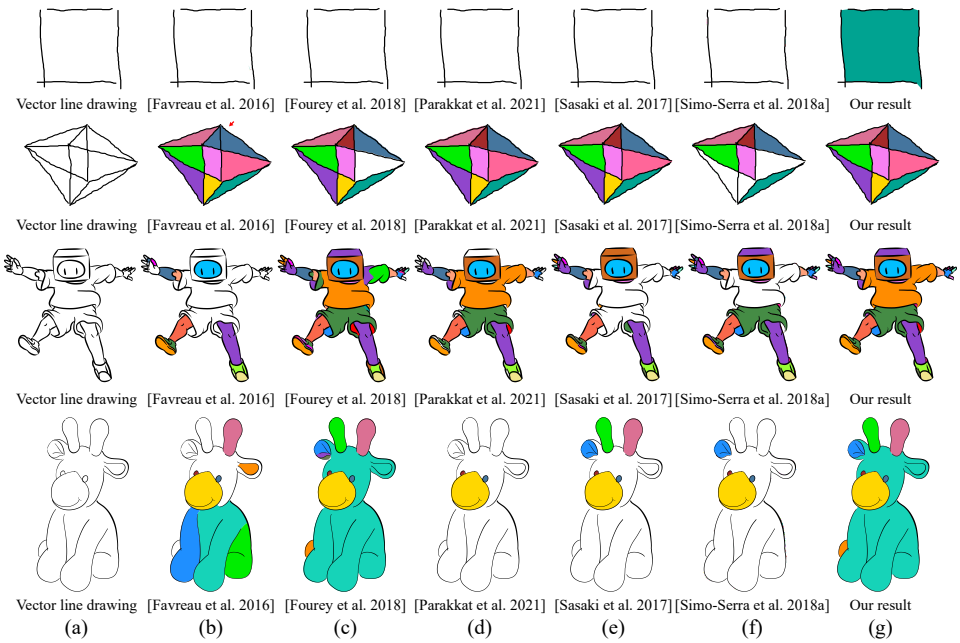


Figure 4.16: Additional results and comparisons. Input images from top to bottom ©Company et al. [23]; ©Lien-ze Tsao under CC BY 4.0; ©Enrique Rosales.

The demonstrated preferences were highly statistically significant ($p < 0.001$ for all methods). These numbers convincingly demonstrate our significant improvement over the state of the art, in the context of detecting intended junctions in vector drawings.

Closure Ablation. We conducted a geometry variation and a parameter variation ablation experiment on the closure cue. Fig. 4.14 (top) demonstrates how the closing of gaps is robust to the gap size. For this specific input, the closure step continues to connect the gap until the distance becomes 2.8 times larger than the original. Fig. 4.14 (bottom) shows the impact of changing the value of the closure factor C during our final, global closure-aware classification step (Sec. 4.5). If C is too low, the gap ratio R_C does not have sufficient influence in this stage, and major regions such as the face are not captured. If C is too high, however, the global step may yield undesirable false positives, such as the front of the tunic in the example figure.

This both validates the importance of incorporating the gap ratio during this final stage, and our choice of closure factor. An interesting area for future work could be to learn an adaptive closure factor based on local or global drawing properties.

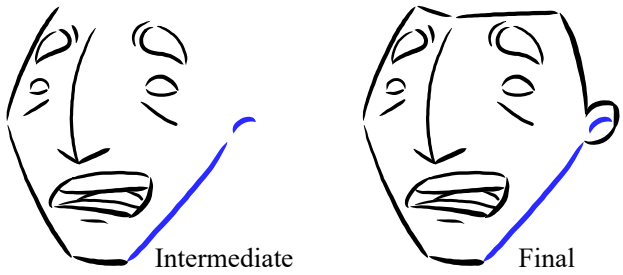


Figure 4.17: When presented with an incomplete drawing both our method and human observers could perceive some intended gaps as unintended. Source: ©The "Hero" artist Team under CC BY 4.0.

Incremental Processing. As an alternative to our approach, we also experimented with an incremental workflow based on drawing order. Unfortunately, when presented with an incomplete drawing (Figure 4.17) both our method and human observers perceive some intended gaps as unintended; automatically closing such intended gaps mid-drawing is highly disruptive to the artist. Processing complete drawings provides our method with more complete decision-making context, leading to outputs better reflecting artist intent.

Junction Statistics Across the 95 inputs in our test set, our method forms 1584 junctions in total: 638 (40.3%) junctions are binary end-end junctions, 855 (54.0%) junctions are binary T-junctions, and only 91 (5.7%) junctions are high-valence junctions. We note that the sparsity of high-valence junctions (fewer than 1 per input on average) validates our design choice to use the pairwise junction classifiers to predict the likelihood of higher valence junctions.

Junction Formation Counts per Step. Across the 95 inputs in our test set, 1452 (91.7%) junctions are formed during our primary step, 58 (3.7%) are formed during

our secondary step, and 74 (4.7%) are formed during our closure-aware step. This confirms that our primary step forms most of the junctions, whereas the secondary and closure-aware steps form fewer, yet visually critical, junctions (as demonstrated in Fig. 4.1 and Fig. 4.5.)

Limitations While as demonstrated above our method significantly outperforms all state-of-the-art alternatives, a non-negligible number of comparative study participants selected the "neither" option when faced with our and alternative inputs. This suggests that while we significantly advance the state of the art, additional effort is necessary to detect intended junctions in free-hand drawings fully automatically. Furthermore, while our pre-processing is capable of detecting and consolidating sketches which contain some amount of overdrawing, our core method is designed to operate on inputs with no or minimal overdrawing. Using our method on sketchy inputs with large amount of overdrawing or hatching requires a more robust consolidation pre-process; while the methods reviewed in Sec. 2.5 for this task can often be used for such pre-processing, robust consolidation remains an open research problem.

Chapter 5

StripMaker: Perception-driven Learned Vector Sketch Consolidation

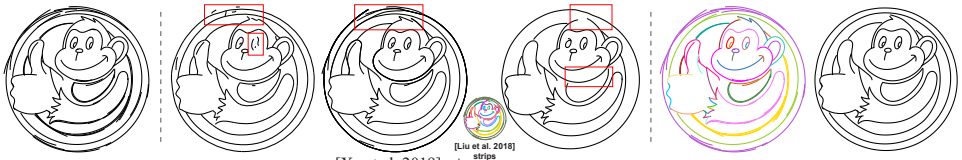


Figure 5.1: Given a vector sketch with multiple overdrawn strokes (a) StripMaker automatically consolidates it (f) replacing each detected viewer perceived strip of strokes (e, each strip in different color) with the corresponding intended curve. StripMaker outputs (e) are better aligned with user expectations than those produced by state-of-the-art algorithmic alternatives (b,c,d). Inset in (d) shows strips generated by Chapter 3. Frames point to artifacts in outputs of previous methods. Source: Yan et al. [112].

StrokeAggregator, the consolidation method we present in Chapter 3, generates clean line drawings based on local geometric cues. These local geometric cues are far from the complete set of perceptual cues involved in sketch consolidation. This makes us wonder if additional cues that are contextual and temporal would bring improvement and also support a new framework that can be easily adapted to an interactive workflow. In this chapter, we propose StripMaker, a new and robust learning based method for automatic consolidation of raw vector sketches. While the strategy of conducting an individual study per cue in Chapter 3 is shown to be promising, this strategy is hardly scalable for contextual and temporal cues. Inspired by our research on connectivity that faces the similar challenge in Chapter 4, we avoid the need for an unsustainably large manually annotated learning corpus by utilizing observations about artist workflow and perceptual cues viewers employ when mentally consolidating sketches.

5.1 Introduction

When presented with raw overdrawn sketches, human observers effortlessly imagine the artist-intended stroke groups and their corresponding curves. However, annotating or retracing sketches to produce these viewer imagined consolidated outputs is highly time-consuming [67]. While a range of attempts have been made to automate consolidation of both vector [67, 69] and raster [100, 111] sketches (Section 2.4), existing consolidation algorithms frequently fail to produce viewer expected results (Figure 5.1b-d). We propose a new vector sketch consolidation method that produces outputs significantly better aligned with viewer expectations than those produced by these alternatives. We focus on vector inputs since, as noted in earlier chapters, such sketches and the interfaces used to create them are increasingly ubiquitous, and the additional information they contain can be potentially used to simplify the consolidation task. Vector sketch consolidation can be thought of as a combination of two tasks: *clustering* strokes into groups that jointly depict intended curves (Figure 5.1e) and fitting the best corresponding curve to each such group (Figure 5.1f). We focus on the clustering task, and use the state-of-the-art method of Pagurek van Mossel et al. [79] for the latter. Following Van Mossel et al., we refer to stroke groups that depict intended curves as *strips* and consequently refer to our method as *StripMaker*.

Prior research and our observations (Section 5.2) suggest that viewers decide which strokes belong to the same strip based on the spatial relations between these strokes, the local context surrounding these strokes, and the global properties of the viewed sketches (Figure 5.2). In particular, when evaluating whether groups of strokes form strips, observers mentally establish dense correspondences between the side-by-side portions of these strokes [67, 79] and use these correspondences to assess the compatibility between them (Figure 5.2a-e). Our analysis suggests that viewers are impacted by the presence of actual or viewer-perceived intersections between the assessed and neighboring strokes (Figure 5.2f). Lastly, we speculate that viewers are more likely to combine farther apart strokes on drawings that appear less accurate overall, and to be less aggressive given drawings which appear more neat (Figure 5.2g). Still, it remains unknown how viewers measure or balance the different factors, or cues, involved.

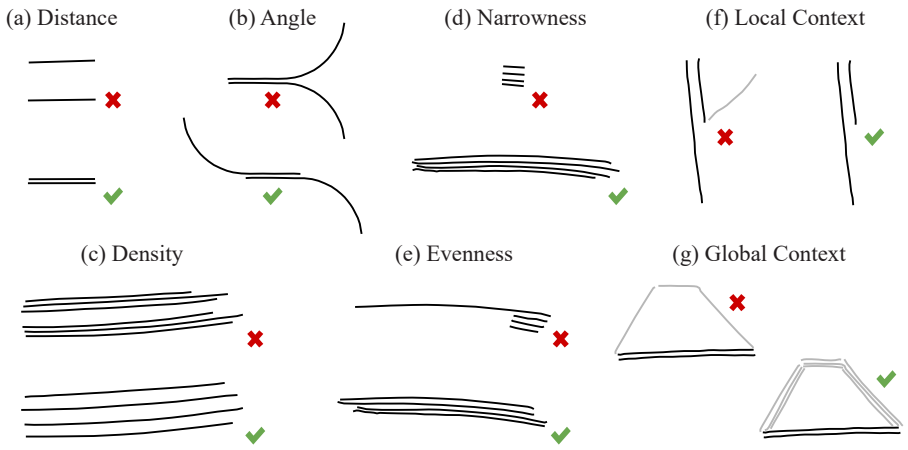


Figure 5.2: Consolidation cues: Locally groups of strokes are seen as belonging to the same strip if they are proximate (a), roughly parallel (b) , and approximately evenly spaced (c). Strips are expected to be narrow (d) and have roughly even width throughout (e). Local (f) and global (g) context impacts strip perception.

A potential approach for addressing perception motivated tasks with similar unknowns is to learn the viewer desired outcomes from manually annotated data [115]. Learning to consolidate sketches requires addressing several challenges. Even with a user-friendly UI, strip annotation takes between 10 and 30 minutes for small- to medium-complexity sketches (see Appendix B.3.1) This makes collecting thousands or even hundreds of annotated training examples impractical. At the same time, the need to account for global and contextual factors impacting human consolidation choices strongly suggests that brute-force learning of viewer preferences is only possible using large datasets which span a diverse spectrum of local, contextual, and global factor combinations. We address this challenge by leveraging a number of observations that allow us to break our clustering problem into sub-problems, the answers to which can be learned using limited amounts of training data. We first note that while correctly clustering strokes often requires global context, many clustering decisions can be made using purely local information. In other words, we can often correctly classify groups of strokes as belonging to the same strip without considering the properties of any other strokes in the drawing. We also observe that given an approximately consolidated sketch, we can compactly encode the global context required for making even more accurate

local consolidation choices. Following this observation we use a two step consolidation process: we first use purely local properties to obtain an approximate, or *preliminary* consolidation; we then refine this preliminary outcome by combining local cues with contextual and global features computed using preliminary strips (Section 5.3).

Both stages of our algorithm require a way to robustly and efficiently cluster strokes into strips using relevant geometric features. Even when focusing on local or compactly encoded global features, learning N-way clustering where N can vary is likely to require large amounts of training data. We dramatically reduce the amount of training data necessary by focusing on binary classification: given two groups of strokes, we train our classifiers to determine if the union of the two forms a common strip (Section 5.4). Since typical sketches contain dozens of strips, and exponentially more *sub-strips* (groups of strokes which are part of a strip), such classifiers can be successfully trained using a relatively small set of diverse sketches (our classifiers were trained on 66 annotated sketches). Robustly assessing if a pair of sub-strips belongs together requires capturing the different features that impact human clustering decisions, and thus requires establishing dense correspondences between the side-by-side portions of the sub-strips; computing such correspondences algorithmically is far from instant [67, 79]. We therefore require a principled way to keep the number of such correspondence computations and the classifier calls that trigger them small without sacrificing output accuracy.

Bottom-up strategies that first apply a classifier to all pairs of individual strokes in a sketch, and then repeatedly apply it to all pairs consisting of newly formed and other sub-strips, are unsuitable for our needs as they are likely to require a prohibitive number of classifier calls. We obtain our preliminary consolidations while keeping down the number of calls by observing that during the drawing process artists often, though not always, draw strokes belonging to the same strip temporally close to one another. Following this observation, we employ an incremental pairwise sub-strip evaluation order that leverages this workflow and allows us to dramatically limit the number of classifier calls (Section 5.3.1).

We refine the resulting preliminary consolidation by re-evaluating the clustering decisions within each preliminary strip and in-between adjacent preliminary strips using our second classifier which uses both local and contextual features and

is trained on the same compact set of annotated drawings (Section 5.3.2). In our cross-validation experiments, our refinement step improves consolidation accuracy, measured as distance between algorithmically and manually consolidated sketches by 20%.

We validate our method via a range of quantitative and qualitative comparisons to prior art and manual consolidation (Section 5.6). Our comparative study participants preferred our results over the closest alternative 67% of the time, judged them as on par 19% of the time, and preferred the alternative only 14% of the time. Our evaluations demonstrate that StripMaker significantly outperforms the state of the art in terms of alignment with user needs.

5.2 Analysis of Overdrawn Sketches

Professional and amateur artists often depict intended curves using strips of overdrawn strokes [2, 32, 112]. They use overdrawing to correct or refine earlier strokes, emphasize specific curves, and break down hard to draw long and complex curves into shorter, easier to sketch strokes. Observers easily overcome such inaccuracies and correctly interpret artist intent. To match this intent when forming strips, we therefore consider both artistic practices and research on human perception. While the exact mechanism viewers employ to parse sketches remains unknown, based on prior work and our own observations we speculate that viewers employ the following cues when consolidating sketches (Figure 5.2). We leverage those cues in our algorithm (Section 5.3).

Correspondence. At its core, consolidation merges together groups of strokes that are fully or partially *side-by-side*, or next to one another (Figure 5.3). Research suggests that when making consolidation decisions, viewers rely on implicit *dense correspondence* (orange isolines in inset) between side-by-side stroke sections when evaluating the degree of compatibility between them [67], and expect each strip to allow for a *low distortion 1D parameterization* and be well approximated by a single curve [79].

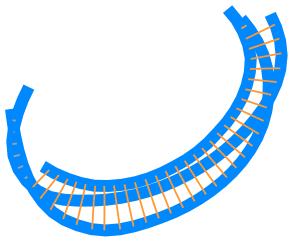


Figure 5.3: Correspondense between strokes are defined by 1D parameterization. The isolines of this 1D parameterization are shown in gray and orange. The orange isolines indicate the side-by-side section between the top stroke and the remaining ones.

Local Geometry. Application of Gestalt psychology grouping principles [61, 104] to strokes suggests that *parallelism*, *distances*, and *density* play a major role in consolidation decisions (Figure 5.2a-c). Viewers are more likely to group strokes which are more parallel and closer to one another along their side-by-side sections. Density suggests that viewers are more inclined to see strokes as forming a strip if the distances between adjacent strokes are more even, in particular this suggests that wider sub-strips are more likely to be seen as belonging to the same strip than more narrow sub-strips spaced at the same distance, see Figure 5.2c. In addtion, Liu et al. [67] demonstrate that viewers expect strips to be *narrow*, having a small width to length ratio (Figure 5.2d). We further observe that viewer perceived strips typically have roughly the same, or *even*, width throughout with no drastic changes (Figure 5.2e).

Drawing Order. Our analysis of sketch drawing order confirms observations in prior literature [42, 76] that strokes belonging to the same intended strip are often drawn temporally close to one another, and are frequently drawn consecutively; in Figure 5.4 the coloring reflects drawing order—while few strips are drawn fully consecutively, large portions of many strips are.

More generally, we note that incomplete sketches, i.e., sketches visualized at any intermediate drawing time steps, share many properties with finished ones, see Figure 5.4b which has the first half of the strokes of the cupcake above it. In particular, strokes perceived as belonging to the same strip in an incomplete

sketch are highly likely to be perceived as such in the finished one. The same holds to a weaker degree in the inverse direction—strokes perceived as being apart in an incomplete sketch more often than not continue to be seen as belonging to different strips in the final sketches. We refer to this property as *temporal persistence*.

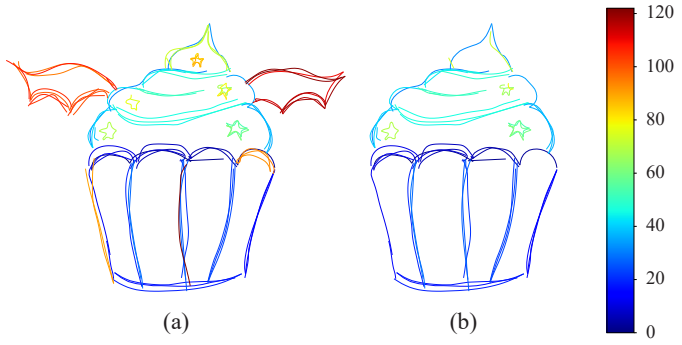


Figure 5.4: Strokes belonging to the same intended strip are often drawn temporally close to one another as indicated by coloring (a). Incomplete sketches share many properties with finished ones and strokes perceived as belonging to the same strip in an incomplete sketch are highly likely to be perceived as such in the finished one (b). Source: Yan et al. [112].

Global and Local Context. Our analysis suggests that viewers' consolidation decisions are impacted by the overall sketch *precision*. Viewers are more likely to group widely spaced strokes together on rough messy drawings, where all strips have more spaced out strokes. In contrast, on cleaner drawings, views are likely to see adjacent side-by-side strokes as separate intentional details rather than a byproduct of sketchy overdrawing (Figure 5.5a left vs right). In particular, viewers are likely to incorporate stand-alone, *outlier*, strokes (red in the top Figure 5.5a) into one of their adjacent strips when their surrounding clusters are less precise, [67].

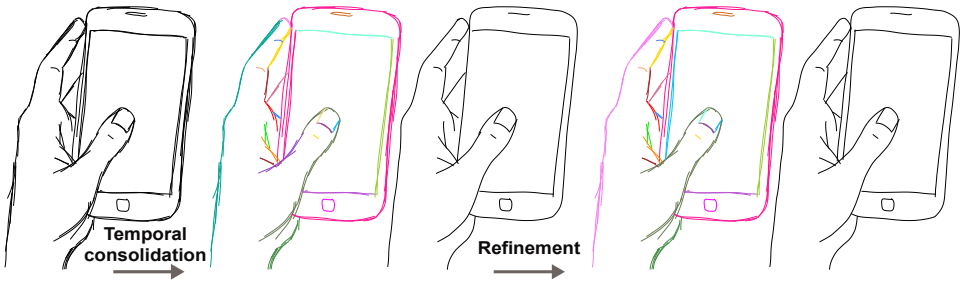
Lastly and importantly, consolidation decisions are impacted by perception of inter-strip junctions. Specifically, viewers expect connectivity to be non-accidental, and are less likely to mentally consolidate strokes when doing so reduces the number of perceived inter-strip junctions. In the Figure 5.5b, the two highlighted groups of strokes look likely to be in the same strip in isolation (left), but are viewed as

(a) (b)

Figure 5.5: Viewers' consolidation decisions are impacted by global and local context: the overall sketch precision (a) and the perception of inter-strip junctions (b).

apart when the gray strip is present (right). We refer to this property as *connectivity preservation*.

5.3 Algorithm



(a) (b) (c)

Figure 5.6: StripMaker first generates preliminary consolidations (b) of the input sketches (a) and then refines those using global cues to arrive at the desired output (c). Note that the over-merged left side of the screen (b) is corrected separated through refinement (c). Source: Pagurek van Mossel et al. [79].

Input. The input to our method is a vector format sketch, where strokes are fixed-width curves and sorted based on their drawing order. We pre-process each curve into an evenly and densely sampled polyline, remove end-point hook artifacts resulting from inadequate device capture [115], and break strokes at sharp corners, enabling processing of cases where users use zigzag overdrawing patterns; see Appendix A.3 for details.

Workflow. We first consolidate the inputs using a method based on an analysis of local geometric feature (Section 5.3.1, Figure 5.6b). While not perfectly accurate, the approximate consolidations it computes are close enough to the viewer expected output, enabling us to estimate sketch precision and likely strip connectivity. We use these estimates in our refinement pass generating the final outputs (Section 5.3.2,Figure 5.6c). We discuss the design and training of the classifiers used in both stages in Section 5.4 and provide implementation details in Section 5.5.

Output. We fit an aggregate curve to each output strip using the method of [79]. Prior to the fitting, we identify strips that form continuation end-end junctions using a simplified version of the method of [115]. We merge strips connected via continuation junctions and fit them jointly. Similar to Liu et al. [67] we delete single stroke strips which almost completely overlap multi-stroke ones, as these single stroke strips are perceived as noise.

5.3.1 Local Temporal Consolidation

Our main consolidation step uses a local-feature-based classifier to group the input strokes into preliminary strips (Figure 5.6ab). Computing the geometric features necessary to evaluate whether two sub-strips belong to the same strip requires computing a correspondence between them, a computationally non-trivial task. We thus require a consolidation workflow that keeps the number of classifier calls and corresponding feature computations small.

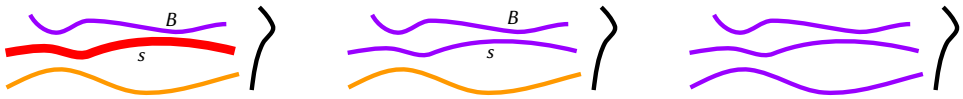


Figure 5.7: A new stroke (red) is considered against existing strips (a) and added to the strip (purple) with highest probability that is larger than 0.5 (b). The newly added stroke makes the updated strip more likely to be combined with another existing strip (orange) and thus we iteratively combine strips (c).

We limit the number of classifier calls by leveraging temporal persistence and drawing order. Given the time-ordered sequence of sketch strokes as the input, our

algorithm processes one stroke at a time, see Figure 5.7. Given the new stroke s (red in the Figure 5.7), we measure the likelihood that the stroke belongs to one of the previously formed sub-strips using our local classifier (Section 5.4). If the classifier indicates that the stroke may belong to one or more of these sub-strips, we add it to the sub-strip B with the highest classifier likelihood that is larger than 0.5 (purple in the Figure 5.7); otherwise, we store the stroke as a separate sub-strip. In case a stroke is added to a sub-strip, the classification process is iterated, this time assessing if this new sub-strip should be combined with one of the previously formed sub-strips. If yes, the new sub-strip is combined with the sub-strip with the highest classifier likelihood that is larger than 0.5 (Figure 5.7c). We repeat the process until there are no sub-strips to combine.

Speed-up Naively testing new strokes or sub-strips against all previously formed sub-strips at each iteration would imply performing numerous classifier calls, the vast majority of which are likely to provide a negative answer. To gain necessary performance, as a pre-filter to the classifier, we first evaluate the *compatibility* of the assessed sub-strips, and only call the classifier if they are deemed compatible, i.e., , have non-zero likelihood of belonging to a common strip. We consider sub-strips compatible if three conditions are satisfied: (1) they are at least proximate, (2) they are weakly parallel, (3) their joint parameterization has sufficiently low distortion, and (4) their side-by-side sections are sufficiently long. More specifically, if they fail any of the conditions in this order below, the sub-strips are highly unlikely to belong to the same strip and there is no need to call the classifier to evaluate them.

(1) We check if the shortest pointwise distance between the two sub-strips is below 10 times stroke thickness. If not, the strips are not compatible. If yes, we proceed with additional evaluations using the already computed fitting curves of each sub-strip. Given these two fitting curves, we compute the shared parameterization of them for the faster pre-filter checks (note that such parameterization is much faster to compute than a parameterization of all strokes in the sub-strips which is required to compute the classifier features). (2) We compute the angle difference between the combined fit curves along the side-by-side sections and check if the average angle is below 35°. (3) We check if the joint parameterization

has excessive distortion (the maximum magnitude of the alignment term [79] in Equation 5.2 is greater than 2). (4) We check that in the common parameterization the side-by-side section of the fitted curves is longer than six times the stroke width and is at least 20% of the length of the shorter sub-strip. A pair that fails one of these tests is deemed incompatible. All threshold values above were determined based on training data set statistics.

Similarly, comparing each newly formed sub-strip against all other sub-strips necessitates many classifier calls, the vast majority of which return a negative answer. We speed up the computation by observing that a stroke is more likely to belong to the same strip as its temporally previous stroke, than to belong to a different multi-stroke strip.

As described in Algorithm 2, we at each iteration keep track of the sub-strip R that the most recently processed stroke belongs to. After locating the strip B that a new stroke s is deemed to belong to, if $B = R$ we add s to B, delay any new sub-strip comparisons, and proceed to the next stroke in the temporal order. If B is different from R or if s is not added to any existing strip, and R has more than one stroke, we assess if R can be merged with other strips (TryMerging(R, C) runs a fixed number of passes, 3 in our implementation): we use our local classifier to evaluate the sub-strip R against all previously formed sub-strips, merging it with an existing sub-strip if the classifier deems the two to belong to the same strip. We repeat the iterative evaluation when a merger occurs. Our delayed evaluation reduces the number of classifier calls by an order of magnitude: the comparisons between two multi-stroke strips are reduced to 56% on average. This effect is more evident on inputs with thick strips: the most extreme case in our validation set is the car (Figure 2.4) with the comparisons reduced to 13%.

5.3.2 Refinement

After the temporal pass is complete, we expect the vast majority of the resulting strips to match viewer expectations (in our cross validation experiments, Section 5.6, the consolidations produced at this stage were 97% consistent with the ground truth labels). We thus only revisit clustering decisions locally where the temporal pass results are most likely to need refinement. In doing so, we seek to

ALGORITHM 2: Local Temporal Consolidation

Data: A sequence of strokes $S = \{s_i\}, i = 1, \ldots, N$
Result: A set of strips $C = \{C_j\}$, s.t. $\cup C = S$
R $\leftarrow \emptyset$;
$C \leftarrow \emptyset$;
for $i = 1, \ldots, N$ **do**
    $C' \leftarrow$ all strips in C that are compatible with s_i;
    $p, D = \max_{c \in C'} \text{classify}(c \cup \{s_i\})$;
    if $p \geq 0.5$ **then**
        $B \leftarrow D \cup \{s_i\}$
    else
        $B \leftarrow \{s_i\}$
    end
    if $B \neq R$ **then** `// The stroke belongs to a different strip`
        R $\leftarrow$ TryMerging(R, C);
        $C = C \cup$ R;
        R = B;
    end
end
R $\leftarrow$ TryMerging(R,C) `// Final merging attempt`
$C = C \cup$ R

balance the global and local classifier choices. On the one side, our global classifier and the algorithm around it are able to leverage contextual information that is not available during our temporal pass. On the other side, our global classifier relies on features computed using complete sketches, and is thus more sensitive to the fact that our training corpus is by necessity not large and thus may not have the necessary overall drawing style diversity to fully generalize. We thus change preliminary consolidation decisions conservatively, and only use our global context aware classifier to re-evaluate and split existing strips, when necessary, and to merge adjacent strips that clearly warrant merging.

Strip Re-evaluation. We re-evaluate each multi-stroke strip taking context into account (see Figure 5.8). For each strip, we select two seed strokes by finding the stroke pair least likely to belong together using our global classifier (Section 5.4). If this likelihood is sufficiently high (> 0.6), the strip is left as is. Otherwise, if multiple pairs have the same likelihood of being together, we select the pair with the largest average stroke-wise distance as the seeds. We mark all non-seed

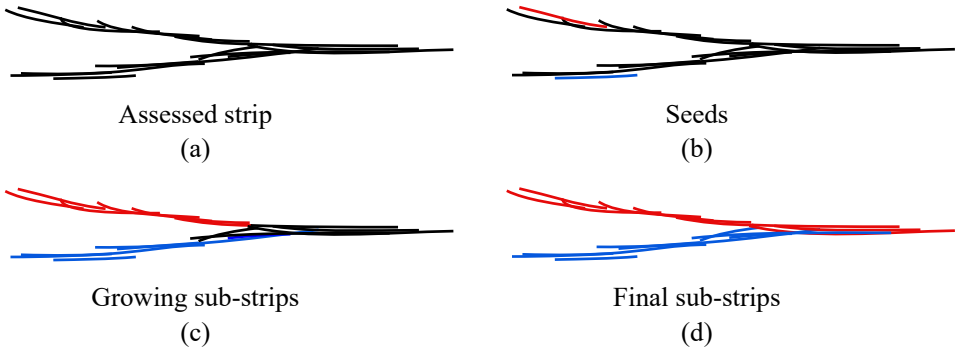


Figure 5.8: Multi-stroke strip are split and re-evaluated (a). A pair of strokes with the lowest probability are selected as seeds (b). The sub-strips are grown from these seeds (c) until all strokes are assigned (d).

strokes as unassigned, and grow sub-strips incrementally from the two seeds, by adding unassigned strokes to one of the sub-strips, and stopping when all strokes are assigned. At each iteration, we assess for all unassigned strokes the likelihood of them belonging to one of the seed sub-strips. If more than one stroke has a likelihood of 0.5 or higher, we prioritize strokes that are side-by-side to both seed strips. Among those we select the ones with the highest likelihood value, and break ties by prioritizing strokes that are closest to the corresponding seed strip. If no unassigned strokes are deemed to belong to a seed sub-strip, we classify one of the unassigned strokes as a new seed, and continue.

Once all strokes are assigned to sub-strips, we re-evaluate if any pair of sub-strips belongs to the same strip. We merge pairs back into common strips if they are deemed to likely belong together by our global classifier, and if doing so does not change the viewer-perceived sketch connectivity as discussed below. Specifically, subject to the connectivity assessment below, we merge multi-stroke sub-strips together if the likelihood is above 0.5 and merge single strokes with other sub-strips if the likelihood is above $T = 0.3$ (the conservative threshold is motivated by the observation that human decisions on such pairs are less affected by context).

Strip Re-evaluation Speed-Up To speed up our strip reevaluation step, during the entire strip refinement process, we use the parameterization of the preliminary

strips to compute the local features used by the classifier. Only if at the end of the re-evaluation the strip is deemed in need of a split, we reparameterize the sub-strips and re-evaluate the split decision.

Connectivity Preservation. The connectivity preservation property suggests that if a sub-strip is perceived to form a junction with another strip at one of its end-points, it is more likely to be perceived as being a *stand-alone* strip. We detect perceived junctions at the end-points of the assessed sub-strips by fitting them with the corresponding intended curves and use the classifier in Yin et al. [115] determining the likelihood of two curves forming a junction. If a junction is detected, we do not merge the assessed sub-strips if the global classifier likelihood is below $1-T$ and the distance from the junction to the other assessed sub-strip is high (1.5 sub-strip width).

Strip Merging We merge adjacent strips in the temporal pass output if they are deemed to be part of the same strip by our global classifier and if they pass the connectivity preservation test above. Specifically following the conservative logic above, we merge multi-stroke sub-strips together if the likelihood is above 0.55, merge single strokes with mutli-stroke sub-strips if the likelihood is above 0.5, and merge singe strokes if the probability is above $1-T$.

5.4 Classifier Design

At the core of our iterative consolidation pipeline lie two binary Random Forest [50] classifiers, responsible for predicting the probability that two given sub-strips belong to the same or different viewer-perceived strips. The classifiers output a number $c \in [0,1]$; if $c \geq 0.5$, the pair is more likely to belong together than apart. Random Forests had been shown to be well suited for the type of problems we address [43, 115].

Our local classifier, used in our temporal consolidation step, operates on features that can be computed purely on the evaluated sub-strips, and is trained on sub-strips similar to the ones encountered during temporal consolidation. Our global classifier uses the same set of features with the addition of a *relative precision*

feature that encodes the precision of the assessed pair of sub-strips relative to the precision of the rest of the sketch, and is trained on sub-strips similar to the ones encountered during the refinement step. For additional details of the composition of the classifier training corpuses see Appendix B.3.1.

Our classifier features are inspired by the analysis of cues observers employ when making consolidation decisions (Section 5.2, Figure 5.2). Computing robust features to capture these cues requires point-to-point correspondences between the assessed sub-strips (see Figure 5.9). We obtain these correspondences via StrokeStrip parameterization [79]. The parameterization isolines provide a reliable and intuitive pointwise correspondence across all sub-strip strokes and the length of each isoline provides an estimate of the strip widths. We define the parameter span shared between the two sub-strips as the common *side-by-side section*. We measure all pairwise geometric features over isolines in that interval only (orange in Figure 5.9). We normalize all computed distance by the stroke thickness.

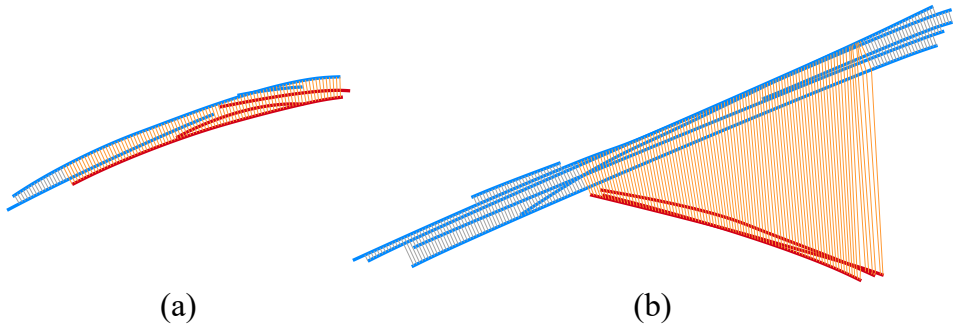


Figure 5.9: Parameterization [79] computed on given sub-strips (in blue and red). Correspondences are indicated by isolines in gray and orange; features are measured within the side-by-side section highlighted in orange. Positive (a) and negative (b) sub-strip pairs have visual difference which we try to capture via geometric features.

Angles and Distances. We measure the angles between tangents at corresponding points, over all shared isolines and measure distances between the closest corresponding points on the two sub-strips, purple in Figure 5.10 (for intertwined sub-strips this distance is defined as zero). For each of these measurements, we compute averages and medians over all the relevant isolines (4 features overall).

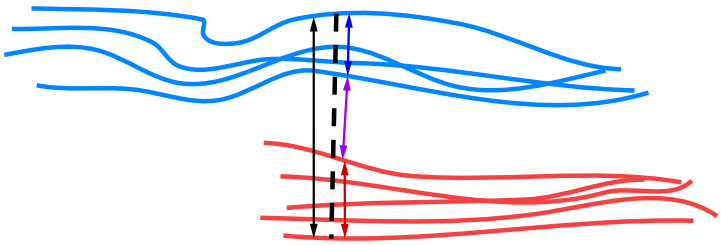


Figure 5.10: Distances are measured within each side-by-side isoline and aggregated along the side-by-side section. Within an example isoline illustrated as a dash line, the distance between sub-strips is shown in purple; the width of the wider sub-strip in red; the width of the narrower sub-strip in blue; the width of the combined strip in black.

Density. We encode density by measuring the distances between closest points on different sub-strips (purple) and normalizing those by the widths of the wider (red) and narrower (blue) sub-strips, and the width of the entire isoline (black). For each of these measurements, we compute averages and medians over all the relevant isolines. We also measure the ratios between the 90^{th} and 10^{th} percentile within strip distances and the inter-strip distance (10 features overall).

Narrowness and Side-by-Side Extent. We capture how narrow a strip is as the ratio of its length, measured as its full parameterization span, to its average and median widths. We measure the minimum and maximum of the two ratios computed for each sub-strip, as well as the ratios for the combined strip (6 features). We measure the side-by-side extent of the two sub-strips as the ratio of the length of the shared parameter span to the full parameter span.

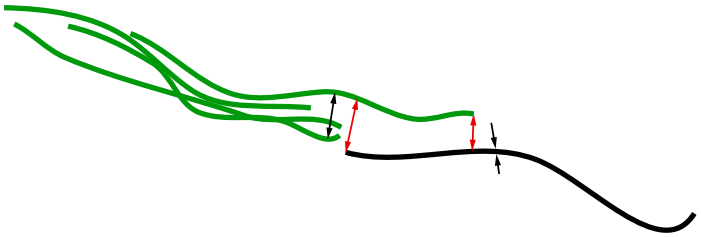


Figure 5.11: Evenness is measured based on the isolines at the ends of side-by-side section (red) and the isolines immediately outside the side-by-side section (black). Large difference in length between red and black isolines indicates unevenness.

Evenness. We define the evenness of the combined strip as the difference in width between the side-by-side segment of the combined strip and the widths outside this segment. At both ends of the side-by-side segment (red in the Figure 5.11), we compute the ratio of the width of the first isoline inside the segment to that of the outside isoline just next to it, red to black distance ratio in Figure 5.11 (we use 1 as the value if no outside isoline exists). We also compute the ratio between the average width of the side-by-side section and the width of the sub-strips inside the parameter intervals before and after it (we use 1 as the value if there is no such interval); we store both sets of values ordered as maximum and minimum.

1D Parameterization Distortion. Since viewer perceived strips are expected to allow for a low-distortion parameterization [79], the quality of this parameterization is in itself an indirect indicator of whether a group can be interpreted as a strip or not. We measure the distortion using the energy terms in the original StrokeStrip formulation [79].

We compute two values for a given potential strip: the maximum deviation of the tangent length (velocity) from 1, and the maximum misalignment [79]. Here $u(x)$ is a parameterization, defined for every strip point, $C(t)$ is the isoline for the parameter value t, $\tau(t)$ is an average tangent over the isoline, and $n(t)$ is the average normal.

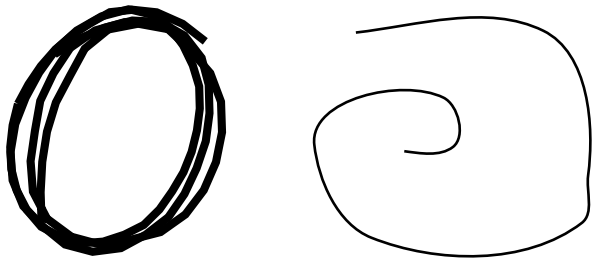
$$E_{\text{length}} = \int_0^L \left| \frac{1}{W(t)} \int_{C(t)} \nabla u(x) \cdot \tau(t) dx - 1 \right|^2 dt \tag{5.1}$$

$$E_{\text{align}} = \int_0^L \int_{C(t)} |\nabla u(x) \cdot n(t)|^2 dx dt \tag{5.2}$$

Relative Precision. Our global classifier combines the features above with a family of features which relate the distance between the assessed sub-strips to the stroke density across all other strips in the current consolidation. Specifically, we measure for all strips the average and median inter-stroke distances along all isolines, and record the median, average and 90^{th} percentile results across all strips. We similarly measure the median and average distance between the assessed sub-strips. We record all ratios between these values as features (6 features in total).

5.5 Algorithm Details

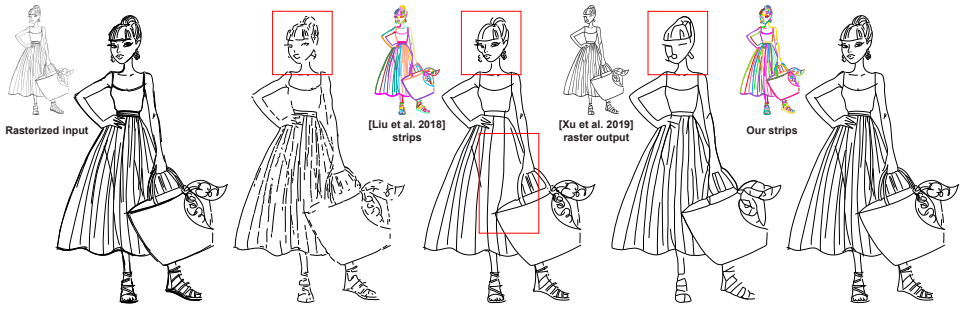
Postprocesing Our postprocessing detects continuations between strips and enforces those during fitting by merging the strips. We first detect actual or intended strip end-end intersections, and treat pairs of intersecting strips as continuations if the angle between the tangents at their endpoints is under 20° [12, 49] and the two local strip widths differ by less than 4 times. We detect highly-likely junctions (with probabilities $> 90\%$) as intended junctions using Yin et al. [115], and consider strips as forming actual end-end junctions if they intersect immediately next to their respective end-points (within 20% of the strip length and three times the strip width from the endpoints). As noted by Liu et al. [67], artists often do not delete extreme outlier strokes if these are essentially covered, or hidden, by other stroke strips. Similarly to Liu et al., we detect and delete such outliers. We define a single stroke strip as an outlier if more than 90% of its area is covered by the envelope of another strip extended by 50% its width. Lastly, we detect single stroke overdrawn ellipses and fit them as closed strips [79].



(a) (b)

Figure 5.12: Two examples of winding strokes. The overdrawn ellipse is supposed to be fit as a closed strip (a); while the intentional spiral is supposed to be left open (b).

We consider an input stroke as potentially an overdrawn ellipse if its total signed curvature magnitude, $|\kappa| > 2\pi$. In this case, we compute the substrokes, corresponding to the *loops* with $|\kappa| \leq \pi$. We distinguish between actual overdrawn



(a) Input sketch (b) [Stanko et al. 2020] (c) [Liu et al. 2018] (d) [Xu et al. 2019] → [Puhachov et al. 2021] (e) Our output

Figure 5.13: Consolidating typical inputs (a) using state-of-the-art methods for simultaneous consolidation and vectorization [100] (b), and vector [67] (c) and raster [111] (d) space consolidation, often results in both loss of details and under-consolidation. Rasterized input used for (b) and (d) shown as inset in (a). The raster output of [111] (shown in the inset in (d)) was vectorized using the method of [86]. Our method (e) produces viewer expected consolidations on these inputs. Please zoom-in to see details. Source: Yan et al. [112].

ellipses (Figure 5.12a) and intentional spirals (Figure 5.12b) using the following heuristic. We find the barycenter of each loop and set d_{mass} to the maximal Euclidean distance between those. We parameterize the substroke as a closed strip using the method of Pagurek van Mossel et al. [79]. We measure the maximal distance between adjacent points along parameterization isolines g and compute the strip radius $r = L/2\pi$ where L is the strip length. Given the stroke width w, we consider the stroke to be an ellipse if $g < 50w$ and one of the following holds $d_{\text{mass}}/r < 0.25$ or $d_{\text{mass}}/r < 0.45$ and $g < 3w$.

Random Forest Our classifiers have 150 trees with maximal tree depth capped at 20. Our average tree depth for the local classifier is 13.7, and the mean number of leaves is 129. We use the scikit-learn library [85] implementation of random forest.

5.6 Results and Validation

We tested our method on 191 previously unseen sketches, including 107 sketches from sketch processing benchmarks [44, 112] as well as 82 sketches we commissioned from 12 different artists. 16 of these are shown in the paper. These sketches span a vast range of styles and content, and varrying degrees of precision form highly sketchy ones such as the hand in Figure 5.15 to much more precise ones, such as the girl in Figure 5.13. Visual inspection confirms that our consolidation results are well aligned with viewer expectations.

We further validate our method via the evaluations and comparisons below.

Cross-Validation We evaluate both of our classifiers via a round-robin leave-one-out cross-validation on the 66 sketches in our training set. We leave one sketch out, train the classifier on the remaining sketches and then compare our classifier results to the ground truth annotations. Both classifiers achieve 99% accuracy (the local classifier fails on 118 sub-strip pairs out of 15959 and the global fails on 51 sub-strip pairs out of 6280).

To evaluate our end-to-end consolidation pipeline on this data, we similarly leave one sketch out, train both classifiers on the remaining sketches and then use those within our algorithm pipeline to consolidate the left-out sketch. We then measure the distance between our fitted outputs and those produced using ground truth annotations (Tab. 5.1, left). Our average distance, normalized by stroke width is less than 0.15, indicating very high degree of agreement. This number is significantly lower than the error obtained after applying only our preliminary step (0.179). Using distances to assess consolidation quality enables us to evaluate diverse methods via the same metric and is motivated by [112].

Comparison to Manual Consolidation. We compare our consolidation outputs on unseen data to manual consolidations. We collected manual consolidation annotations for 20 complete sketches from 12 participants. Each sketch was annotated by two participants. We evaluate agreement between participants by measuring the distance between the consolidated sketches produced using their annotations. As expected, while participant agreement is high, they are not 100% aligned (Ta-

Table 5.1: Average L_1 and $L_{\max}$ distances to consolidations generated using manual labelings. (left) result on our cross-validation set; (right) results on unseen annotation set. Our method achieves the best performance among all algorithms tested, approaching human performance.

	Cross Validation		Human Annotation	
	L_1	$L_{\max}$	L_1	$L_{\max}$
Human	-		0.548	12.924
[Stanko et al. 2020]	2.252	11.668	1.574	19.075
[Xu et al. 2019]	1.106	10.293	1.201	11.617
[Liu et al. 2018]	0.287	5.629	0.920	14.477
Our temporal consolidation	0.179	3.409	0.708	11.598
Our final	**0.149**	**3.141**	**0.645**	**10.353**

ble 5.1, right). We measure the degree to which our algorithm agrees with human choices by using the smaller between per-sketch distances (L_1 and $L_{\max}$) between our and manually fitted results for each input sketch and report the averages of these measurements. Our error of 0.645 is just 0.1 higher than the one between different human annotations, suggesting that our method is nearing human performance.

Comparison Against Prior Art. We compare our method against prior art in three related categories: raster-space consolidation methods, simultaneous consolidation and vectorization methods, and vector-space consolidation methods. We first demonstrated qualitatively by examples that our method visibly significantly outperforms all earlier approaches. To apply the raster space methods to our data, we rasterize our inputs as discussed in Appendix B.3.2. We focus our comparisons on the latest or best performing automatic methods in each category [67, 100, 111] (Figure 5.1, 5.13, 5.14, 5.15). While raster-space methods [86, 100, 111] often produce artifacts of both loss of details and under-consolidation, and the state-of-the-art vector consolidation approach of [67] over-merges, our method consistently produces consolidations better aligned with viewer expectations. As shown in Figure 5.16 and 5.17, earlier sketch consolidation methods [69, 97], often fail to adequately consolidate typical sketches; and simultaneous consolidation and vectorization methods [35, 73, 82], fail to generate viewer-expected outputs when applied to rasterizations of typical overdrawn vector sketches.

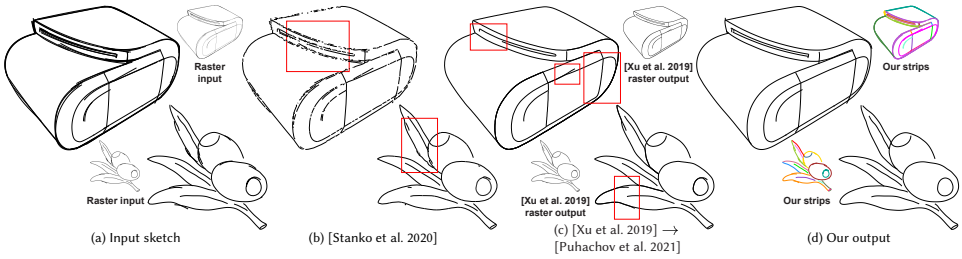


Figure 5.14: Consolidating typical inputs (a) using raster-space methods (b) [100] (c) [111] (vectorized using the method of [86]) often results in both loss of details and under-consolidation (raster consolidation outputs shown as insets). Our method (d) produces viewer expected consolidations on these inputs. Source: Yan et al. [112] (top), © Rami Alsafadi (below).

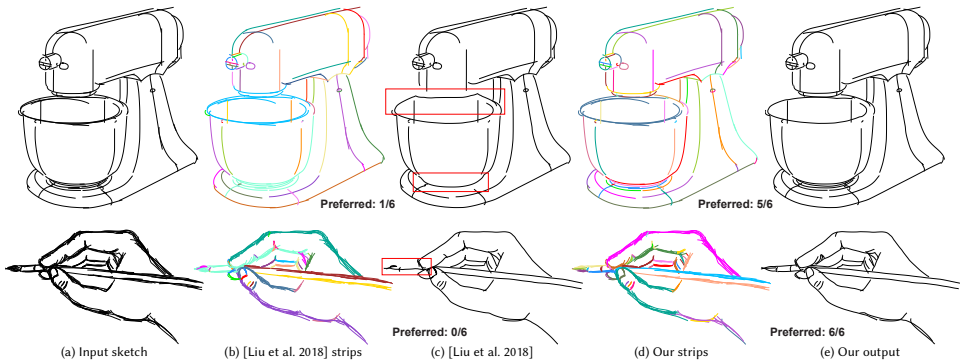


Figure 5.15: Our method (d,e) consistently produces consolidations better aligned with viewer expectations than those produced by the state-of-the-art vector consolidation approach of [67] (b,c) on diverse overdrawn inputs (a). Stroke grouping is shown with each strip rendered in a different color (b,d). Source: Gryaditskaya et al. [44] (top), © Tina Nowarre (below).

We measure mean and maximal distances between the outputs of these methods and our two ground truth corpuses as discussed above (to eliminate any misalignment we apply an ICP alignment step to all pairs of algorithmically generated and ground truth consolidations). This metric allows us to meaningfully compare vector and raster space methods performance. While the quality of vector space consolidation can in theory be evaluated by comparing which strokes are grouped together and which are not, the method of Liu et al. [67] uses a pre-process that

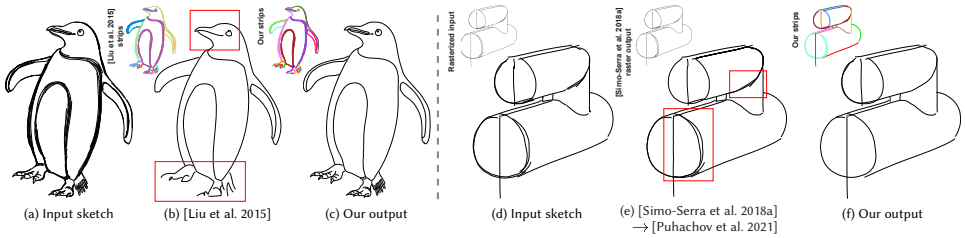


Figure 5.16: Earlier sketch consolidation methods, such as [69] (left) and [97] (right) often fail to adequately consolidate typical sketches (a,d) that our method succeeds on (c,f). On the left we used classifiers trained excluding the input shown (we have some results of [69] but no access to their code). Source: © Enrique Rosales (left), Gryaditskaya et al. [44] (right).

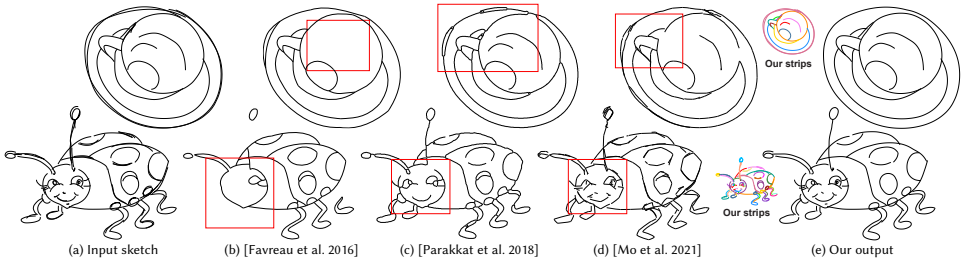


Figure 5.17: Comparison to simultaneous consolidation and vectorization methods: (b) [35], (c) [82], (d) [73] on typical overdrawn sketches (a). Our method (e) produces viewer expected results on this data. Source: © Val Novikov (top), © Rami Alsafadi (below).

deletes short strokes and splits raw strokes in high curvature areas. As a result, we cannot directly compare the strips it forms against ground truth data, as there is no one-to-one map between the strokes they operate on and the manually consolidated ones. To enable the most fair comparison we re-fit the strips produces by [67] using our fitting method. As reported in Table 5.1, the distances for all the methods we compare to are at least 30% higher than those achieved by StripMaker.

We compare the perceptual accuracy of our method against the prior approaches of [67, 100, 111] via a comparative study (Appendix B.3.3). Study participants were shown an input sketch and two consolidations of this sketch, one obtained using StripMaker and one generated using an alternative and were asked to evaluate which of the two was a cleaner and accurate version of the input. Overall we

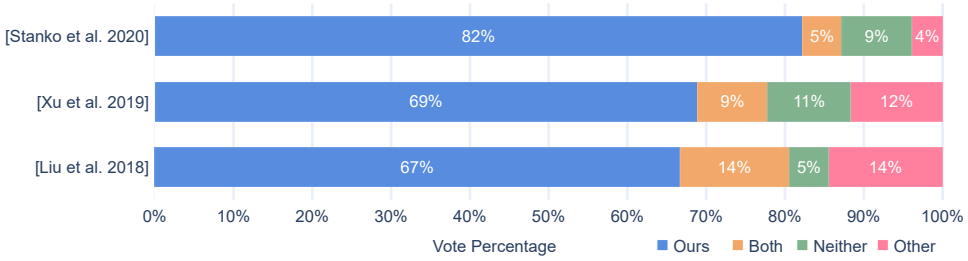


Figure 5.18: Comparative study summary: Participants preferred our results over all alternatives by a significant margin.

collected answers to 90 questions (30 per method), 6 answers per question (540 answers in total). Figure 5.18 summarizes the study results. In comparisons against the best performing alternative, participants preferred our results 67% of the time, preferred the alternative just 14% of the time, judged both results as equally good 14% percent of the time, and as equally bad 5% of the time. The measured preference was highly statistically significant ($p < 0.001$ for all methods). These numbers convincingly demonstrate that our consolidation method provides a significant improvement over the state-of-the-art. Figure 5.19 shows three inputs where viewers preferred the alternative over StripMaker (alternative methods were preferred on 6 out of 90 inputs shown). We further discuss the limitations in Section 6.1.

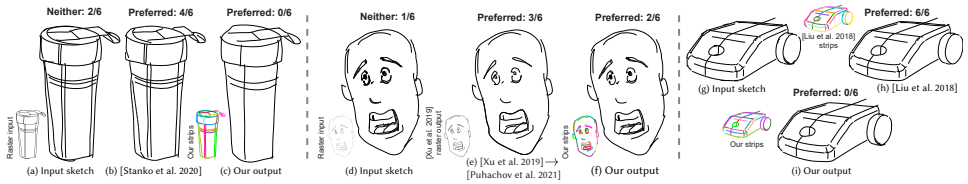


Figure 5.19: Limitations: Our results are not preferred on these three examples. Source: © Val Novikov (left), © Champ Semalulu (middle), Gryaditskaya et al. [44] (right).

Runtimes. Our median runtime across all inputs in our test set is 50 seconds per sketch. All runtimes measured on MacBook Pro (2020), Apple M1 chip (8-core CPU), 8 GB memory. The code is parallelized with 8 threads. The time bottleneck in our method, as expected, is the parameterization of sub-strip pairs.

Gini Coefficients. Tables 5.2 and 5.3 report the Gini coefficients of our classifiers. All features contribute to the decision making. The top five important features of local classifier are average distance, density average distance over local min width, median distance, median angle, density median distance over local min width. They corresponds to distances, density, and angle cues. The top five important features of contextual classifier are average distance, average distance over global average of average inter-stroke distances, average distance over global 90^{th} percentile of average inter-stroke distances, density average distance over local min width, median angle. They corresponds to distances, relative precision, density, and angle cues. For both local and contextual classifiers, density and angle are shown to be important which verifies the usage of these cues in Chapter 3. Distance is shown to be the most important cue based on Gini. While it is hard to incorporate the distance cue via simple thresholding in Chapter 3, in this Chapter, we are able to include this cue without the potential risks of using hard threshold, since we could model the interactions between cues automatically. For the contextual classifier, relative precision cue is shown to be important in addition to the ones important to local classifier.

Performance with different sets of features. We had experimented with removing different subsets of features from the classifiers. In all instances, performance declined or remained on par. Additionally, we experimented with removing different subsets of features from the classifiers. In all instances, performance declined or remained on par. Among the features of our classifiers, angle, narrowness, and density categories, in this order, have the most impact: removing them decrease the accuracy in the cross-validation experiment by 0.93%, 0.31%, and 0.06% respectively. We also experimented with adding the temporal distance between sub-strips as a classifier feature, performance did not improve.

Consolidation applications. The consolidated results produced by our method can be directly processed by downstream applications. For instance, applying topology cleanup [115] directly to a typical input (Figure 5.20a) produces numerous undesirable tiny regions (545 on this input) (Figure 5.20b); while consolidating

the input with our method produces the viewer expected topology that can facilitate colorization (Figure 5.20d). In addition, our output strips (Figure 5.20f) facilitate per-strip manipulations, such as recolorization with gradient based on per-strip parameterization (Figure 5.20g).

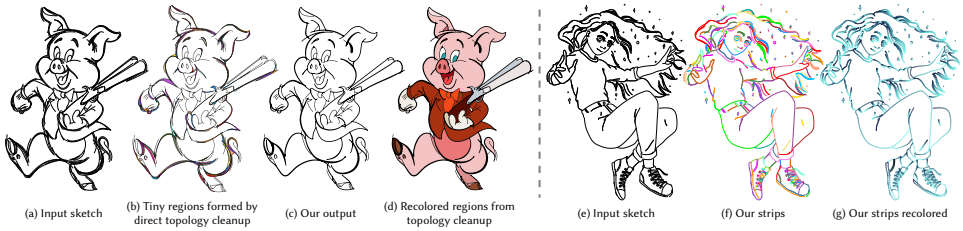


Figure 5.20: Consolidation applications: We follow consolidation by topology extraction [115] to facilitate colorization (ab); We use the consolidated strips to directly edit the input drawing (cd). Source: Yan et al. [112].

Table 5.2: Gini importances of local classifier features.

average angle	0.072
median angle	0.075
average distance	0.162
median distance	0.105
density avg distance to Max Width	0.058
density median distance2LocalMaxWidth	0.062
density avg distance2LocalMinWidth	0.146
density median distance2LocalMinWidth	0.073
density avg distance2width combined	0.066
densitymedian distance2width combined	0.053
density max 90th LocalMedianGap2width	0.002
density min 90th LocalMedianGap2width	< 0.001
density max 10th LocalMedianGap2width	0.002
density min 10th LocalMedianGap2width	< 0.001
side-by-side length to combined length	0.004
max avg narrowness	0.011
max median narrowness	0.012
min avg narrowness	0.004
min median narrowness	0.005
avg combined narrowness	0.003
median narrowness combined	0.002
max non-side-by-side to side-by-side	0.002
min non-side-by-side to side-by-side	0.041
max Avg non-side-by-side to Avg side-by-side	0.028
min Avg non-side-by-side to Avg side-by-side	0.007
parameterization velocity	0.003
parameterization alignment	0.001

Table 5.3: Gini importances of global classifier features.

average angle	0.065
median angle	0.075
average distance	0.146
median distance	0.067
density avg distance to Max Width	0.036
density median distance2LocalMaxWidth	0.031
density avg distance2LocalMinWidth	0.081
density median distance2LocalMinWidth	0.027
density avg distance2width combined	0.059
densitymedian distance2width combined	0.014
density max 90th LocalMedianGap2width	0.001
density min 90th LocalMedianGap2width	< 0.001
density max 10th LocalMedianGap2width	0.001
density min 10th LocalMedianGap2width	0.001
side-by-side length to combined length	0.011
max avg narrowness	0.006
max median narrowness	0.006
min avg narrowness	0.004
min median narrowness	0.003
avg combined narrowness	0.002
median narrowness combined	0.001
max non-side-by-side to side-by-side	0.004
min non-side-by-side to side-by-side	0.01
max Avg non-side-by-side to Avg side-by-side	0.012
min Avg non-side-by-side to Avg side-by-side	0.002
parameterization velocity	0.002
parameterization alignment	0.004
average distance2global average average distance	0.111
median distance2global average median distance	0.04
average distance2global median average distance	0.013
median distance2global median median distance	0.015
average distance2global p90th average distance	0.111
median distance2global p90th median distance	0.037

Chapter 6

Conclusion and Discussions

In this thesis, we addressed the barriers to direct usage of freehand vector sketches in downstream sketch processing applications: overdrawing strokes and inaccurate junctions. In current industry practice, these issues are solved via manual cleanup that is tedious and time-consuming. In Chapter 3, we identified basic local cues, i.e., density, angle, and narrowness, for consolidation based on perception research literature and our own observations, and established a parameterization based strategy for pointwise correspondence and measurement given an arbitrary number of strokes. Assisted by these building blocks, we presented *StrokeAggregator*, a method for automatic consolidation that groups raw strokes into strips and fits aggregated curves. In Chapter 4, we examined local and contextual cues guiding connectivity perception. To overcome data sparsity issue in learning, we observed that the connectivity problem can be decomposed into local binary decision and global inference. While the limited annotated data is insufficient for learning contextual cues, it is adequate to train local classifiers which can be integrated into a global solving framework. This gap closure method was evaluated on a variety of sketches and used in Chapter 5 to provide contextual information. In Chapter 5, we further improved on consolidation by accounting for not only local cues but also contextual and temporal cues. Conducting individual studies to obtain a threshold per cue as in Chapter 3 is no longer sustainable. We thus applied a strategy similar to Chapter 4: to train a local classifier and integrate it with global refinement steps designed for contextual and temporal cues. We demonstrated that this resulting method out-performs StrokeAggregator on a diverse test set. Our research on consolidation has been evaluated by a survey study [112] and concluded to generate the best path quality at that time; it also has started to benefit downstream applications, for instance, to significantly save computation time as a preprocessing step for a design sketch lifting method [45].

6.1 Future Work

The research on vector sketch consolidation and connectivity can be further explored in multiple directions. One immediate extension to our consolidation method in Chapter 5 is an interactive mode. Both of our consolidation methods are constructed without considering the semantics of the input sketch, since it is difficult if not impossible to quantify these semantics without large amounts of data. A bypass to this issue is to directly ask artists for instructions, similar to many semi-automatic methods discussed in Chapter 2. In terms of interaction design, these previous methods define the correction input as a procedure separate from sketching which negatively impacts intuitiveness of sketching for creation and interaction. For this future direction, the drawing system should work incrementally recording a new stroke and updating the current consolidated sketch on the fly. Instead of defining a different correction operation, the system should support user correction in the same form as regular sketching stroke so the sketching process can be carried out as a whole rather than be interleaved with distractive correction operations. In terms of technical challenge, this system requires real-time update which is infeasible if directly applying our current method in Chapter 5. This is because our current method has to recompute the expensive parameterization from scratch for every comparison, which can be slow when the number of existing strokes is high. To achieve real-time performance, the parameterization algorithm by Pagurek van Mossel et al. [79] needs to be modified to support a fast incremental construction.

Our consolidation method in Chapter 5 utilizes sketch precision as a contextual cue. Currently, this precision is measured on all but the current strip in question. This is representative under the assumptions that the initial strips are roughly correct and the ground truth strips in a sketch share homogenous precisions. However, these two assumptions may not always hold. Figure 6.1(a) shows an example where our method generates an initial result containing multiple over-merged strips, which biases the refinement to continue this incorrect trend and over-merges more in the final result. Figure 6.1(b) presents an example where the majority of ground truth strips are single-stroke strips. When considering the non-single-stroke strip (arrowed in Figure 6.1(b)), the method incorrectly decides that it is over-merged and produce the final over-segmented result. This heterogeneous property

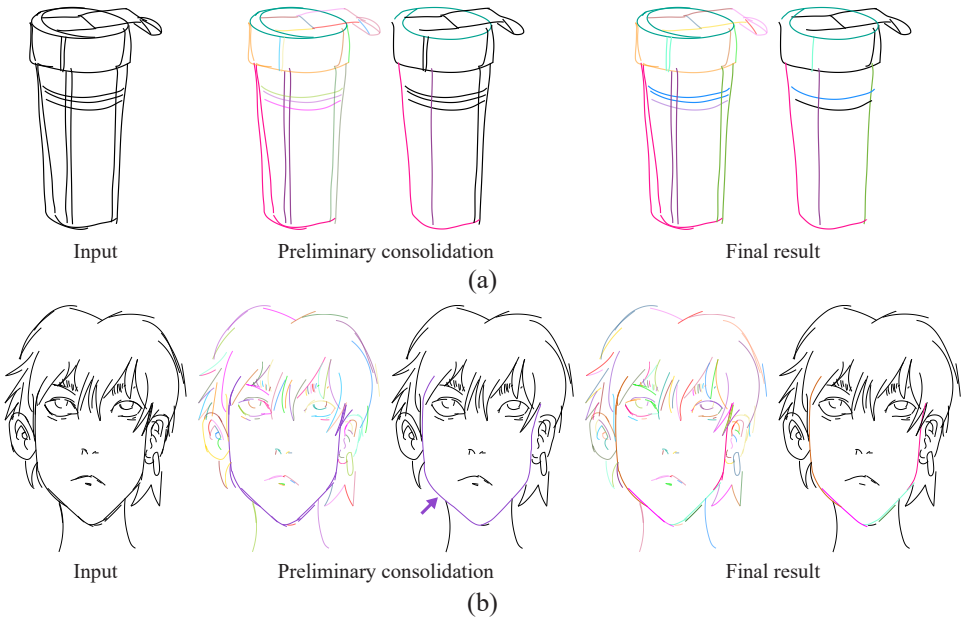


Figure 6.1: Limitation of StripMaker. The refinement step using sketch precision could fail and over-segment when the intermediate correctness assumption (a) and homogenous assumption (b) do not hold. The problematic strips are highlighted as opaque in the clustering view and as colorful in the clean line drawing view. Source: © Val Novikov (top), © Edwin González Espitia (below).

even within a single freehand sketch has been a constant challenge for both raster and vector consolidation methods. It would be interesting to consider adding a similarity matching step so a strip is only considered against the truly similar strips in the same sketch.

Apart from these instant extensions, the consolidation can be further studied for two questions. One is the shading and texture strokes in sketch. As surveyed by Yan et al. [112], the shading and texture strokes form a common component of sketch yet most methods including ours do not handle them in any manner. Research focusing on the shading and texture strokes, for example, automatic extraction of the shading and texture from a sketch, could be important for downstream sketch processing applications. Another question is the raster consolidation. As discussed above, the heterogeneous property posts a challenge to vector consolidation methods. The small handful of raster consolidation methods that accept over-

drawn inputs handle this challenge even more poorly, which is partially reflected by the heavy input resolution dependency. An exciting future direction would be to transfer findings in vector consolidation to the raster space while overcoming the reduced information provided by raster input.

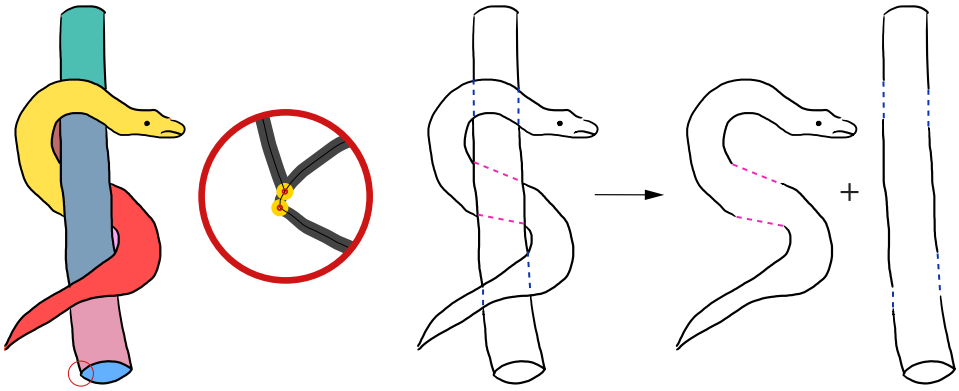


Figure 6.2: Potential extensions to our gap closure method. The accuracy of already formed junctions can be further improved by intelligently merging close misaligned intersections, for instance, around the lower corner of this snake example these three strokes do not intersect exactly (a). When paired up, junctions can imply occlusion, a necessary piece of information for animating occluded objects (b).

One direct extension to our gap closure method in Chapter 4 is to more closely examine remaining inaccuracies that are currently not tackled by our method. Like unintended gaps, artists could unintentionally misalign stroke endpoints forming an inaccurate junction as in Figure 6.2(a). Our current strategy is to move intersections to the nearby endpoint if some heuristic conditions are satisfied. Similarly, our method handles over-shot stroke segments using simple heuristics which may not always be efficient. Moreover, although our solve strategy avoids forming extra small regions, we do not have a strategy to clean up redundant small regions formed by intersections. One more ambitious future direction is to recover beyond just junctions. Vector graphics can have more advanced topology that provides the facilities for editing and animation [24, 25, 34]. For instance, a pair of T-junctions implies occlusion [11, 12] in Figure 6.2(b) and successful detection of underlying

topology is beneficial to applications, such as auto-completion of occluded contents.

Data sparsity is a common challenge for vector-space learning. Although we successfully combined local classifier and global inference framework for both consolidation and connectivity problems, our methods are unaware of semantics. Semantics-aware raster vision and graphics methods have seen huge successes in raster visual content creation [89] and processing recently (see [110] for a survey about learning based sketch processing methods). One may wonder if the current development could benefit research on sketch data. However, all these deep learning based methods are built upon vast sets of training examples, mostly annotated. Conducting the same scale of data annotations is more expensive due to the higher difficulties of vector annotation tasks and would not be unlike reinventing the wheel. One promising approach is to be less supervised, e.g., to apply unsupervision or self-supervision [13] or train on synthetic data [17, 48, 68]. Another existing potential solution is to wrap a vector layer on mature raster or text based models, as Vinker et al. [103] demonstrated, saving the efforts to build an equally powerful model from scratch.

Bibliography

[1] Adobe Inc. Adobe illustrator, 2022. URL https://adobe.com/products/illustrator. → page 19

[2] R. Arora, I. Darolia, V. P. Namboodiri, K. Singh, and A. Bousseau. Sketchsoup: Exploratory ideation using design sketches. In *Computer Graphics Forum*, volume 36, pages 302–312. Wiley Online Library, 2017. → pages 11, 22, 85

[3] P. Asente, M. Schuster, and T. Pettit. Dynamic planar map illustration. *ACM Trans. Graph.*, 26(3):30–es, July 2007. → page 19

[4] S.-H. Bae, R. Balakrishnan, and K. Singh. Ilovesketch: As-natural-as-possible sketching system for creating 3d curve models. In *Proc. UIST*, pages 151–160, 2008. → page 18

[5] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004. → page 31

[6] B. Bao and H. Fu. Vectorizing line drawings with near-constant line width. In *2012 19th IEEE International Conference on Image Processing*, pages 805–808, Sept. 2012. → page 13

[7] I. Baran, J. Lehtinen, and J. Popović. Sketching Clothoid Splines Using Shortest Paths. *Comput. Graph. Forum*, 29(2):655–664, 2010. → pages 12, 30, 126, 128

[8] P. Barla, J. Thollot, and F. Sillion. Geometric clustering for line drawing simplification. In *Proceedings of the Eurographics Symposium on Rendering*, 2005. URL http://maverick.inria.fr/Publications/2005/BTS05a. → pages 16, 26, 31

[9] T. Baudel. A mark-based interaction paradigm for free-hand drawing. In *Proc. UIST*, pages 185–192, 1994. → page 18

[10] M. Bessmeltsev and J. Solomon. Vectorization of Line Drawings via Polyvector Fields. *ACM Trans. Graph.*, 38(1):9:1–9:12, Jan. 2019. → page 13

[11] M. Bessmeltsev, W. Chang, N. Vining, A. Sheffer, and K. Singh. Modeling character canvases from cartoon drawings. *Transactions on Graphics (2015)*, 34(5), 2015. doi:10.1145/2801134. → pages 6, 26, 112

[12] M. Bessmeltsev, N. Vining, and A. Sheffer. Gesture3d: Posing 3d characters via gesture drawings. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH Asia 2016)*, 35(6), 2016. → pages 6, 98, 112

[13] A. K. Bhunia, P. N. Chowdhury, Y. Yang, T. M. Hospedales, T. Xiang, and Y.-Z. Song. Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5672–5681, 2021. → pages 13, 113

[14] Blender. Blender Cloud, 2021. URL https://cloud.blender.org/p/gallery/5b642e25bf419c1042056fc6. → pages 72, 73

[15] Blender. Grease pencil, 2022. URL https://www.blender.org/features/grease-pencil/. → page 58

[16] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Trans. Knowl. Discov. Data*, 10(1):5:1–5:51, 2015. → page 35

[17] C. Chan, F. Durand, and P. Isola. Learning to generate line drawings that convey geometry and semantics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7915–7925, 2022. → page 113

[18] S. Cheema, S. Gulwani, and J. LaViola. QuickDraw: Improving drawing experience for geometric diagrams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1037–1064, May 2012. → page 12

[19] J. Chen, G. Guennebaud, P. Barla, and X. Granier. Non-oriented mls gradient fields. In *Computer Graphics Forum*, volume 32, pages 98–109, 2013. → pages 13, 46

[20] J. Chen, Q. Lei, Y. Miao, and Q. Peng. Vectorization of line drawing image based on junction analysis. *Sci. China Inf. Sci.*, 58(7):1–14, July 2015. → page 13

[21] J. Chen, M. Du, X. Qin, and Y. Miao. An improved topology extraction approach for vectorization of sketchy line drawings. *Vis Comput*, 34(12): 1633–1644, Dec. 2018. → page 14

[22] Y.-J. Chu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965. → page 48

[23] P. Company, R. Plumed, P. A. C. Varley, and J. D. Camba. Algorithmic Perception of Vertices in Sketched Drawings of Polyhedral Shapes. *ACM Trans. Appl. Percept.*, 16(3):18:1–18:19, Aug. 2019. → pages xix, xx, 19, 61, 62, 73, 76, 78, 127

[24] B. Dalstein, R. Ronfard, and M. Van De Panne. Vector graphics complexes. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. → pages 6, 112

[25] B. Dalstein, R. Ronfard, and M. van de Panne. Vector graphics animation with time-varying topology. *ACM Trans. Graph.*, 34(4), July 2015. → pages 6, 112

[26] M. R. Davis and T. O. Ellis. The rand tablet: A man-machine graphical communication device. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), page 325–331, New York, NY, USA, 1964. Association for Computing Machinery. ISBN 9781450378895. → page 1

[27] T. K. Dey. *Curve and surface reconstruction: algorithms with mathematical analysis*, volume 23. Cambridge University Press, 2006. → page 63

[28] L. Donati, S. Cesano, and A. Prati. An Accurate System for Fashion Hand-Drawn Sketches Vectorization. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 2280–2286, Oct. 2017. → page 13

[29] L. Donati, S. Cesano, and A. Prati. A complete hand-drawn sketch vectorization framework. *Multimed Tools Appl*, 78(14):19083–19113, July 2019. → page 13

[30] J. Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71B(4):233–240, 1967. → page 48

[31] V. Egiazarian, O. Voynov, A. Artemov, D. Volkhonskiy, A. Safin, M. Taktasheva, D. Zorin, and E. Burnaev. Deep vectorization of technical

drawings. In *European Conference on Computer Vision*, pages 582–598. Springer, 2020. → page 13

[32] K. Eissen and R. Steur. *Sketching: Drawing Techniques for Product Designers*. Bis Publishers, 2008. → pages 11, 22, 27, 85

[33] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? *ACM Trans. Graph.*, 31(4):44:1–44:10, July 2012. → pages 58, 73

[34] E. Entem, A. D. Parakkat, M.-P. Cani, and L. Barthe. Structuring and layering contour drawings of organic shapes. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expressive '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450358927. → page 112

[35] J.-D. Favreau, F. Lafarge, and A. Bousseau. Fidelity vs. simplicity: a global approach to line drawing vectorization. *ACM Transactions on Graphics (TOG)*, 35(4):1–10, 2016. → pages xvii, xix, xxiii, 14, 19, 21, 51, 52, 57, 74, 75, 76, 77, 101, 103, 133

[36] M. Finch, J. Snyder, and H. Hoppe. Freeform vector graphics with controlled thin-plate splines. 30(6):1–10, dec 2011. ISSN 0730-0301. → page 12

[37] J. Fišer, P. Asente, S. Schiller, and D. Sýkora. Advanced drawing beautification with ShipShape. *Computers & Graphics*, 56:46–58, May 2016. → page 12

[38] S. Fourey, D. Tschumperlé, and D. Revoy. A fast and efficient semi-guided algorithm for flat coloring line-arts. In *Proceedings of the Conference on Vision, Modeling, and Visualization*, EG VMV '18, page 1–9, Goslar, DEU, 2018. Eurographics Association. doi:10.2312/vmv.20181247. → pages xvii, xix, 6, 12, 21, 57, 61, 74, 75, 77, 133

[39] M. Gangnet, J.-M. Thong, and J.-D. Fekete. Automatic Gap Closing for Freehand Drawing. In *ACM SIGGRAPH 94 Technical Sketch*, July 1994. → page 19

[40] S. Ge, V. Goswami, L. Zitnick, and D. Parikh. Creative Sketch Generation. In *International Conference on Learning Representations*, Sept. 2020. → pages xxiv, 73, 132

[41] S. Grabli, F. Durand, and F. X. Sillion. Density measure for line-drawing simplification. In *12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004. Proceedings.*, pages 309–318. IEEE, 2004. → page 12

[42] C. Grimm and P. Joshi. Just drawit: A 3d sketching system. In *Proc. SBIM*, pages 121–130, 2012. → pages 18, 86

[43] L. Grinsztajn, E. Oyallon, and G. Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. → page 94

[44] Y. Gryaditskaya, M. Sypesteyn, J. W. Hoftijzer, S. Pont, F. Durand, and A. Bousseau. OpenSketch: A richly-annotated dataset of product design sketches. *ACM Trans. Graph.*, 38(6):232:1–232:16, Nov. 2019. → pages xxii, xxiii, 11, 72, 100, 102, 103, 104, 127, 135, 136, 137

[45] Y. Gryaditskaya, F. Hähnlein, C. Liu, A. Sheffer, and A. Bousseau. Lifting freehand concept sketches into 3D. *ACM Trans. Graph.*, 39(6): 167:1–167:16, Nov. 2020. → pages 2, 19, 58, 74, 109

[46] Y. Guo, Z. Zhang, C. Han, W. Hu, C. Li, and T.-T. Wong. Deep Line Drawing Vectorization via Line Subdivision and Topology Reconstruction. *Computer Graphics Forum*, 38(7):81–90, Oct. 2019. → page 13

[47] D. Ha and D. Eck. A Neural Representation of Sketch Drawings. In *International Conference on Learning Representations*, Feb. 2018. URL https://openreview.net/forum?id=Hy6GHpkCW. → pages 72, 73, 127

[48] F. Hähnlein, C. Li, N. J. Mitra, and A. Bousseau. Cad2sketch: Generating concept sketches from cad sequences. *ACM Transactions on Graphics (TOG)*, 41(6):1–18, 2022. → page 113

[49] R. Hess and D. Field. Integration of contours: new insights. *Trends in Cognitive Sciences*, 3(12):480–486, 1999. → pages 32, 98, 129

[50] T. K. Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282, 1995. → page 94

[51] T. Igarashi, S. Matsuoka, S. Kawachiya, and H. Tanaka. Interactive beautification: A technique for rapid geometric design. In *Proceedings of*

the 10th Annual ACM Symposium on User Interface Software and Technology, UIST '97, pages 105–114. Association for Computing Machinery, Oct. 1997. → page 12

[52] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005. → page 12

[53] J. Jiang, H. S. Seah, H. Z. Liew, and Q. Chen. Challenges in Designing and Implementing a Vector-Based 2D Animation System. In *The Digital Gaming Handbook*. CRC Press, 2020. → pages xix, 77

[54] J. Jiang, H. S. Seah, and H. Z. Liew. Handling gaps for vector graphics coloring. *Vis Comput*, 37(9):2473–2484, Sept. 2021. → pages xiii, 21, 73

[55] G. Johnson, M. D. Gross, J. Hong, and E. Yi-Luen Do. Computational support for sketching in design: A review. *Foundations and Trends® in Human–Computer Interaction*, 2(1):1–93, jan 2009. ISSN 1551-3955. doi:10.1561/1100000013. → pages 6, 58

[56] R. D. Kalnins, P. L. Davidson, L. Markosian, and A. Finkelstein. Coherent stylized silhouettes. *ACM Trans. Graph.*, 22(3):856–861, 2003. → page 12

[57] G. Kanizsa. *Organization in Vision: Essays on Gestalt Perception*. Praeger, Sept. 1979. → page 58

[58] L. B. Kara and K. Shimada. Sketch-based 3d-shape creation for industrial styling design. *IEEE Computer Graphics and Applications*, 27(1):60–71, 2007. → page 16

[59] M. Keuper, E. Levinkov, N. Bonneel, G. Lavoué, T. Brox, and B. Andres. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1751–1759, 2015. → page 31

[60] B. Kim, O. Wang, A. C. Öztireli, and M. Gross. Semantic segmentation for line drawing vectorization using neural networks. In *Computer Graphics Forum*, volume 37, pages 329–338. Wiley Online Library, 2018. → page 13

[61] K. Koffka. *Principles of Gestalt Psychology*. International library of psychology, philosophy, and scientific method. Routledge & K. Paul, 1955. → pages 62, 86

[62] Krita. Krita, 2021. URL https://krita.org/. → page 74

[63] I.-K. Lee. Curve reconstruction from unorganized points. *Computer aided geometric design*, 17(2):161–177, 2000. → pages 45, 47

[64] D. Levin. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*, pages 37–49. 2004. → pages 45, 47

[65] C. Li, H. Pan, A. Bousseau, and N. J. Mitra. Free2cad: parsing freehand drawings into cad commands. *ACM Transactions on Graphics (TOG)*, 41 (4):1–16, 2022. → page 12

[66] H. Lipson and M. Shpitalni. Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design*, 28(8): 651 – 663, 1996. → page 12

[67] C. Liu, E. Rosales, and A. Sheffer. StrokeAggregator: consolidating raw sketches into artist-intended curve drawings. *ACM Transactions on Graphics (TOG)*, 37(4):97, 2018. ISSN 0730-0301. doi:10.1145/3197517.3201314. URL https://dl.acm.org/citation.cfm?id=3201314. → pages xxii, 82, 84, 85, 86, 87, 89, 98, 99, 101, 102, 103

[68] D. Liu, M. Fisher, A. Hertzmann, and E. Kalogerakis. Neural strokes: Stylized line drawing of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14204–14213, 2021. → page 113

[69] X. Liu, T.-T. Wong, and P.-A. Heng. Closure-aware sketch simplification. *ACM Trans. Graph.*, 34(6):168, 2015. → pages xii, xvi, xvii, xxii, 7, 17, 19, 30, 31, 43, 50, 51, 52, 53, 54, 62, 82, 101, 103, 126, 133

[70] J. Lu, F. Yu, A. Finkelstein, and S. DiVerdi. Helpinghand: Example-based stroke stylization. *ACM Trans. Graph.*, 31(4), jul 2012. ISSN 0730-0301. → page 2

[71] B. Manda, P. P. Kendre, S. Dey, and R. Muthuganapathy. Sketchcleannet—a deep learning approach to the enhancement and correction of query sketches for a 3d cad model retrieval system. *Computers & Graphics*, 107:73–83, 2022. → page 13

[72] J. McCrae and K. Singh. Sketching piecewise clothoid curves. *Computers & Graphics*, 33(4):452–461, 2009. → pages 30, 126

[73] H. Mo, E. Simo-Serra, C. Gao, C. Zou, and R. Wang. General virtual sketching framework for vector line art. *ACM Trans. Graph.*, 40(4), jul 2021. ISSN 0730-0301. doi:10.1145/3450626.3459833. → pages xxiii, 14, 101, 103

[74] S. Murugappan, S. Sellamani, and K. Ramani. Towards beautification of freehand sketches using suggestions. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 69–76. Association for Computing Machinery, Aug. 2009. → page 12

[75] L. Nan, A. Sharf, K. Xie, T.-T. Wong, O. Deussen, D. Cohen-Or, and B. Chen. Conjoining gestalt rules for abstraction of architectural drawings. *ACM Transactions on Graphics (TOG)*, 30(6):1–10, 2011. → page 12

[76] G. Noris, D. Sỳkora, A. Shamir, S. Coros, B. Whited, M. Simmons, A. Hornung, M. Gross, and R. Sumner. Smart scribbles for sketch segmentation. In *Computer Graphics Forum*, volume 31, pages 2516–2527, 2012. → pages 18, 19, 86

[77] G. Noris, A. Hornung, R. W. Sumner, M. Simmons, and M. Gross. Topology-driven vectorization of clean line drawings. *ACM Trans. Graph.*, 32(1):4:1–4:11, Feb. 2013. → page 13

[78] G. Orbay and L. B. Kara. Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Transactions on Visualization and Computer Graphics*, 17:694–708, 2011. ISSN 10772626. doi:10.1109/TVCG.2010.105. → pages xii, xv, xvi, xvii, 16, 17, 31, 34, 47, 50, 51, 52, 53, 54, 133

[79] D. Pagurek van Mossel, C. Liu, N. Vining, M. Bessmeltsev, and A. Sheffer. Strokestrip: Joint parameterization and fitting of stroke clusters. *ACM Transactions on Graphics*, 40(4), 2021. doi:10.1145/3450626.3459777. → pages xxi, 4, 16, 58, 73, 82, 84, 85, 88, 89, 91, 95, 97, 98, 99, 110, 133, 136

[80] A. D. Parakkat, U. B. Pundarikaksha, and R. Muthuganapathy. A delaunay triangulation based approach for cleaning rough sketches. *Computers & Graphics*, 74:171–181, 2018. → page 14

[81] A. D. Parakkat, P. Madipally, H. H. Gowtham, and M.-P. Cani. *Interactive Flat Coloring of Minimalist Neat Sketches*. The Eurographics Association, 2020. → pages 19, 21, 61, 73

[82] A. D. Parakkat, M.-P. R. Cani, and K. Singh. Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450380966. doi:10.1145/3411764.3445215. → pages xii, xix, xxiii, 6, 12, 14, 15, 19, 21, 58, 61, 73, 74, 75, 77, 101, 103, 133

[83] B. Paulson and T. Hammond. PaleoSketch: Accurate primitive sketch recognition and beautification. In *Proceedings of the 13th International Conference on Intelligent User Interfaces*, IUI '08, pages 1–10. Association for Computing Machinery, Jan. 2008. → page 12

[84] T. Pavlidis and C. J. Van Wyk. An automatic beautifier for drawings and illustrations. *SIGGRAPH Comput. Graph.*, 19(3):225–234, jul 1985. ISSN 0097-8930. doi:10.1145/325165.325240. → page 12

[85] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12: 2825–2830, 2011. → pages 72, 99

[86] I. Puhachov, W. Neveu, E. Chien, and M. Bessmeltsev. Keypoint-driven line drawing vectorization via polyvector flow. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)*, 40(6), 12 2021. → pages xxii, 13, 15, 99, 101, 102

[87] A. Qi, Y. Gryaditskaya, J. Song, Y. Yang, Y. Qi, T. M. Hospedales, T. Xiang, and Y.-Z. Song. Toward Fine-Grained Sketch-Based 3D Shape Retrieval. *IEEE Trans. Image Process.*, 30:8595–8606, 2021. → pages 73, 127

[88] Y. Qu, T.-T. Wong, and P.-A. Heng. Manga colorization. *ACM Trans. Graph.*, 25(3):1214–1220, July 2006. → pages 6, 12, 19

[89] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. → page 113

[90] P. L. Rosin. Grouping curved lines. In *BMVC*, pages 1–10. Citeseer, 1994. → pages 16, 26

[91] B. Sadri and K. Singh. Flow-complex-based shape reconstruction from 3d curves. *ACM Trans. Graph.*, 33(2), apr 2014. → page 63

[92] P. Sangkloy, N. Burnell, C. Ham, and J. Hays. The sketchy database: Learning to retrieve badly drawn bunnies. *ACM Trans. Graph.*, 35(4): 119:1–119:12, July 2016. → page 73

[93] K. Sasaki, S. Iizuka, E. Simo-Serra, and H. Ishikawa. Joint Gap Detection and Inpainting of Line Drawings. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5768–5776, July 2017. → pages xix, 20, 74, 75, 77, 133

[94] C. Shao, A. Bousseau, A. Sheffer, and K. Singh. CrossShade: Shading concept sketches using cross-section curves. *ACM Trans. Graph.*, 31(4), jul 2012. ISSN 0730-0301. → pages 2, 12, 61, 74

[95] A. Shesh and B. Chen. Efficient and dynamic simplification of line drawings. In *Computer Graphics Forum*, volume 27, pages 537–545, 2008. → page 16

[96] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa. Learning to simplify: Fully convolutional networks for rough sketch cleanup. *ACM Trans. Graph.*, 35(4):121:1–121:11, 2016. → pages 13, 19, 20

[97] E. Simo-Serra, S. Iizuka, and H. Ishikawa. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Trans. Graph.*, 37(1):11:1–11:13, 2018. → pages xix, xxii, 13, 19, 20, 51, 52, 74, 75, 77, 101, 103, 133, 136

[98] E. Simo-Serra, S. Iizuka, and H. Ishikawa. Real-time data-driven interactive rough sketch inking. *ACM Trans. Graph.*, 37(4):98:1–98:14, 2018. → page 13

[99] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proc. Symposium on Geometry Processing*, pages 175–184, 2004. → page 45

[100] T. Stanko, M. Bessmeltsev, D. Bommes, and A. Bousseau. Integer-grid sketch simplification and vectorization. In *Computer Graphics Forum*, volume 39, 2020. → pages xii, xxii, 14, 82, 99, 101, 102, 103, 136

[101] D. Sýkora, J. Dingliana, and S. Collins. LazyBrush: Flexible Painting Tool for Hand-drawn Cartoons. *Comput. Graph. Forum*, 28(2):599–608, 2009. → pages xix, 6, 12, 19, 58, 74, 75

[102] Y. Thiel, K. Singh, and R. Balakrishnan. Elasticurves: Exploiting stroke dynamics and inertia for the real-time neatening of sketched 2D curves. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 383–392. Association for Computing Machinery, Oct. 2011. → page 12

[103] Y. Vinker, E. Pajouheshgar, J. Y. Bo, R. C. Bachmann, A. H. Bermano, D. Cohen-Or, A. Zamir, and A. Shamir. Clipasso: Semantically-aware object sketching. *ACM Trans. Graph.*, 41(4), jul 2022. ISSN 0730-0301. doi:10.1145/3528223.3530068. URL https://doi.org/10.1145/3528223.3530068. → page 113

[104] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012. → pages 2, 23, 26, 27, 62, 86

[105] S. Wang, Q. Zhang, S. Wang, X. Jing, and M. Gao. Endpoint fusing method of online freehand-sketched polyhedrons. *Vis Comput*, 36(2): 291–303, Feb. 2020. → page 20

[106] Y. Wang, Y. Chen, J. Liu, and X. Tang. 3D reconstruction of curved objects from single 2D line drawings. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1834–1841, June 2009. → page 20

[107] B. Wilson and K.-L. Ma. Rendering complexity in computer-generated pen-and-ink illustrations. In *Proc. NPAR*, pages 129–137, 2004. → page 12

[108] C. Winder and Z. Dowlatabadi. *Producing Animation*. CRC Press, third edition, 2019. → page 11

[109] B. Xu, W. Chang, A. Sheffer, A. Bousseau, J. McCrae, and K. Singh. True2form: 3d curve networks from 2d sketches via selective regularization. *ACM Trans. Graph.*, 33(4):131:1–131:13, 2014. → pages 2, 12, 61, 74

[110] P. Xu, T. M. Hospedales, Q. Yin, Y.-Z. Song, T. Xiang, and L. Wang. Deep learning for free-hand sketch: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 45(1):285–312, 2022. → pages 14, 113

[111] X. Xu, M. Xie, P. Miao, W. Qu, W. Xiao, H. Zhang, X. Liu, and T.-T. Wong. Perceptual-aware sketch simplification based on integrated vgg

layers. *IEEE Transactions on Visualization and Computer Graphics*, 2019. → pages xii, xxii, 13, 15, 82, 99, 101, 102, 103, 136

[112] C. Yan, D. Vanderhaeghe, and Y. Gingold. A benchmark for rough sketch cleanup. *ACM Trans. Graph.*, 39(6), nov 2020. ISSN 0730-0301. → pages xx, xxii, xxiii, 6, 8, 11, 15, 16, 58, 81, 85, 87, 99, 100, 102, 106, 109, 111, 135, 136, 137

[113] C. Yan, J. J. Y. Chung, K. Yoon, Y. Gingold, E. Adar, and S. R. Hong. FlatMagic: Improving flat colorization through ai-driven design for digital comic professionals. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI, 2022. → pages 6, 20

[114] W. Yang, H.-S. Seah, Q. Chen, H.-Z. Liew, and D. Sýkora. FTP-SC: Fuzzy Topology Preserving Stroke Correspondence. *Comput. Graph. Forum*, 37 (8):125–135, 2018. → page 58

[115] J. Yin, C. Liu, R. Lin, N. Vining, H. Rhodin, and A. Sheffer. Detecting viewer-perceived intended vector sketch connectivity. *ACM Transactions on Graphics*, 41, 2022. → pages xxiii, 83, 88, 89, 94, 98, 105, 106

[116] L. Zhang, Y. Ji, and C. Liu. Danbooregion: An illustration region dataset. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 137–154. Springer, 2020. → page 20

[117] S.-H. Zhang, T. Chen, Y.-F. Zhang, S.-M. Hu, and R. R. Martin. Vectorizing Cartoon Animations. *IEEE Trans. Vis. Comput. Graph.*, 15(4): 618–629, July 2009. → pages 14, 21

Appendix A
Preprocessing Strokes

Raw strokes captured via a stylus-on-tablet interface are often noisy due to both involuntary hand movement and capture software inaccuracy [7, 72]. In particular, such interfaces often do not accurately detect when the artist lifts the stylus away from the tablet, leaving small "hook" sections at the ends of strokes. In some cases, as noted by Liu et al [69] artists barely lift the stylus up inbetween drawing near parallel strokes, in which case the capture interface records multiple strokes as one. We account for these artifacts by pre-processing raw strokes. The pre-processing strategies for Chapter 3, 4, 5 are described in Section A.1, A.2, A.3 respectively. In addition, we detail the consolidation step used to pre-process input sketches for Chapter 4 in Section A.2.

A.1 StrokeAggregator: Artist-Intended Vector Sketch Consolidation

We smooth and densely resample the strokes using the Cornucopia algorithm [7]. As we seek to preserve the input stroke shape as much as possible, we set Cornucopia "error cost" to 5, which keeps the output stroke very close to the input. We cut each original stroke at Cornucopia-detected C^0 discontinuities, as well as at sharp curvature extrema where the curvature is both high (larger than 0.125) and distinct from that along the rest of the curve (at least three times the median curvature). Since the hooks at the end of strokes are a capture artifact and not part of the intended artist input, we delete their hanging portion (we classify a segment between a detected discontinuity and an end point as a hook if it is both short in absolute terms $15W_s$ and forms less than 15% of the overall stroke length).

A.2 Detecting Viewer-Perceived Intended Vector Sketch Connectivity

Hook Removal. When finishing a stroke, if the artist rapidly switches direction before the pen raise is registered by the drawing device, an unintentional hook can appear. These can interfere with tangent and distance computations. Handling hooks robustly remains an open problem and is not a contribution of our paper. Let vertices that remain after applying the Ramer-Douglas-Peucker algorithm be corners. We use a simple heuristic: the section from the endpoint to the first corner is a hook if the distance from the endpoint p_1 to the line segment between its two nearest corners is shorter than $W_1 f$, where f is a parameter, and the hook is shorter than $\min(1.5W_1 f, \frac{1}{2}L_1)$. We use $f = 3$ for Company et al. [23], Gryaditskaya et al. [44], Ha and Eck [47], Qi et al. [87] and $f = 1$ for all other sources. We preserve near-connections by only de-hooking an endpoint if it would not increase the envelope distance to its nearest stroke by more than a factor of 2. Across all processed inputs, we further manually removed 10 hooks not found by the criteria above.

Consolidation. Most of the inputs we process were not consolidated previously, and thus may contain overdrawing. To align stroke length and local context feature computations to human perception, we weakly consolidate our inputs using a simple heuristic that accounts for the presence of varying stroke widths and light oversketching in our data. We locate all pairs of partially side-by-side strokes and densely sample their side-by-side sections using orthogonal cross-sections. If the sections have roughly similar lengths (ratio $\in [1/1.2, 1.2]$), have similar tangent directions at corresponding cross-section points ($< 20°$) that are themselves close enough (all $< \frac{3}{2}\max(W_1, W_2)$), the endpoints are close enough ($< \max(W_1, W_2)$), and if either the endpoints both overlap with the other stroke or the substrokes are both longer than their pen widths, we replace the two strokes with a single stroke fitted to both. We repeat this on all pairs until no more strokes can be merged. Finally, we chain strokes by merging them into a single stroke when their endpoints overlap and the endpoint tangents align within 20°.

Dangling Endpoints. A subset of stroke endpoints in our drawings already overlap other strokes, and can be connected without consulting our classifier. These endpoints are non-dangling. In the case of an overshot connection, where the two stroke centerlines intersect, we define the intersection diameter d as the largest stroke width at the intersection. An endpoint is non-dangling if the length of the overshot portion is shorter than 15% of the larger of the stroke length and d, and the Euclidean distance from the intersection to the edge of the endpoint cap is less than $1.5d$.

Self-Connections. We define a connection between points p_1, p_2 on the same stroke to be valid if they form a loop—if $\max(3\|p_1 - p_2\|, \pi W_1)$ is less than the distance between p_1, p_2 along the stroke. We then define the projection of an endpoint onto its own stroke as the closest valid point, if it exists. From there, feature computations work as described in the main paper.

A.3 StripMaker: Perception-driven Learned Vector Sketch Consolidation

We evenly resample all strokes in the inputs, with the sampling rate set to 1.2 times the stroke width, and the minimal number of samples per stroke set to 5. As typical of methods operating on raw vector sketches in Chapter 3, 4, we remove hook artifacts resulting from the device continuing to record pen motion after the user lifts it off the touch screen. We use a hook-removal method that follows Chapter 3 but remove potential hook sub-strokes only when their length is below 8 times the stroke width. We cut strokes at the points where the angle between consecutive tangents exceeds 90° or at C^0 corners detected by Baran et al. [7] if the tangents at these corners differ by more than 25°.

Appendix B
Study Details

User study is a critical component of research summarized in this thesis. Via these studies, we determine algorithm parameters (Section B.1), validate and evaluate the results (Section B.2,B.3), establish ground truth for training and testing (Section B.3). In these following sections, we describe the study designs and provide the study findings in details.

B.1 StrokeAggregator: Perception Driven Parameter Setting

To cluster strokes, we employ three perception motivated parameters: angular compatibility threshold T_a, relative proximity factor T_p, and curve narrowness threshold T_n. We rely on prior research to set the angular threshold T_a [49], but have no such sources for the other two parameters. We set these parameters to values consistent with human perception by conducting two informal perceptual studies.

Narrowness. To establish the narrowness threshold, we show participants a range of rectangles with varying short to long side ratios S_r (Figure B.1,top). In total, we show viewers 36 questions in randomized order. Intuitively, we expect viewers to perceive rectangles with low ratios as lines (thick or thin) and those with high ratios as actual rectangles. We ask viewers "Does the image below show a line (thick or thin) or a rectangle?" and provide them three answer options "thin line, thick line, rectangle". As the answers summary (Figure B.1, right) shows, there is a strong correlation between participant responses and the ratio S_r, confirming our hypothesis that viewers use short to long side ratios to determine if the shape they look at is one or two dimensional. To avoid forming two-dimensional clusters, we use a threshold designed to ensure that approximately 2/3 of respondents perceive the input as a line. In our computations, we use the long to short side ratio; we use

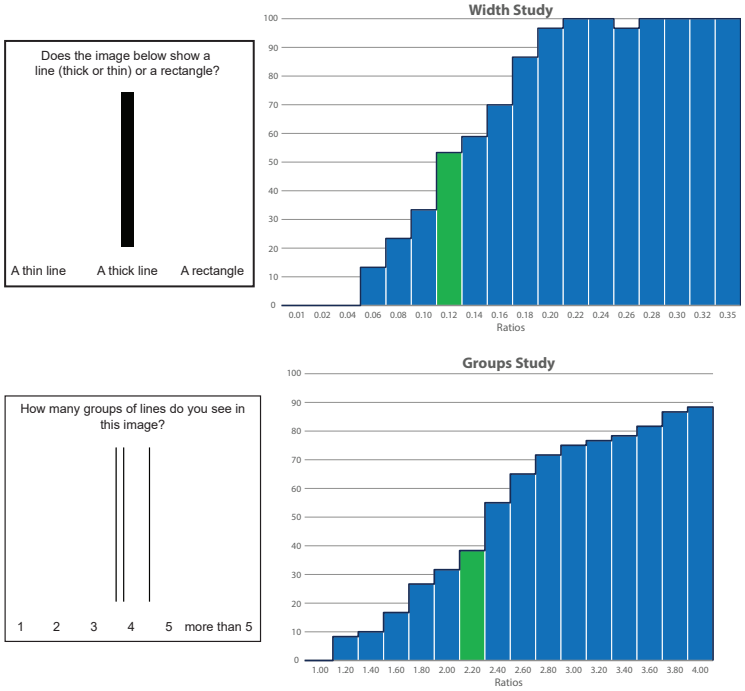


Figure B.1: Narrowness threshold question examples and answer distribution (top); proximity question examples and answer distributions (bottom).

a round value setting the threshold to $T_n = 8.5 \approx 1/0.117$.

Proximity Factor. To establish the proximity factor, we show participants triplets or quadruplets of vertical lines with different spacing (Figure B.1,bottom). For triplets we keep the distance d_1 between one pair of lines fixed, and have the third line placed to the right or left of them at intervals d_2 varying from d_1 to $3d_1$. For quadruplets we use the same placing strategy for three of the lines but place the fourth line far away from the others—at distance $6d_1$ away. In total, we show viewers 64 questions in randomized order. We ask the viewers "How many groups of lines do you see in this image?" and provide six answer options "1, 2, 3, 4, 5, more than 5". We use those to derive the *separation ratio* $D_r = d_2/d_1$ at which the viewers separate the third line from the first two. For triplets we look at the number of 1 versus 2 answers and for quadruplets at the number of 2 versus 3

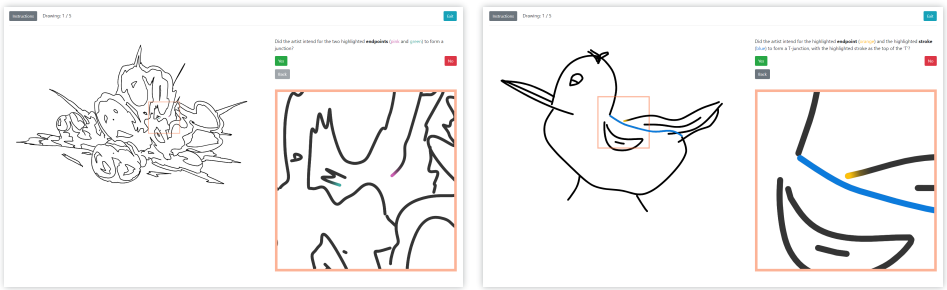
answers. As the answers summary (Figure B.1,bottom) shows, there is a strong correlation between participant responses and the ratio D_r, confirming the hypothesis that proximity ratio plays a major role in perceptual grouping. The answer curves for both questions, the one with only the potentially grouped lines, and the one with an extra "support" line are essentially identical. This confirms our hypothesis that one can assess relative proximity within a potential cluster without considering distance to other clusters. In our computations, we use the threshold as admissible upper bound, thus to obtain conservative clustering results, we again seek a number at which approximately 2/3 of respondents perceive the input as a single cluster. We thus set $T_d = 2.1$.

We collected responses from 15 different participants for each study. The validity of these studies is further corroborated by the fact that the parameters gleaned from them allowed us to consolidate a wide range of raw input drawings in a manner consistent with viewer expectations (Section 3.5).

B.2 Detecting Viewer-Perceived Intended Vector Sketch Connectivity: Study Design

Junction Annotation Study. To validate our final connection decisions, we collected additional manual annotations of 91 potential end-to-end and T-junctions across 10 drawings. In this study (Fig. B.2), for a given question, participants were shown a full line drawing with colors indicating potential endpoint-endpoint connections (endpoints coloured pink and green with gradients) and T-junctions (the endpoint coloured orange with a gradient and the other stroke flatly painted in blue), as well as a zoomed-in view around the potential junction in question. Participants were shown "a series of magnified views where one or two strokes are highlighted at the region of interest." Participants were then asked to "identify whether the strokes were intended by the artist to form a junction at the highlighted region of interest" and to answer the question "Did the artist intend for the two highlighted endpoints (pink and green) to form a junction?" or "Did the artist intend for the highlighted endpoint (orange) and the highlighted stroke (blue) to form a T-junction, with the highlighted stroke as the top of the 'T'?", depending on

the shown junction type. We recruited 16 non-expert participants (nine males and seven females, split between two sessions with 43 and 48 questions respectively), resulting in each potential junction labelled by eight participants. Two examples of the study question are shown in Fig. B.2.



End-end junction question example (a)

T-junction question example (b)

Figure B.2: Junction annotation study example questions and interface. The full line drawing is shown on the left, the zoomed-in view of the potential junction in question and the corresponding question is shown on the right. In both views, we use color gradients to indicate endpoints (pink and green in (a) and orange in (b)) and a solid blue to indicate the non-endpoint stroke of a T-junction (b). Images left and right ©Nahu under CC BY 4.0; [40].

Comparative Study. We conducted a comparative study to evaluate human perceptual preference between our method and existing gap closure methods. Participants were shown an input line drawing on top (A), and colorizations of this drawing obtained using our method and an alternative method, randomly assigned to (B) and (C) below. Participants were asked to "envision which strokes in [the input] drawings are intended by the artist to form closed loops," to "Identify the differences between the two [shown] colorings (ignore small color bleedings)," and then to answer "Which of the images on the bottom. (B) or (C), better corresponds to the partition you envisioned?" by selecting from "(B)," "(C)," "Both," and "Neither." We recruited 30 participants (19 male, 11 female), resulting in six responses per question for 27 inputs from 11 data sources with diverse authors and styles. We used the same 27 inputs for all five comparison methods and ensured that no

drawing was shown more than once in the same questionnaire.

We generated results for comparison methods by providing participants with inputs rasterized at 600px for Favreau et al. [35], Fourey et al. [38], Sasaki et al. [93], and Simo-Serra et al. [97] and Parakkat et al. [82]. We generated the colorizations of resulting closed loops by recoloring the output colorizations from Fourey et al. [38] and Parakkat et al. [82] (setting their "unassigned" pixels to light gray); by identifying and coloring closed loops in the vector outputs of Favreau et al. [35]; and by first binarizing the output grayscale raster images with a threshold of 0.5 (for pixel intensities in [0, 1]) and then flood-filling with a size of 1 px for Sasaki et al. [93] and Simo-Serra et al. [97]. We chose colors for each result pair in a question such that the corresponding closed loops between the two results were assigned the same color, and different closed loops within a partition were given different colors.

B.3 StripMaker: Study Details

B.3.1 Data Collection

Training data corpus

We use 66 manually annotated sketches generated by the authors of Pagurek van Mossel et al. [79] for testing curve fitting to strips, as our training data. These sketches are sourced from multiple prior publications, including StrokeAggregator (Chapter 3), [69, 78] and different artists.

Local Classifier We generate both positive and negative training examples using the dataset above. In generating the training examples, we recall that our classifier is designed to match viewer perception, namely given two groups of strokes that viewers perceive as strips, it assesses if the combined group of strokes is also perceived as a strip. As such, for all the positive examples in the training data we want the combined group of strokes to be also perceived as strip, and for the negative ones we want the combined group to not be perceived as a strip. Notably, a random subset of strokes taken from a human annotated strip may or may not be perceived

as a strip on its own (e.g., in isolation the farthest apart strokes in a wide strip may be too far from one another to be perceived as belonging together).

We first generate negative training examples that satisfy the criteria above by forming pairs of complete ground truth strips paired with either any other stroke in the drawing, or with another complete ground truth strip. In both cases both elements of such pairs are by definition perceived as strips, but are not perceived as being part of the same strip.

To generate the positive examples we recall that each strip is a time-ordered sequence of strokes. We therefore take manually labeled strips, randomly pick a moment in time splitting the sequence into the parts before and after that moment. The union of these parts forms a ground truth strip, and both parts are likely to be perceived as sub-strips due to temporal persistence. We identify and discard examples where this is not the case.

We exclude positive examples from our training data if the median distance between the sub-strips (measured along parameterization isolines) is twice as large as the median of these distance measured on the entire training set. We exclude positive examples if the parameterization of the two sub-strips alone is not consistent with the parameterization of the strip they originate from (i.e., there is no monotone mapping from one to the other).

We remove training examples on which feature computation fails, including the ones where the parameterization method we use produces highly distorted results. Lastly, to better reflect the distribution of classifier inputs, we pre-filter the examples using the criteria in Sec. 5.3.1, and manually remove additional ambiguous examples.

Global Classifier. We form positive training examples by splitting ground truth strips spatially, starting from farthest apart side-by-side strokes and randomly growing either one the other seed by adding the closest side-by-side stroke, until the strip is fully partitioned. We use pairs of ground truth strips as negative training examples.

Test Set

We assembled our test set so that it includes drawings we source directly from 12 artists (82 sketches), as well as inputs from two vector sketch benchmarks, from the "Benchmark for Rough Sketch Cleanup" [112] (46 sketches) and OpenSketch [44] (63 sketches). In the latter, sketches of a small number of CAD objects drawn by different designers from different angles. As noted by Yan et al. Yan et al. [112], their "vectorized data has been normalized to have uniform line width". Consequently, "as-is" their data is unrepresentative of artist sketches, since as Chapter 3 note, stroke width plays a major role in viewer perception of sketches. We manually adjusted the width of all strokes in the inputs sourced from Yan et al. [112] to match their provided raster references.

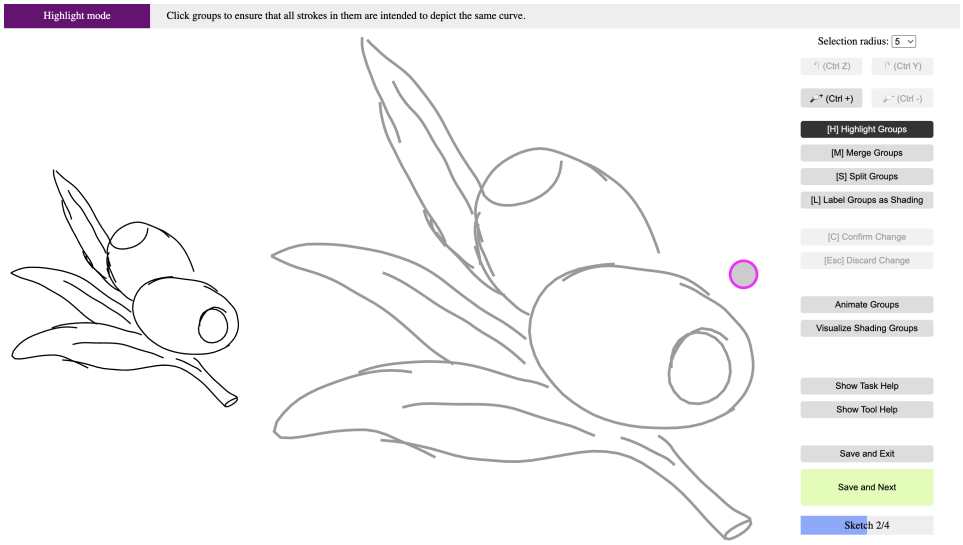


Figure B.3: Our strip annotation interface.

Additional Manually Consolidated Inputs

To validate our consolidation decisions and evaluate ours and alternative methods on unseen data we collected additional manual annotations of 20 complete sketches from 12 different arms-length annotators. The annotated sketches included one

from OpenSketch [44], 3 from Yan et al. [112], and the rest were sourced by us from artists. The sketches were selected as to allow for complete individual sketch annotation in 20 minutes or less. Annotators used the same interface as used for data collection. Annotators took on average 12-15 minutes to annotate each sketch, with up to 30 minutes for larger sketches in the set.

B.3.2 Comparison Setup

Rasterization. To compare our method to raster space approaches we rasterize our inputs using the settings recommended by [112] and Chapter 4, setting the raster resoluton to be 500px along the longest image size and then adding 20px padding to resolve boundary artifacts that otherwise show up in [97, 111]. We use inkscape with the parameter settings of [112]. We noticed that the method of Stanko et al. Stanko et al. [100] sometimes dramatically fails with this anti-aliased setting, and performs better on black and white aliased raster inputs; to accommodate we generated both types of rasterizations and used the better of the two outputs of Stanko et al. throughout all comparisons.

Fitting Curves for StrokeAggregator. Liu et al. propose both methods for stroke clustering into strips and for fitting curves to these strips. Van Mossel et al. Pagurek van Mossel et al. [79] had demonstrated that their new fitting method StrokeStrip outperforms the fitting of StrokeAggregator (Chapter 3). Thus in our comparisons we fit curves to the strips produced by Liu et al. using StrokeStrip. In our experiments StrokeStrip indeed performs better for most inputs; using it to fit both our and Liu's strips allows our quantitative and qualitative comparisons to focus on the differences between our and Liu's stroke clustering approaches.

B.3.3 Comparative Study Design

We conducted a comparative study to evaluate human perceptual preference between our method and representative alternatives StrokeAggregator (Chapter 3), [100, 111].

Each query in this study included an input drawing on top and two consolidated

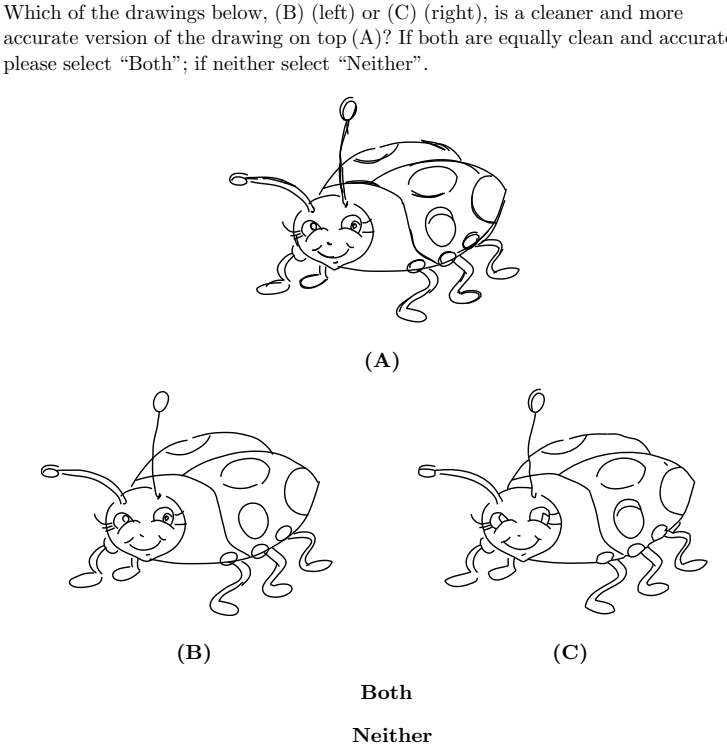


Figure B.4: Study question layout.

outputs below it, presented side-by-side and in random order: one generated by our algorithm, and one generated by an alternative method. The layout of the questions is shown in Fig, B.4. We asked "Which of the drawings below, (B) (left) or (C) (right), is a cleaner and more accurate version of the drawing on top (A)? If both are equally clean and accurate, please select "Both"; if neither select "Neither"." The answer options were "(B)," "(C)," "Both," and "Neither." We used different inputs for each question. We recruited 24 participants (16 male, 8 female), resulting in six responses per question for 90 inputs with diverse authors and styles. 32 inputs were from [112], 22 from [44], and the remaining 36 were inputs commissioned directly from artists.

We followed the study protocol of Chapter 3. Participants were provided a task description and shown a simple reshaping example, both taken from Chapter 3; no other explanation was provided. We use the question from Chapter 3 to discard

answers from participants who did not read the task description. All participants correctly answered this question.