StripMaker: Perception-driven Learned Vector Sketch Consolidation

CHENXI LIU, University of British Columbia, Canada
TOSHIKI AOKI, University of Tokyo, Japan
MIKHAIL BESSMELTSEV, Université de Montréal, Canada
ALLA SHEFFER, University of British Columbia, Canada

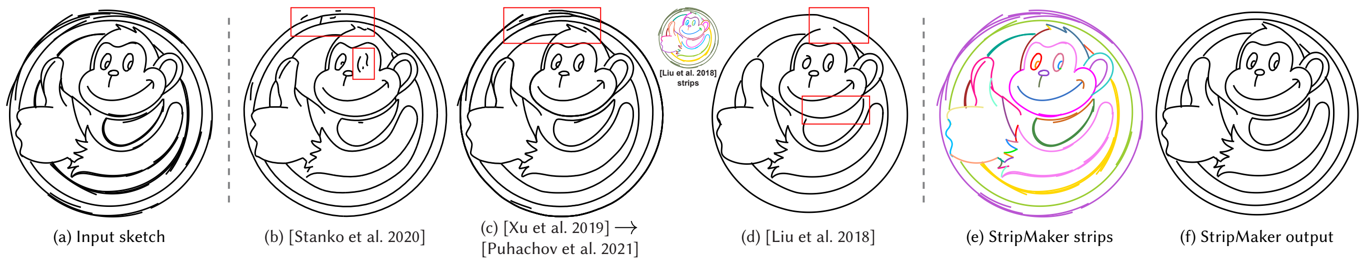


Fig. 1. Given a vector sketch with multiple overdrawn strokes (a) StripMaker automatically consolidates it (f) replacing each detected viewer perceived strip of strokes (e, each strip in different color) with the corresponding intended curve. StripMaker outputs (e) are better aligned with user expectations than those produced by state-of-the-art algorithmic alternatives (b,c,d). Inset in (d) shows [Liu et al. 2018] strips. Frames point to artifacts in outputs of previous methods. Input image © jwalsh under CC-BY-2.0.

Artist sketches often use multiple *overdrawn* strokes to depict a single intended curve. Humans effortlessly mentally *consolidate* such sketches by detecting groups of overdrawn strokes and replacing them with the corresponding intended curves. While this mental process is near instantaneous, manually annotating or retracing sketches to communicate this intended mental image is highly time consuming; yet most sketch applications are not designed to handle overdrawing and can only operate on overdrawing-free, consolidated sketches. We propose *StripMaker*, a new and robust learning based method for automatic consolidation of raw vector sketches. We avoid the need for an unsustainably large manually annotated learning corpus by leveraging observations about artist workflow and perceptual cues viewers employ when mentally consolidating sketches. We train two perception-aware classifiers that assess the likelihood that a pair of stroke groups jointly depicts the same intended curve: our first classifier is purely local and only accounts for the properties of the evaluated strokes; our second classifier incorporates global context and is designed to operate on approximately consolidated sketches. We embed these classifiers within a consolidation framework that leverages artist workflow: we first process strokes in the order they were drawn and use our local classifier to arrive at an approximate consolidation output, then use the contextual classifier to refine this output and finalize the consolidated result. We validate StripMaker by comparing its results to manual consolidation outputs and algorithmic alternatives. StripMaker achieves comparable performance to manual consolidation. In a comparative study participants preferred our results by a 53% margin over those of the closest algorithmic alternative (67% versus 14%, other/neither 19%).

CCS Concepts: • **Computing methodologies → Image manipulation**.

Authors' addresses: Chenxi Liu, chenxil@cs.ubc.ca, University of British Columbia, Vancouver, Canada; Toshiki Aoki, aoki-toshiki1127@g.ecc.u-tokyo.ac.jp, University of Tokyo, Tokyo, Japan; Mikhail Bessmeltsev, bmpix@iro.umontreal.ca, Université de Montréal, Montréal, Canada; Alla Sheffer, sheffa@cs.ubc.ca, University of British Columbia, Vancouver, Canada.

Additional Key Words and Phrases: Vector graphics, sketch consolidation, line art

ACM Reference Format:
Chenxi Liu, Toshiki Aoki, Mikhail Bessmeltsev, and Alla Sheffer. 2023. StripMaker: Perception-driven Learned Vector Sketch Consolidation. *ACM Trans. Graph.* 42, 4 (August 2023), 15 pages. https://doi.org/10.1145/3592130

1 INTRODUCTION

Free-form sketches are highly effective in communicating artist-intended content, and are frequently used as input to various computer applications. While historically sketches were often drawn on paper, an increasing number are now traced using touch and pen displays and recorded in vector form [Adobe Inc. 2021; Blender 2022; van Mossel et al. 2021; Yin et al. 2022]. While artists often seek to use their raw sketches as input to various computer applications, their drawings often do not conform to the input requirements of these applications [Yan et al. 2020]. In particular, while artists frequently employ groups of tightly drawn strokes to depict individual intended curves (Fig. 1a), sketch processing software typically expects each intended curve to be depicted using a single stroke. *Consolidating* sketches by replacing groups of overdrawn strokes (Fig. 1e) by their corresponding intended curves produces clean, and more aesthetic, versions of the original sketches (Fig. 1f) that are well-suited for downstream applications.

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 [Liu et al. 2018]. While a range of attempts have been made to automate consolidation of both vector [Liu et al. 2018, 2015] and raster [Stanko et al. 2020; Xu et al. 2019] sketches (Sec. 2), existing consolidation algorithms frequently fail to produce viewer-expected results (Fig. 1b-d). We propose a new vector sketch consolidation method that produces outputs significantly better aligned

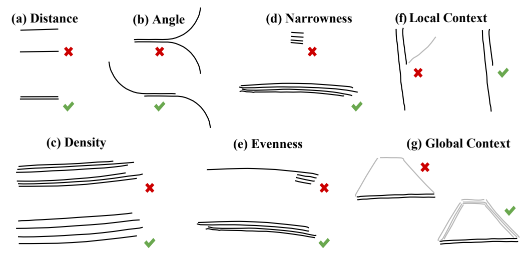


Fig. 2. Consolidation cues: Locally groups of strokes are seen as belonging to the same strip if they are proximate (a), roughly parallel (b) , and approximately evenly spaced (c). Strips are expected to be narrow (d) and have roughly even width throughout (e). Local (f) and global (g) context impacts strip perception.

with viewer expectations than those produced by these alternatives. We focus on vector inputs since, as noted above, 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 (Fig. 1e) and fitting the best corresponding curve to each such group (Fig. 1f). We focus on the clustering task, and use the state-of-the-art method of [van Mossel et al. 2021] 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 (Sec. 3) 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 (Fig. 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 [Liu et al. 2018; van Mossel et al. 2021] and use these correspondences to assess the compatibility between them (Fig. 2a-e). Our analysis suggests that viewers are impacted by the presence of actual or viewer-perceived intersections between the assessed and neighboring strokes (Fig. 2f). Lastly, we speculate that viewers are more likely to see farther apart strokes as belonging together, if the drawings appear less accurate overall, and to group strokes less aggressively given drawings which appear more neat (Fig. 2g). Still, it remains unknown how viewers measure or balance the different factors, or cues, involved.

A potential approach for addressing perception motivated tasks with similar unknowns is to learn the viewer desired outcomes from manually annotated data [Yin et al. 2022]. 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 Supp. 2.3) 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: our first *temporal* consolidation step uses purely local properties to obtain an approximate, or *preliminary* consolidation; our second step *refines* this preliminary outcome by combining local cues with contextual and global features computed using preliminary strips (Sec. 4).

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 (Sec. 5). 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 [Liu et al. 2018; van Mossel et al. 2021]. 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 (Sec. 4.1).

We refine the resulting preliminary consolidation by reevaluating 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 (Sec. 4.2). 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 (Sec. 7). 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. An additional advantage of our method over earlier techniques is that our temporal consolidation step can be run interactively as the artist adds strokes and only the refinement needs to be performed at the end. This makes our framework well suited for interactive settings.

2 RELATED WORK

Vector Sketch Processing. Multiple tools target processing of artist sketches, facilitating tasks such as colorization [Adobe Inc. 2021], shading [Finch et al. 2011; Shao et al. 2012], beautification [Baran et al. 2010; Cheema et al. 2012; Fišer et al. 2016] editing [Igarashi et al. 2005] and 3D inference [Lipson and Shpitalni 1996; Xu et al. 2014]. Most of these tools are designed for processing consolidated or overdrawing-free vector sketches whose strokes correspond to intended, meaningful curves [Yan et al. 2020]. By facilitating automatic consolidation, we enable a seamless workflow where these tools can be directly used on raw sketches (Fig. 11).

Topology Cleanup. Topological cleanup methods correct imprecise raster or vector sketch connectivity and connect strokes that are intended to intersect but stop short of doing so [Asente et al. 2007; Fourey et al. 2018; Jiang et al. 2021; Wang et al. 2020; Yin et al. 2022]. Vector-space connectivity extraction methods [Asente et al. 2007; Jiang et al. 2021; Yin et al. 2022] target overdrawing-free sketches, and require a consolidation pre-process to be applied to raw sketches. Yet, topology cleanup and consolidation are interconnected — human decisions regarding which strokes define the same intended curves are impacted by viewer-perceived inter-stroke connectivity [Liu et al. 2015]. We successfully account for this interconnectedness when making consolidation decisions (Sec. 4.2).

Raster Sketch Consolidation and Vectorization. Multiple methods simultaneously consolidate and vectorize raster sketches [Chen et al. 2018; Egiazarian et al. 2020; Favreau et al. 2016; Kim et al. 2018; Mo et al. 2021; Parakkat et al. 2021, 2018; Stanko et al. 2020] either automatically or semi-interactively. Alternatively, one can first consolidate these sketches in raster space either automatically [Simo-Serra et al. 2018a, 2016; Xu et al. 2019] or interactively [Simo-Serra et al. 2018b] and then vectorize them using vectorization methods designed for overdrawing-free sketches [Bessmeltsev and Solomon 2019; Bhunia et al. 2021; Das et al. 2021; Donati et al. 2017, 2019; Guo et al. 2019; Puhachov et al. 2021]; see, e.g., Xu et al. [2022] for a recent survey. Raster CAD sketches can be consolidated using the domain specific method of [Manda et al. 2022].

While one can potentially consolidate vector sketches by rasterizing them first and then using one of these workflows, this approach faces major challenges. First, as recently acknowledged by Parakkat et al. [2021], the problem of automatically vectorizing

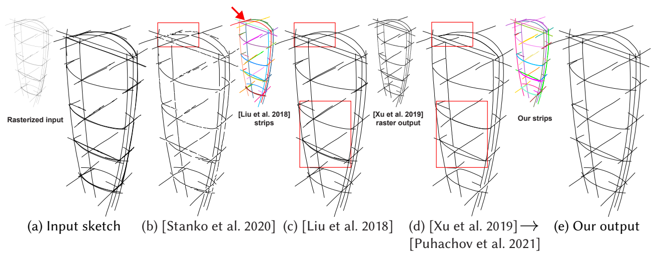


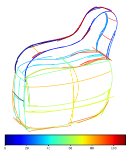
Fig. 3. Consolidating typical inputs (a) using state-of-the-art methods for simultaneous consolidation and vectorization [Stanko et al. 2020] (b), and vector [Liu et al. 2018] (c) and raster [Xu et al. 2019] (d) space consolidation, often results in both loss of details and under-consolidation. Rasterized input used for (b) and (d) shown as inset in (a). The raster output of [Xu et al. 2019] (shown in the inset in (d)) was vectorized using the method of [Puhachov et al. 2021]. Our method (e) produces viewer expected consolidations on these inputs. Please zoom-in to see details. Input image from [Gryaditskaya et al. 2020].

raster sketches, while simultaneously consolidating them, remains unsolved. Yan et al. [2020] reach a similar conclusion for both raster space workflows (raster-to-vector and raster-to-raster-to-vector). 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 [Yan et al. 2020]. Figs. 3 and 1 compare our method against state of the art approaches using these workflows ([Xu et al. 2019] and [Stanko et al. 2020]); see Appx. D.1 for rasterization details. Additional comparisons to these and other raster space methods [Favreau et al. 2016; Mo et al. 2021; Parakkat et al. 2021] are included in Sec. 7 and the supplementary. As these examples show, StripMaker is consistently better at preserving fine details often destroyed by the raster-space methods (e.g. intersections in Fig. 3) and correctly consolidates overdrawn content these methods leave as-is (e.g. outline in Fig. 1). In our comparative study (Sec. 7), viewers preferred our outputs over those of the methods of [Xu et al. 2019] and [Stanko et al. 2020] by margins of 57% and 78% respectively.

Vector Sketch Consolidation. Vector sketch consolidation involves solving two separate problems: locating strips, or groups of strokes perceived as depicting single curves, and fitting the desired curves to each strip [Liu et al. 2018, 2015; Orbay and Kara 2011]. While successful solutions to the strip fitting problem exist, e.g. [Liu et al. 2018; van Mossel et al. 2021], clustering of strokes into strips remains an open problem [Yan et al. 2020]. We therefore focus our efforts on stroke clustering and fit curves to our output strips using the method of van Mossel et al. [2021].

Early consolidation methods compared features such as proximity and degree of parallelism between strokes against user specified per-input thresholds to determine which strokes belong to the same strip [Bao and Fu 2012; Barla et al. 2005; Chen et al. 2013; Nan et al. 2011; Rosin 1994; Shesh and Chen 2008]. Gryaditskaya et al. [2020] and Yin et al. [2022] similarly employ proximity and tangent similarity thresholding to obtain approximate, conservative consolidations; as they acknowledge these features alone are not sufficient to accurately predict intended stroke grouping.

Orbay and Kara [2011] use per-artist learned thresholds on nearest point distance and tangent difference to obtain initial strips. They then algorithmically refine these strips using hardcoded parameters; as the inset shows this approach dramatically over-merges strokes (each strip colored in a different color).

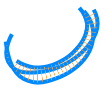


Several drawing systems consolidate sketches on-the-fly, deciding for each newly drawn stroke whether it should be grouped with some of the previously drawn ones [Bae et al. 2008; Baudel 1994; Grimm and Joshi 2012] based on spatial and *temporal* proximity. Noris et al. [2012] leverage drawing order as a consolidation cue and incorporates user input when faced with ambiguous drawings. Our analysis of manually consolidated inputs shows that around 30% of viewer perceived multi-stroke strips contain strokes which were not drawn consecutively, see typical example on the right (color represents drawing order). StripMaker uses drawing order to guide consolidation workflow, and uses other perception motivated features to determine which strokes to consolidate.

Liu et al. [2018] and Liu et al. [2015] use a combination of local and global cues to guide their consolidation strategies, and rely on either user-specified or fixed parameters in their decision process. Figs. 1, 3 compare our results against those of Liu et al. [2018]; additional comparisons against both methods are included in Sec. 7. Qualitative and quantitative comparisons (Sec. 7) demonstrate that StripMaker produces outputs better aligned with viewer expectations than these methods. We achieve this improvement by leveraging a combination of a learning-based strategy and an efficient computation workflow based on sketch drawing order.

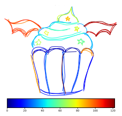
3 ANALYSIS OF OVERDRAWN SKETCHES

Professional and amateur artists often depict intended curves using strips of overdrawn strokes [Arora et al. 2017; Eissen and Steur 2008; Yan et al. 2020]. 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 (Fig. 2). We leverage those cues in our algorithm (Sec. 4).

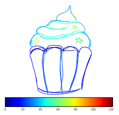


Correspondence. At its core, consolidation merges together groups of strokes that are fully or partially *side-by-side* , or next to one another. Research suggests that when making consolidation decisions, viewers rely on implicit *dense correspondence* (orange isolines in inset) between side-by-side stroke sections when evaluating the degree of compatibility between them [Liu et al. 2018], and expect each strip to allow for a *low distortion 1D parameterization* and be well approximated by a single curve [van Mossel et al. 2021].

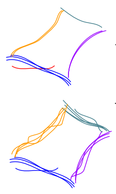
Local Geometry. Application of Gestalt psychology grouping principles [Koffka 1955; Wagemans et al. 2012] to strokes suggests that *parallelism*, *distances*, and *density* play a major role in consolidation decisions (Fig. 2a-c). Viewers are more likely to group strokes which are more parallel and closer 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 Fig. 2c. In addition, Liu et al. [2018] demonstrate that viewers expect strips to be *narrow*, having a small width to length ratio (Fig. 2d). We further observe that viewer perceived strips typically have roughly the same, or *even*, width throughout with no drastic changes (Fig. 2e).



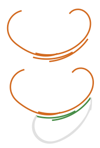
Drawing Order. Our analysis of sketch drawing order confirms observations in prior literature [Grimm and Joshi 2012; Noris et al. 2012] that strokes belonging to the same intended strip are often drawn temporally close to one another, and are frequently drawn consecutively; in the inset the coloring reflects drawing order (bluer strokes are drawn earlier and redder ones later) — 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 inset on the left 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.*



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 (inset top vs bottom). In particular, viewers are likely to incorporate stand-alone, *outlier*, strokes (red in the top inset) into one of their adjacent strips when their surrounding clusters are less precise [Liu et al. 2018].



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 inset the two highlighted groups of strokes look likely to be in the same strip in isolation (top), but are viewed as apart when the gray strip is present (bottom). We refer to this property as *connectivity preservation.*

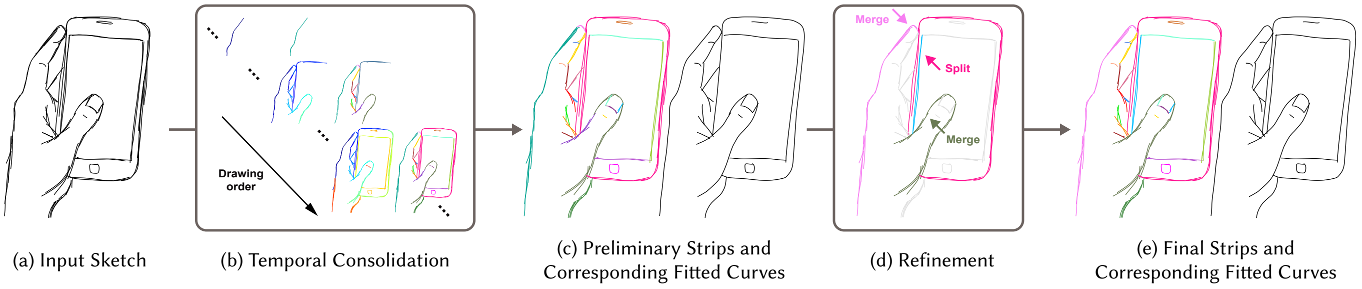

Drawing order
Merge
Split
Merge
(a) Input Sketch
(b) Temporal Consolidation
(c) Preliminary Strips and Corresponding Fitted Curves
(d) Refinement
(e) Final Strips and Corresponding Fitted Curves

Fig. 4. Given an input sketch (a) StripMaker uses a two step process to compute strips (e,left) and fitted curves (e,right) consistent with viewer-expectations. Our first temporal consolidation step (b) leverages drawing order (visualized via stroke color) and local cues to generate preliminary strips (c, fitted curves included for clarity). Our second, refinement step, further improves this preliminary result by leveraging local and global context (d) to arrive at the final desired output (e). Input image © Dave Pagurek van Mossel.

4 ALGORITHM

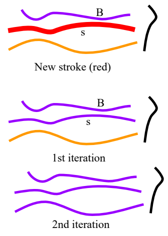
Input. The input to our method is a vector format sketch, where each stroke is a fixed-width curve. We pre-process each curve into an evenly and densely sampled polyline, remove end-point hook artifacts resulting from inadequate device capture [Yin et al. 2022], and break strokes at sharp corners, enabling processing of cases where users use zigzag overdrawing patterns; see Sec. 6 for details.

Workflow. As discussed in Sec. 1 our algorithm is designed to produce consolidations consistent with viewer expectations (Fig. 4e) and to do so efficiently while using a limited size training data set. We achieve these goals by leveraging the observations detailed in Sec. 3 regarding artist workflow and viewer perception. These observations suggest that while correctly clustering strokes often requires global context, many clustering decisions can be made using purely local geometric 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. Moreover, temporal persistence suggests that many such decisions can be accurately made by considering only the current stroke and those drawn before it. Based on these observations we use a two stage workflow (Fig. 4). We first consolidate the inputs using an efficient temporal consolidation method that leverages drawing order and local geometric features of the evaluated strokes and sub-strips (Sec. 4.1, Fig. 4b). While not perfectly accurate, the approximate, preliminary consolidations it computes are close enough to the viewer expected output, enabling us to robustly estimate sketch precision and likely strip connectivity. We use these estimates in our refinement pass generating the final outputs (Sec. 4.2, Fig. 4c). By breaking the computation into two stages we are able to dramatically speed up the consolidation process and overcome training data scarcity. Both stages of our algorithm use pairwise classifiers that assess how likely pairs of sub-strips are to belong to the same viewer-perceived strip. We discuss the design and training of these classifiers in Sec. 5. Please check Sec. 6 for all implementation details.

Output. We fit an aggregate curve to each output strip using the method of van Mossel et al. [2021]. Prior to the fitting, we identify strips that form continuation end-end junctions using a simplified version of the method of [Yin et al. 2022]. We merge strips connected via continuation junctions and fit them jointly. Similar to Liu et al. [2018] we delete single stroke strips which almost completely overlap multi-stroke ones, as these are perceived as noise.

4.1 Local Temporal Consolidation

Our main consolidation step uses a local-feature-based classifier to group the input strokes into preliminary strips (Fig. 4ab). 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.



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 inset. Given the new stroke s (red in the inset), we measure the likelihood that the stroke belongs to one of the previously formed sub-strips using our local classifier (Sec. 5). If the classifier indicates 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 (purple in the inset); 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 (inset, bottom). We repeat the process until there are no sub-strips to combine.

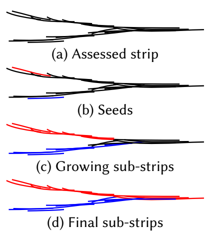
Speed-up. Naively testing new strokes or sub-strips against all previously formed sub-strips at each iteration would imply performing numerous classifier calls, the vast majority of which are likely to provide a negative answer. To gain necessary performance, as a pre-filter to the classifier, we first 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) their side-by-side sections are sufficiently long, (2) they are at least somewhat proximate and weakly parallel, and (3) their joint parameterization has low enough distortion, see Sec. 6 for detailed criteria. These compatibility checks reduce classifier call count and, therefore, runtime by a factor of 7.

Naively 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 leveraging the observation that a stroke s is more likely to belong to the same strip as its temporally previous stroke s_p, than to any other multi-stroke strip. We denote the sub-strip that s_p belongs to as B_p. Given a new stroke s we compute the sub-strip B it is most likely to belong to as per-above. We then distinguish between two cases. If s is deemed most likely to belong to the same sub-strip as s_p ($B = B_p$) we add s to B, delay any new sub-strip comparisons, and proceed to the next stroke in the temporal order. Otherwise, if B_p has more than one stroke, we assess if B_p can be merged with other strips. Specifically, we use our local classifier to evaluate the sub-strip B_p 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, and repeat the iterative evaluation when a merger occurs. This delayed evaluation reduces the number of classifier calls by an order of magnitude.

4.2 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, Sec. 7, 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 balance the global and local classifier choices. On the one hand, 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 hand, 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 reevaluate and split existing strips, when necessary, and to merge adjacent strips that clearly warrant merging.

Strip Reevaluation. We reevaluate each multi-stroke strip taking context into account (see inset). For each strip we select two seed strokes by finding the stroke pair least likely to belong together using our global classifier (Sec. 5). 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 (see inset, b). We mark all non-seed strokes as unassigned, and grow sub-strips incrementally from the two seeds, by adding unassigned strokes to one of the sub-strips (see inset, c), and stopping when all strokes are assigned (see inset, d). 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 reevaluate 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).

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 endpoints, 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. [2022] 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).

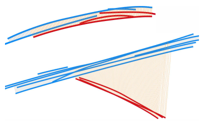
Strip Merging. We merge adjacent strips in the temporal pass output if they are deemed to be part of the same strip by our global classifier and if they pass the connectivity preservation test above. Specifically following the conservative logic above, we merge multi-stroke sub-strips together if the 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 CLASSIFIER DESIGN

At the core of our iterative consolidation pipeline lie two binary Random Forest [Ho 1995] 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 [Grinsztajn et al. 2022; Yin et al. 2022].

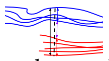
Our local classifier is used in our temporal consolidation step, operates on features that can be computed purely on the evaluated sub-strips, and is trained on sub-strips similar to the ones encountered during temporal consolidation. Our global classifier uses the same set of features with the addition 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 encounter during the refinement step. For additional details on the classifier training corpuses see Appx. C.

Our classifier features are inspired by the analysis of cues observers employ when making consolidation decisions (Sec. 3, Fig. 2).



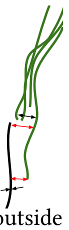
Computing robust features capturing these cues requires point-to-point correspondences between the assessed sub-strips (see inset). We obtain these correspondences via StrokeStrip parameterization [van Mossel et al. 2021]. The parameterization isolines provide a reliable and intuitive pointwise correspondence 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 the inset). We normalize all computed distance by the stroke thickness.

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 the inset below (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).



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 90th and 10th 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.



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 inset) 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 the inset (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 [van Mossel et al. 2021], the quality of this parameterization is in itself an indirect indicator of whether a group can be interpreted as a strip or not. We evaluate the parameterization quality using two properties proposed by [van Mossel et al. 2021]: the deviation of the parameterization tangent length (velocity) from 1, and tangent alignment. 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.

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

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

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 90th 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).

6 ALGORITHM DETAILS

Preprocessing. 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 [Liu et al. 2018; Yin et al. 2022], 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 Liu et al. [2018] 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. 2010] if the tangents at these corners differ by more than 25°.

Postprocessing. 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° [Bessmeltsev et al. 2016; Hess and Field 1999] and the two local strip widths differ by less than 4 times. We detect highly-likely junctions (with probabilities > 90%) using Yin et al. [2022] as intended junctions, 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. [2018], 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 [van Mossel et al. 2021] (see Appx. A for details).

Compatibility Criteria. During our local temporal consolidation, sub-strips are considered non-compatible 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 [2021] in Equation 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 cross-validation dataset statistics.

Strip Reevaluation 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 reevaluation the strip is deemed in need of a split, we reparameterize the sub-strips and reevaluate the split decision.

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 [2011] implementation of random forest and Gini impurity criterion for fitting the training data.

7 RESULTS AND VALIDATION

We tested our method on 191 previously unseen sketches, including 107 sketches from sketch processing benchmarks [Gryaditskaya et al. 2019; Yan et al. 2020] as well as 82 sketches we commissioned from 12 different artists. 16 of these are shown in the paper. The input sourcing and acquisition are detailed in Appendix D. A complete set of inputs and outputs is included in the supplementary materials. Note that for some of the inputs sourced from previous papers, there exist more than one version - in the supplementary we list the source for each of our inputs. These sketches span a vast range of styles and content, and varying degrees of precision form highly sketchy ones such as the hand in Fig. 7 to much more precise ones, e.g. the printer in Fig. 6. 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 in isolation, 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).

Table 1. Average L_1 and $L_{\max}$ distances to consolidations generated using manual labelings. Result on our cross-validation set (left); results on unseen annotation set (middle); results on manual consolidations collected by [Yan et al. 2020] (right). Our method achieves the best performance among all algorithms tested, approaching human performance.

	Cross Validation		Human Annotation		Human Consolidation [Yan et al. 2020]	
	L_1	$L_{\max}$	L_1	$L_{\max}$	L_1	$L_{\max}$
Human	-		0.548	12.924	1.957	30.538
Stanko'20	2.252	11.668	1.574	19.075	2.257	42.017
Xu'19	1.106	10.293	1.201	11.617	2.448	42.528
Liu'18	0.287	5.629	0.920	14.477	2.844	44.164
Our temporal consolidation	0.179	3.409	0.708	11.598	2.511	41.862
Our final	**0.149**	**3.141**	**0.645**	**10.353**	**2.167**	**41.358**

Our consolidation pippeline calls the classifiers dozens or even hundreds of times on each sketch. In theory, a single classifier failure can cause a chain reaction of errors. We confirm that this is not the case by measuring the distance between our consolidation outputs and those produced using manual ground-truth annotations. To perform this experiment we similarly left one sketch out, trained both classifiers on the remaining sketches and then used those within our algorithm pipeline to consolidate the left-out sketch. We then measured the image space (L_1 and $L_{\max}$) distance between our fitted outputs and those produced using ground truth annotations (Tab. 1, left). Our average L_1 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 temporal step (0.179). Using distances to assess consolidation quality enables us to evaluate diverse methods via the same metric and is motivated by [Yan et al. 2020].

Additional Comparisons to Manual Consolidation. We compare our consolidation outputs to manual consolidations on additional unseen sketches from two different sources. We first 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 (Tab. 1, middle). 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.

We also compare our results against pre-existing manually consolidated outputs collected by [Yan et al. 2020] (Tab. 1, right). In their dataset three artists directly sketched what they perceived as clean versions of each input raw sketch. As shown in the table and illustrated in Fig. 5 participant agreement on this data is much lower as different artists often introduce different additional cleaning operations such as removing strokes they perceive as redundant,

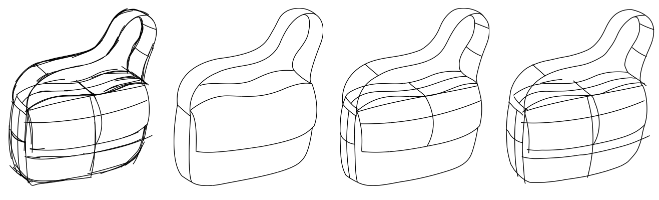


Fig. 5. Comparison against manually cleaned sketch (a) from [Yan et al. 2020] dataset. Notably when cleaning sketches artists go beyond replacing strips with corresponding curves and often delete what they see as redundant strokes and trim or extend strokes to produce clean junctions (b-d). Our method produces visually similar consolidations (e) but does not perform these additional tasks. Input image from [Favreau et al. 2016].

or trimming and extending strokes to form clean junctions. Nevertheless the distances between our results and closest artist ones are similar to those between artists. More importantly, as discussed below our results are numerically closer to artist ones than those produced by all alternative methods.

Comparison Against Prior Art. We compare our method against prior art in three related categories: raster-space consolidation, simultaneous consolidation and vectorization, and vector-space consolidation. We focus our comparisons on the latest or best performing automatic methods in each category [Liu et al. 2018; Stanko et al. 2020; Xu et al. 2019] (Figs. 1, 3, 6, 7). Comparisons to additional methods [Favreau et al. 2016; Liu et al. 2015; Mo et al. 2021; Parakkat et al. 2021; Simo-Serra et al. 2018a] are shown in Figs. 9 and 10. To apply the raster space methods to our data we rasterize our inputs as discussed in Appx. D.1. As these figures and additional comparisons shown in the supplementary demonstrate, our method visibly significantly outperforms all earlier approaches.

We measure mean and maximal distances between the outputs of these methods and our three 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 performance across both vector and raster space methods. To enable the most fair comparison we re-fit the strips produces by [Liu et al. 2018] using our fitting method (see Appx. D.1 for details). As reported in Tab. 1 on both the cross-validation and manual annotation datasets the distances for all the methods we compare to are at least 30% higher than those achieved by StripMaker. StripMaker outperforms the previous methods when compared to the manually cleaned sketches of Yan et al. [2020], however as discussed above and illustrated in Fig. 5 the numbers in this case are impacted by additional cleaning tasks artists may have chosen to perform.

We compare the perceptual accuracy of our method against the prior approaches of [Liu et al. 2018; Stanko et al. 2020; Xu et al. 2019] via a comparative study (Appx. D). 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 collected answers to 90 questions (30 per method), 6 answers per question (540 answers in total). Fig. 8 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. Fig. 12 shows three inputs where viewers preferred the alternative over StripMaker (alternative methods were preferred on 6 out of 90 inputs shown).

Consolidation Applications. The consolidated results produced by our method can be directly processed by downstream applications as illustrated in Fig. 11. In particular, while applying topology cleanup [Yin et al. 2022] directly to a typical input (Fig. 11a) produces numerous undesirable tiny regions (55% regions on this input) (Fig. 11b), consolidating the input using StipMaker and then applying the method of Yin et al. [2022] produces the viewer expected topology. Thus combining these two methods can facilitate colorization of the consolidated sketches (Fig. 11d). Our output (Fig. 11f) can be used to facilitate strip-level sketch manipulations, such as recoloring with gradient based shading (Fig. 11g).

7.1 Ablations

Runtimes. Our median runtime across all inputs in our test set is 50 seconds per sketch, 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.

Performance with Different Sets of Features. We experiment 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. The full Gini coefficients of our classifiers are included in Appx. B. We also experimented with adding the temporal distance between sub-strips as a classifier feature, performance did not improve.

Classifier Choice. We experimented with replacing random forest classifiers with multilayer perceptron (MLP) classifiers. We determined their hyper-parameters via a grid-search, and used the same cross-validation as above. The difference of cross-validation accuracy between MLP and Random Forest is under 1%. We choose to use Random Forest, because it is both simpler and reveals feature importance as a by-product.

Temporal Persistence. Temporal persistence is one of the important components of our method. The first stage of our method follows the temporal drawing order of the sketch and applies a specially designed strategy to reduce the number of classifier calls as described in Sec. 4.1. We validate this design choice by conducting two experiments on our cross-validation set. First, we compare the number of classifier calls with and without the proposed speedup strategy. Using our strategy reduces the number of calls by a factor

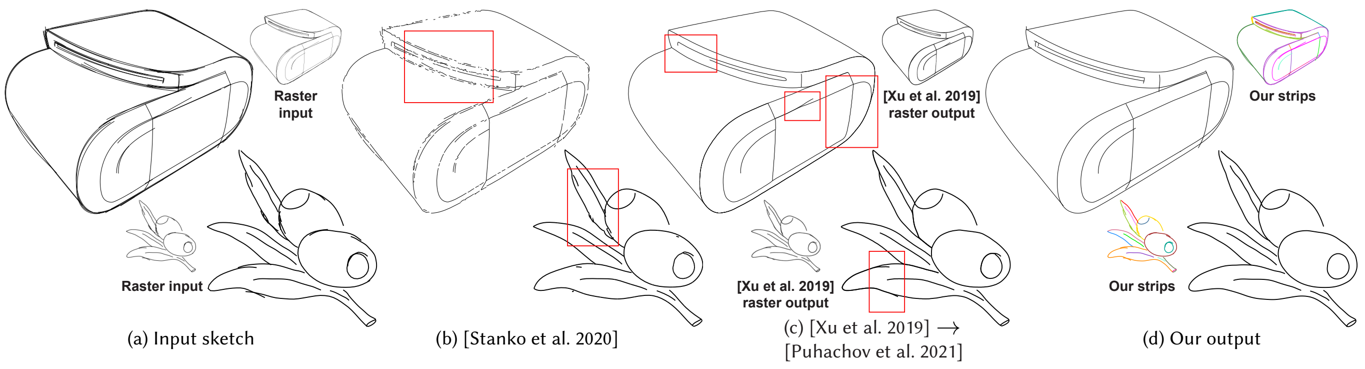

Raster input
Raster input
(a) Input sketch
(b) [Stanko et al. 2020]
[Xu et al. 2019] raster output
[Xu et al. 2019] raster output
(c) [Xu et al. 2019] ⟶ [Puhachov et al. 2021]
Our strips
Our strips
(d) Our output

Fig. 6. Consolidating typical inputs (a) using raster-space methods (b) [Stanko et al. 2020] (c) [Xu et al. 2019] (vectorized using the method of [Puhachov et al. 2021]) 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. Top input image © Akshay Sharma under CC-BY-SA. Bottom input image © Rami Alsafadi.

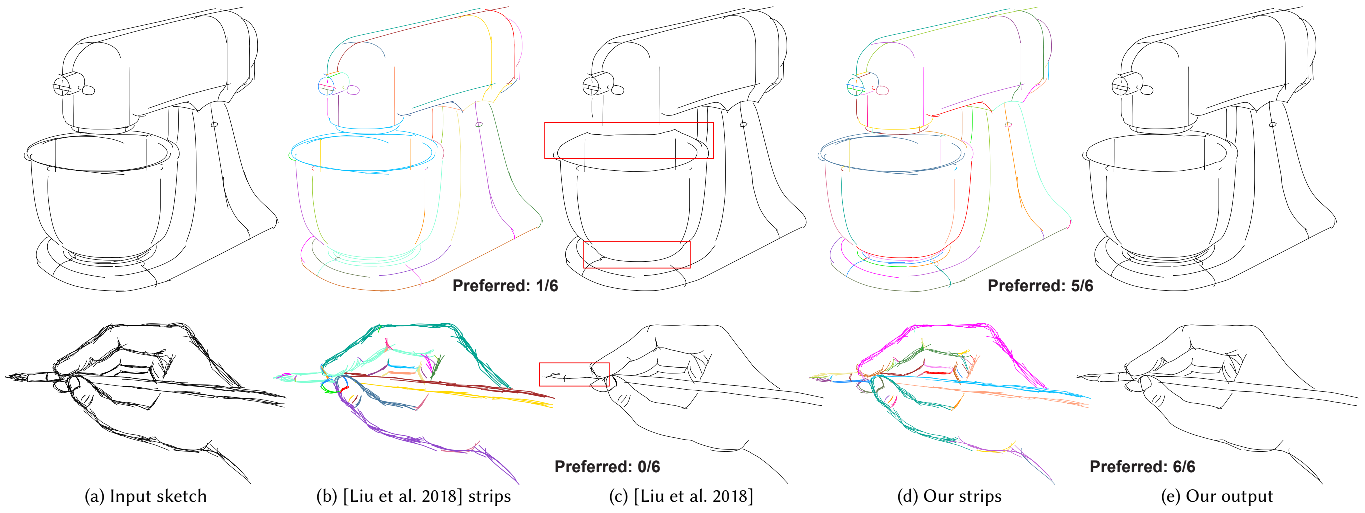


Fig. 7. Our method (d,e) consistently produces consolidations better aligned with viewer expectations than those produced by the state of the art vector consolidation approach of [Liu et al. 2018] (b,c) on diverse overdrawn inputs (a). Stroke grouping is shown with each strip rendered in a different color (b,d). Top input image from [Gryaditskaya et al. 2019]. Bottom input image © Tina Nowarre.

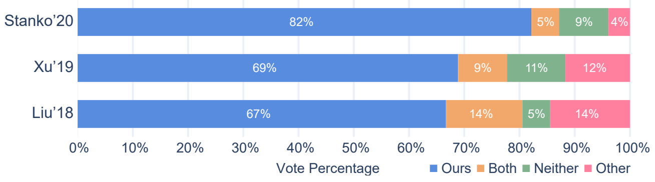


Fig. 8. Comparative study summary: Participants preferred our results over all alternatives by a significant margin.

five. Second, we randomly order the strokes in the input and apply our algorithm. This change increases runtime by 50% and increases distance to manually annotated ground truth on cross-validation set by 43% (L_1: 0.214; $L_{\max}$: 4.833). Notably, these numbers are still better than those obtained via alternative methods (Tab. 1).

7.2 Limitations

Our performance is constrained by data scarcity which prevents greater reliance on context, due to overfitting concerns. Consequently our method primarily relies on local geometric cues and is based around pairwise sub-strip classifiers which do not directly account for input sub-strip interaction with other strokes. Thus, it works best on inputs where local cues are highly predictive of the consolidation outcomes, which is the case for typical drawings. At the same time, as exemplified in Fig. 12, our method can fail to refine the preliminary results when there are only few strips in the sketch and the majority of them are incorrectly consolidated in the preliminary step. Overall, across all inputs in our perception study, such examples were exceedingly rare. Viewers preferred the alternative method's results over ours on only 4 out of 30 inputs in comparisons against [Liu et al. 2018], 3 out of 30 in comparisons against [Xu et al. 2019], and 1 out of 30 in comparisons against [Stanko et al.

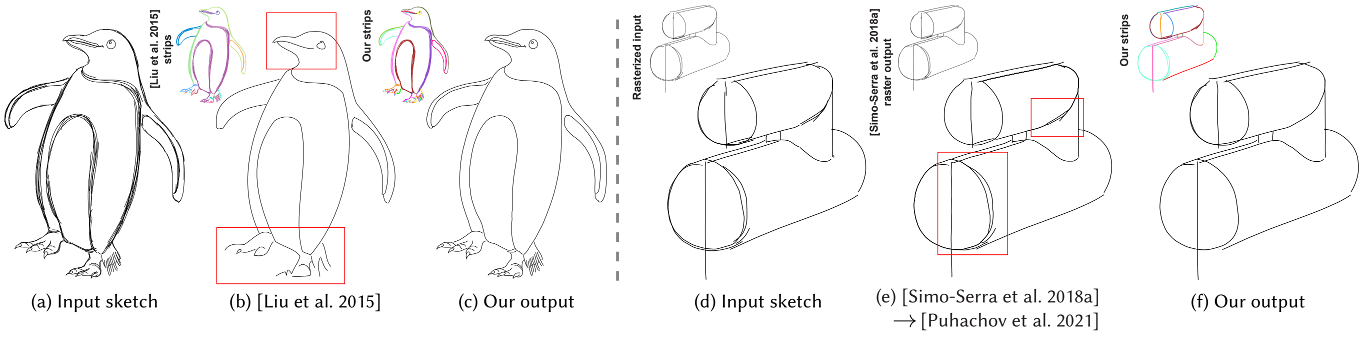


Fig. 9. Earlier sketch consolidation methods, such as [Liu et al. 2015] (left) and [Simo-Serra et al. 2018a] (right) often fail to adequately consolidate typical sketches (a,d) that our method succeeds on (c,f). On the left we used classifiers trained excluding the input shown (we have some results of [Liu et al. 2015] but no access to their code). Left input image © Enrique Rosales. Right input image from [Gryaditskaya et al. 2019].

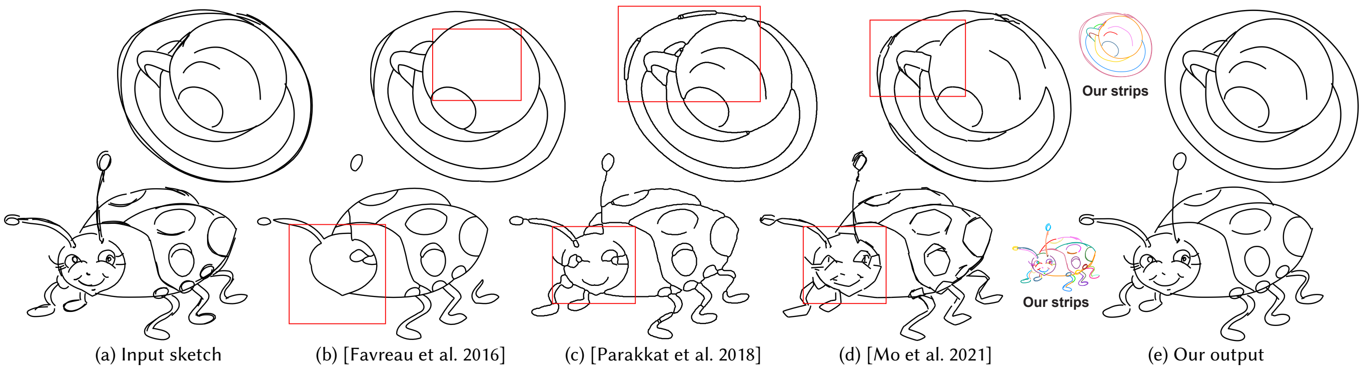


Fig. 10. Comparison to simultaneous consolidation and vectorization methods: (b) [Favreau et al. 2016], (c) [Parakkat et al. 2018], (d) [Mo et al. 2021]. These methods fail to generate viewer-expected outputs when applied to rasterizations of typical overdrawn vector sketches (a). Our method (e) produces viewer expected results on this data. Top input image © Val Novikov. Bottom input image © Rami Alsafadi.

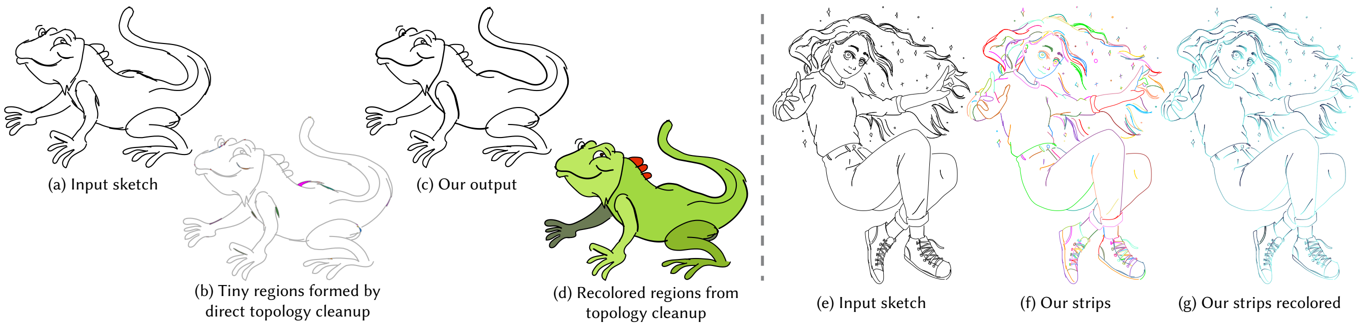

(a) Input sketch
(b) Tiny regions formed by direct topology cleanup
(c) Our output
(d) Recolored regions from topology cleanup
(e) Input sketch
(f) Our strips
(g) Our strips recolored

Fig. 11. Consolidation applications: Applying topology cleanup [Yin et al. 2022] directly to a typical input (a) produces numerous undesirable tiny regions (55% regions on this input) (b). Consolidating the input with our method produces the viewer expected topology facilitating colorization (d). Our output strips (f) facilitate per-strip manipulations, such as gradient-based recolorization with gradient (g). Left input image © Rami Alsafadi. Right input image © Maria Fiddler (aka Maria Hegedus) under CC-BY-NC-SA-4.0.

2020]. Incorporating more contextual cues and detecting similarities between strips within the same input sketch could potentially solve this issue and is an important future research avenue.

8 CONCLUSIONS

We presented a novel vector sketch consolidation method that notably outperforms existing alternatives. Our method is the first to use a principled classification-based approach to vector sketch consolidation. We identified and modeled a variety of perceptual cues

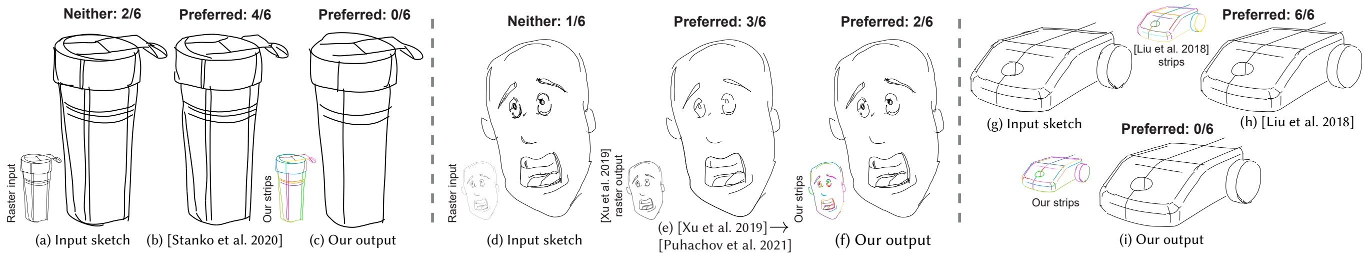


Fig. 12. Limitations: While significantly outperforming other methods overall, study participants prefered the outputs of the alternative methods over StripMaker outputs on the inputs shown. Left input image © Val Novikov. Middle input image © Champ Semalulu. Right input image from [Gryaditskaya et al. 2019].

that are novel for this task, such as evenness and temporal persistence. We leveraged these cues to design perception aware classifiers that reliably predict whether a pair of sub-strips belongs to the same intended strip. Our results can likely be further improved via more tight integration with topological cleanup. An interesting avenue for future work is to incorporate our method into an online sketching system designed to unobtrusively correct overdrawing on the fly as users add new strokes. Our identified perceptual cues are essential to consolidation and can be used by future sketch processing methods.

ACKNOWLEDGMENTS

We thank the study participants, the artists, Chuan Yan, Yulia Gryaditskaya and other previous work authors for making their data available, Silver Burla and Dave Pagurek van Mossel for helping with data collection, Nicholas Vining for paper editing, and the reviewers for their helpful suggestions. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2019-05097 ("Creating Virtual Shapes via Intuitive Input"), NSERC grant RGPIN-2018-03944 ("Broad-Based Computational Shape Design"), and the NSERC - the Fonds de recherche du Québec - Nature et technologies (FRQNT) NOVA Grant No. 314090.

REFERENCES

Adobe Inc. 2021. *Adobe Illustrator*. https://adobe.com/products/illustrator

Rahul Arora, Ishan Darolia, Vinay P. Namboodiri, Karan Singh, and Adrien Bousseau. 2017. SketchSoup: Exploratory Ideation Using Design Sketches. *Computer Graphics Forum* (2017).

Paul Asente, Mike Schuster, and Teri Pettit. 2007. Dynamic Planar Map Illustration. *ACM Trans. Graph.* 26, 3 (2007), 10 pages.

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: As-natural-as-possible Sketching System for Creating 3D Curve Models. In *Proc. UIST*. 151–160.

B. Bao and H. Fu. 2012. Vectorizing Line Drawings with Near-Constant Line Width. In *2012 19th IEEE International Conference on Image Processing*. 805–808.

Ilya Baran, Jaakko Lehtinen, and Jovan Popović. 2010. Sketching Clothoid Splines Using Shortest Paths. *Comput. Graph. Forum* 29, 2 (2010), 655–664.

Pascal Barla, Joëlle Thollot, and François X. Sillion. 2005. Geometric Clustering for Line Drawing Simplification. In *ACM SIGGRAPH 2005 Sketches (SIGGRAPH '05)*. Association for Computing Machinery, 96–es.

Thomas Baudel. 1994. A Mark-based Interaction Paradigm for Free-hand Drawing. In *Proc. UIST*. 185–192.

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of Line Drawings via Polyvector Fields. *ACM Trans. Graph.* 38, 1 (Jan. 2019), 9:1–9:12.

Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. 2016. Gesture3D: Posing 3D Characters via Gesture Drawings. *ACM Trans. Graph.* 35, 6 (2016).

Ayan Kumar Bhunia, Pinaki Nath Chowdhury, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. 2021. Vectorization and rasterization: Self-supervised learning for sketch and handwriting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5672–5681.

Blender. 2022. Grease Pencil. https://www.blender.org/features/grease-pencil/

Salman Cheema, Sumit Gulwani, and Joseph LaViola. 2012. QuickDraw: Improving Drawing Experience for Geometric Diagrams. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1037–1064.

Jiazhou Chen, Mengqi Du, Xujia Qin, and Yongwei Miao. 2018. An Improved Topology Extraction Approach for Vectorization of Sketchy Line Drawings. *Vis Comput* 34, 12 (Dec. 2018), 1633–1644.

Jiazhou Chen, Gael Guennebaud, Pascal Barla, and Xavier Granier. 2013. Non-Oriented MLS Gradient Fields. *Comput. Graph. Forum* 32, 8 (Aug. 2013), 98.

Ayan Das, Yongxin Yang, Timothy M Hospedales, Tao Xiang, and Yi-Zhe Song. 2021. Cloud2curve: Generation and vectorization of parametric sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7088–7097.

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

Luca Donati, Simone Cesano, and Andrea Prati. 2019. A Complete Hand-Drawn Sketch Vectorization Framework. *Multimed Tools Appl* 78, 14 (July 2019), 19083–19113.

Vage Egiazarian, Oleg Voynov, Alexey Artemov, Denis Volkhonskiy, Aleksandr Safin, Maria Taktasheva, Denis Zorin, and Evgeny Burnaev. 2020. Deep vectorization of technical drawings. In *European Conference on Computer Vision*. Springer, 582–598.

Koos Eissen and Roselien Steur. 2008. *Sketching: Drawing Techniques for Product Designers*. Bis Publishers.

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: A Global Approach to Line Drawing Vectorization. *ACM Trans. Graph.* 35, 4 (July 2016), 120:1–120:10.

Mark Finch, John Snyder, and Hugues Hoppe. 2011. Freeform Vector Graphics with Controlled Thin-plate Splines. *ACM Trans. Graph.* 30, 6 (2011), 166:1–166:10.

Jakub Fišer, Paul Asente, Stephen Schiller, and Daniel Sýkora. 2016. Advanced Drawing Beautification with ShipShape. *Computers & Graphics* 56 (May 2016), 46–58.

Sébastien Fourey, David Tschumperlé, and David Revoy. 2018. *A Fast and Efficient Semi-Guided Algorithm for Flat Coloring Line-Arts.* The Eurographics Association.

Cindy Grimm and Pushkar Joshi. 2012. Just DrawIt: A 3D Sketching System. In *Proc. Symp. on Sketch-Based Interfaces and Modeling*. 121–130.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. 2022. 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.*

Yulia Gryaditskaya, Felix Hähnlein, Chenxi Liu, Alla Sheffer, and Adrien Bousseau. 2020. Lifting Freehand Concept Sketches into 3D. *ACM Trans. Graph.* 39, 6 (2020), 167:1–167:16.

Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Frédo Durand, and Adrien Bousseau. 2019. OpenSketch: A Richly-Annotated Dataset of Product Design Sketches. *ACM Trans. Graph.* 38, 6 (2019), 232:1–232:16.

Yi Guo, Zhuming Zhang, Chu Han, Wenbo Hu, Chengze Li, and Tien-Tsin Wong. 2019. Deep Line Drawing Vectorization via Line Subdivision and Topology Reconstruction. *Computer Graphics Forum* 38, 7 (Oct. 2019), 81–90.

Robert Hess and David Field. 1999. Integration of contours: new insights. *Trends in Cognitive Sciences* 3, 12 (1999), 480–486.

Tin Kam Ho. 1995. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, Vol. 1. 278–282.

Takeo Igarashi, Tomer Moscovich, and John F. Hughes. 2005. As-rigid-as-possible Shape Manipulation. *ACM Trans. Graph.* 24, 3 (2005), 1134–1141.

Jie Jiang, Hock Soon Seah, and Hong Ze Liew. 2021. Handling Gaps for Vector Graphics Coloring. *Vis Comput* 37, 9 (Sept. 2021), 2473–2484.

Byungsoo Kim, Oliver Wang, A. Cengiz Öztireli, and Markus Gross. 2018. Semantic Segmentation for Line Drawing Vectorization Using Neural Networks. *Comput.*

Graph. Forum 37, 2 (2018), 329–338.

K. Koffka. 1955. *Principles of Gestalt Psychology.* Routledge & K. Paul.

H Lipson and M Shpitalni. 1996. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design* 28, 8 (1996), 651 – 663.

Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. StrokeAggregator: Consolidating Raw Sketches into Artist-Intended Curve Drawings. *ACM Trans. Graph.* 37, 4 (July 2018), 97:1–97:15.

Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2015. Closure-Aware Sketch Simplification. *ACM Trans. Graph.* 34, 6 (Oct. 2015), 168:1–168:10.

Bharadwaj Manda, Prasad Pralhad Kendre, Subhrajit Dey, and Ramanathan Muthuganapathy. 2022. SketchCleanNet—A deep learning approach to the enhancement and correction of query sketches for a 3D CAD model retrieval system. *Computers & Graphics* 107 (2022), 73–83.

Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. 2021. General Virtual Sketching Framework for Vector Line Art. *ACM Trans. Graph.* 40, 4 (July 2021), 51:1–51:14.

Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. 2011. Conjoining gestalt rules for abstraction of architectural drawings. *ACM Transactions on Graphics (TOG)* 30, 6 (2011), 1–10.

G. Noris, D. Sýkora, A. Shamir, S. Coros, B. Whited, M. Simmons, A. Hornung, M. Gross, and R. Sumner. 2012. Smart Scribbles for Sketch Segmentation. *Comput. Graph. Forum* 31, 8 (Dec. 2012), 2516–2527.

G. Orbay and L. B. Kara. 2011. Beautification of Design Sketches Using Trainable Stroke Clustering and Curve Fitting. *IEEE Trans. Vis. Comput. Graph.* 17, 5 (May 2011), 694–708.

Amal Dev Parakkat, Marie-Paule Cani, and Karan Singh. 2021. Color by numbers: Interactive structuring and vectorization of sketch imagery. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* 1–11.

Amal Dev Parakkat, Uday Bondi Pundarikaksha, and Ramanathan Muthuganapathy. 2018. A Delaunay triangulation based approach for cleaning rough sketches. *Computers and Graphics* 74 (2018), 171–181.

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. 2011. Scikit-Learn: Machine Learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.

Ivan Puhachov, William Neveu, Edward Chien, and Mikhail Bessmeltsev. 2021. Keypoint-Driven Line Drawing Vectorization via PolyVector Flow. *ACM Trans. on Graph. (Proc. of SIGGRAPH Asia)* 40, 6 (12 2021).

Paul Rosin. 1994. Grouping Curved Lines. (1994).

Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: shading concept sketches using cross-section curves. *ACM Trans. Graph.* 31, 4 (2012), 45:1–45:11.

Amit Shesh and Baoquan Chen. 2008. Efficient and Dynamic Simplification of Line Drawings. *Comput. Graph. Forum* 27, 2 (2008), 537–545.

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018a. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Trans. Graph.* 37, 1 (2018), 11:1–11:13.

Edgar Simo-Serra, Satoshi Iizuka, and Hiroshi Ishikawa. 2018b. Real-Time Data-Driven Interactive Rough Sketch Inking. *ACM Trans. Graph.* 37, 4 (2018), 98:1–98:14.

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup. *ACM Trans. Graph.* 35, 4 (2016), 121:1–121:11.

Tibor Stanko, Mikhail Bessmeltsev, David Bommes, and Adrien Bousseau. 2020. Integer-Grid Sketch Simplification and Vectorization. *Computer Graphics Forum (Proc. SGP)* 39, 5 (7 2020).

Dave Pagurek van Mossel, Chenxi Liu, Nicholas Vining, Mikhail Bessmeltsev, and Alla Sheffer. 2021. StrokeStrip: Joint Parameterization and Fitting of Stroke Clusters. *ACM Trans. Graph.* 40, 4 (July 2021), 50:1–50:18.

J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R von der Heydt. 2012. A Century of Gestalt Psychology in Visual Perception I. Perceptual Grouping and Figure-Ground Organization. *Psychological Bulletin* 138, 6 (2012), 1172–1217.

Shuxia Wang, Qian Zhang, Shouxia Wang, Xiaoke Jing, and Mantun Gao. 2020. Endpoint Fusing Method of Online Freehand-Sketched Polyhedrons. *Vis Comput* 36, 2 (Feb. 2020), 291–303.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D Curve Networks from 2D Sketches via Selective Regularization. *ACM Trans. Graph.* 33, 4 (July 2014), 131:1–131:13.

Peng Xu, Timothy M Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. 2022. Deep learning for free-hand sketch: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

Xuemiao Xu, Minshan Xie, Peiqi Miao, Wei Qu, Wenpeng Xiao, Huaidong Zhang, Xueting Liu, and Tien-Tsin Wong. 2019. Perceptual-aware sketch simplification based on integrated VGG layers. *IEEE transactions on visualization and computer graphics* 27, 1 (2019), 178–189.

Table 2. Gini importances of local classifier features.

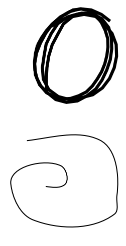
average angle	0.072
median angle	0.075
average distance	0.162
median distance	0.105
density avg distance to Max Width	0.058
density median distance2LocalMaxWidth	0.062
density avg distance2LocalMinWidth	0.146
density median distance2LocalMinWidth	0.073
density avg distance2width combined	0.066
densitymedian 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

Chuan Yan, David Vanderhaeghe, and Yotam Gingold. 2020. A Benchmark for Rough Sketch Cleanup. *ACM Trans. Graph.* 39, 6 (Nov. 2020).

Jerry Yin, Chenxi Liu, Rebecca Lin, Nicholas Vining, Helge Rhodin, and Alla Sheffer. 2022. Detecting Viewer-Perceived Intended Vector Sketch Connectivity. *ACM Transactions on Graphics* 41 (2022). Issue 4.

A ELLIPSE FITTING

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 ellipses (inset, top) and intentional spirals (inset, bottom) 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 van Mossel et al. [2021]. 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 $g < 3w$.



B GINI COEFFICIENTS

Tab. 2 and 3 report the Gini coefficients of our classifiers.

C DATA COLLECTION

C.1 Training data corpus

We use 66 manually annotated sketches generated by the authors of van Mossel et al. [2021] for testing curve fitting to strips, as our training data. These sketches are sourced from multiple prior publications, including [Liu et al. 2018, 2015; Orbay and Kara 2011] and different artists.

Table 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

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. 4.1, 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.

C.2 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" [Yan et al. 2020] (46 sketches) and OpenSketch [Gryaditskaya et al. 2019] (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. [2020], their "vectorized data has been normalized to have uniform line width". Consequently, "as-is" their data is unrepresentative of artist sketches, since as Liu et al. [2018] 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. [2020] to match their provided raster references. For temporal information, we use the stroke order in files as the drawing order since experimentally the two typically correlate.

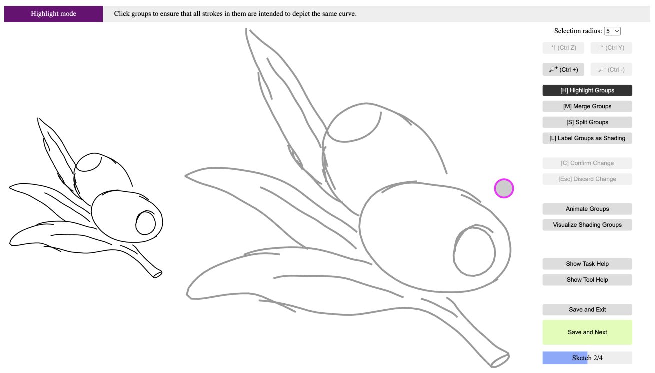

Click groups to ensure that all strokes in them are intended to depict the same curve.

Fig. 13. Our strip annotation interface. Input image © Rami Alsafadi.

C.3 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 (see our supplementary materials for the full set). The annotated sketches included one from OpenSketch [Gryaditskaya et al. 2019], 3 from Yan et al. [2020], 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.

D COMPARATIVE STUDY

D.1 Comparison and Measurement Setup

Rasterization. To compare our method to raster space approaches we rasterize our inputs using the settings recommended by [Yan et al. 2020; Yin et al. 2022], 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 [Simo-Serra et al. 2018a; Xu et al. 2019]. We use inkscape with the parameter settings of [Yan et al. 2020]. We noticed that the method of Stanko et al. [2020] 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 [Liu et al. 2018]. Liu et al. propose both methods for stroke clustering into strips and for fitting curves to these strips. Van Mossel et al. [2021] had demonstrated that their new fitting method StrokeStrip outperforms the fitting of [Liu et al. 2018]. 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.

D.2 Study Design

We conducted a comparative study to evaluate human perceptual preference between our method and representative alternatives [Liu et al. 2018; Stanko et al. 2020; Xu et al. 2019].

Each query in this study included an input drawing on top and two consolidated 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, 14. 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 [Yan et al. 2020], 22 from [Gryaditskaya et al. 2019], and the remaining 36 were inputs commissioned directly from artists.

We followed the study protocol of Liu et al. [2018]. Participants were provided a task description and shown a simple reshaping example, both taken from Liu et al. [2018]; no other explanation was provided. We use the question from Liu et al. [2018] to discard answers from participants who did not read the task description. All participants correctly answered this question. The collated questionnaires, including instructions, the filter question, and response counts per question are included in our supplementary materials.

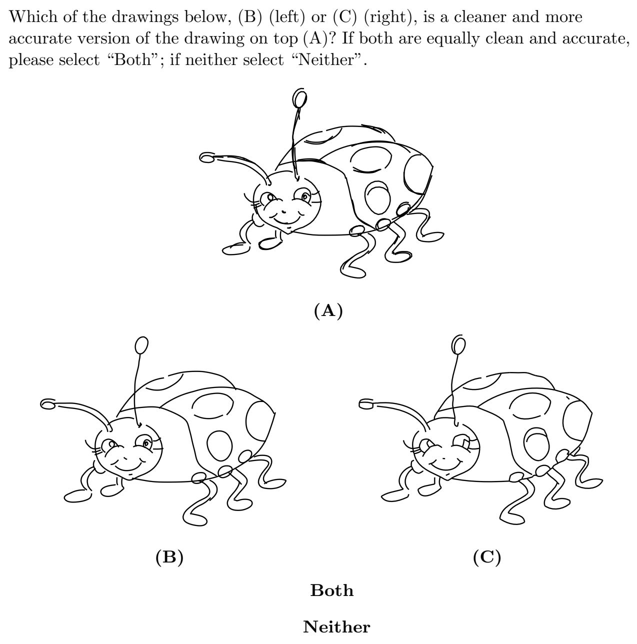

Which of the drawings below, (B) (left) or (C) (right), is a cleaner and more accurate version of the drawing on top (A)? If both are equally clean and accurate, please select "Both"; if neither select "Neither".

Fig. 14. Study question layout. Input image © Rami Alsafadi.