StrokeAggregator: Consolidating Raw Sketches into Artist-Intended Curve Drawings

CHENXI LIU, University of British Columbia
ENRIQUE ROSALES, University of British Columbia and Universidad Panamericana
ALLA SHEFFER, University of British Columbia

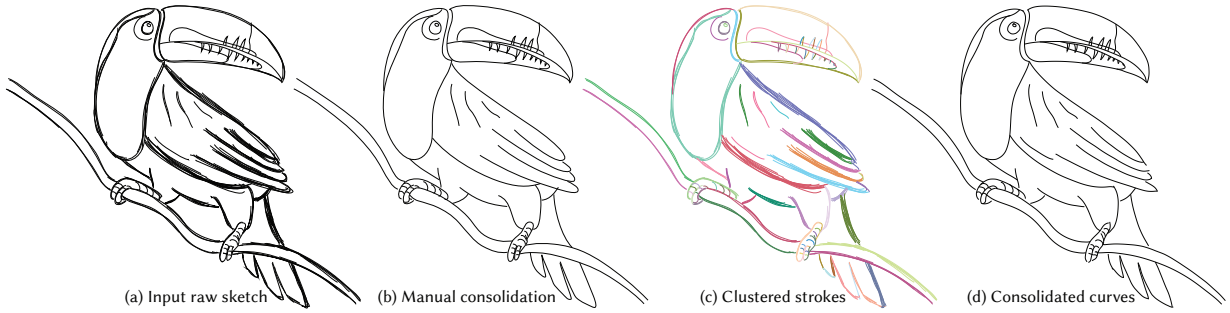


Fig. 1. Stroke consolidation: (a) a raw, vector format, sketch; (b) manually consolidated clean curve drawing; (c) algorithmically clustered strokes and (d) consolidated curves. Our output curve set (d) is of similar quality to the manually generated one (b). Please zoom in online to see image details. Raw sketch: © Enrique Rosales. Manual consolidation: © Elinor Palomares.

When creating line drawings, artists frequently depict intended curves using multiple, tightly clustered, or overdrawn, strokes. Given such sketches, human observers can readily envision these intended, *aggregate*, curves, and mentally assemble the artist's envisioned 2D imagery. Algorithmic stroke consolidation—replacement of overdrawn stroke clusters by corresponding aggregate curves—can benefit a range of sketch processing and sketch-based modeling applications which are designed to operate on consolidated, intended curves. We propose *StrokeAggregator*, a novel stroke consolidation method that significantly improves on the state of the art, and produces aggregate curve drawings validated to be consistent with viewer expectations. Our framework clusters strokes into groups that jointly define intended aggregate curves by leveraging principles derived from human perception research and observation of artistic practices. We employ these principles within a coarse-to-fine clustering method that starts with an initial clustering based on pairwise stroke compatibility analysis, and then refines it by analyzing interactions both within and in-between clusters of strokes. We facilitate this analysis by computing a common 1D parameterization for groups of strokes via common aggregate curve fitting. We demonstrate our method on a large range of line drawings, and validate its ability to generate consolidated drawings that are consistent with viewer perception via qualitative user evaluation, and comparisons to manually consolidated drawings and algorithmic alternatives.

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

Additional Key Words and Phrases: Sketch consolidation, Gestalt perception, stroke clustering, curve fitting

Authors' addresses: Chenxi Liu, University of British Columbia, chenxil@cs.ubc.ca; Enrique Rosales, University of British Columbia, Universidad Panamericana, albertr@cs.ubc.ca; Alla Sheffer, University of British Columbia, sheffa@cs.ubc.ca.

ACM Reference Format:
Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. StrokeAggregator: Consolidating Raw Sketches into Artist-Intended Curve Drawings . *ACM Trans. Graph.* 37, 4, Article 97 (August 2018), 15 pages. https://doi.org/10.1145/3197517.3201314

1 INTRODUCTION

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 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 1b) [Arora et al. 2017; Eissen and Steur 2008]. We present *StrokeAggregator*, a new algorithm that generates consolidated drawings of comparable quality to those generated by artists (Figure 1d).

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 (Section 2). Manually generating consolidated drawings from either raster or vector sketches requires expertise and time. An artist required nearly

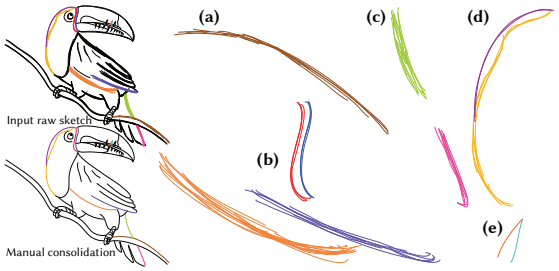


Fig. 2. Manual consolidation examples (color shows stroke grouping): (a) a typical cluster consists of strokes which are angle compatible, or roughly parallel along their side-by-side portions; (b) within each cluster, strokes are roughly evenly spaced and the internal distance is much smaller than the 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.

thirty minutes to create the consolidated drawing in Figure 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 3.1. Intuitively, we expect aggregate curves to correspond to distinct, narrow clusters of roughly evenly spaced strokes (Figure 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 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 2b). Perception literature refers to this spacing-based property as *relative proximity* or relative distance [Wagemans et al. 2012]. 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 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 above. We use these observations as the basis for a coarse-to-fine gradual clustering framework (Section 4). 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 4.1). We then perform local analysis of intra-cluster stroke spacing to detect and separate stroke branches (Section 4.2). 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 4.3) 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 5). 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).

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 3, 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 7 and supplementary material). 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.

2 RELATED WORK

Our work builds upon existing artistic practices and research on processing of line drawings, extending a line of work that employs Gestalt perceptual theory for sketch analysis [Bessmeltsev et al. 2015; Shao et al. 2012; Xu et al. 2014] and visual grouping [Jayaraman et al. 2017; Nan et al. 2011a; Xu et al. 2012].

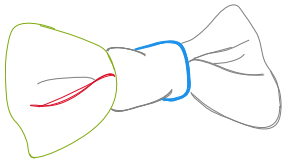


Fig. 3. Overdrawing strategies. © Enrique Rosales.

Artist Sketching Practices. Artists employ overdrawing strategies to achieve a range of effects [Arora et al. 2017; Eissen and Steur 2008]: they use earlier strokes as a visual scaffold that helps them mentally refine their intended imagery, and arrive at the desired aggregate curve form by repeatedly overdrawing existing strokes (inset, red); they leverage overdrawing to emphasize curves (inset, blue); and finally, since it is technically challenging to accurately draw long, complex curves in one continuous motion, artists often depict such curves by using collections of partially overlapping simpler strokes (inset, green). While raw line drawings effectively communicate shape, artists desire a clean, less sketchy

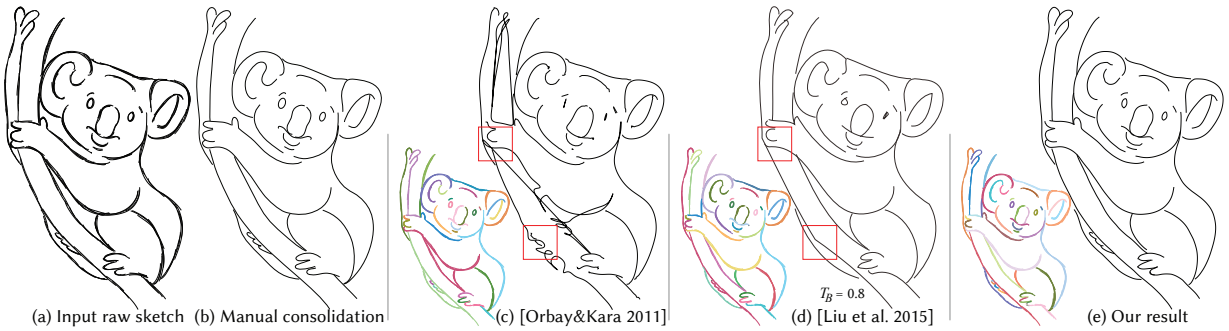


Fig. 4. Consolidation comparison (clusters and fitted curves): (a) input; (b) manually consolidated drawing; (c) [Orbay and Kara 2011]; (d) [Liu et al. 2015]; (e) our result. While the results of prior methods exhibit a range of artifacts, our result (e) is consistent with the manual consolidation output (b). Raw sketch: © Enrique Rosales. Manual consolidation: © Elinor Palomares.

line look when converting them into production-quality colorized or shaded drawings. They therefore manually retrace the original drawings, replacing multi-stroke clusters with carefully drawn corresponding aggregate curves [Arora et al. 2017; Eissen and Steur 2008]. A similar cleanup step is required if artists want to use their sketches as input to most existing sketch shading [Finch et al. 2011; Shao et al. 2012], sketch editing [Igarashi et al. 2005], single-sketch based modeling [Lipson and Shpitalni 1996; Xu et al. 2014], or posing [Bessmeltsev et al. 2016] algorithms. These tools are designed for processing consolidated drawings whose strokes already correspond to complete meaningful curves.

Vectorization of Raster Line Drawings. Traditional vectorization approaches [Bao and Fu 2012; Bo et al. 2016; Hilaire and Tombre 2006; Noris et al. 2013] extract a pixel-wide stroke skeleton from input drawings, and then fit vector curves to skeleton branches. They make only limited attempts to eliminate redundant branches formed due to overdrawing, and are best suited for vectorizing "clean", or previously consolidated, drawings where strokes depicting the same intended curve almost or fully overlap [Noris et al. 2013]. Bartollo et al. [2007] pre-filter raster line drawings by painting over small inter-stroke gaps, facilitating cleaner vectorization. Favreau et al. [2016] initialize the stroke skeleton using a set of interior region boundaries and use morphological thinning to obtain the skeleton branches for dangling strokes. They vectorize and simplify the extracted skeleton by merging skeleton branches at valence two vertices and by collapsing short branches via merging when these operations improve an overall metric that balances simplicity against input fidelity. The optimization does not support more complex operations such as merging of parallel branches, resulting in visible artifacts on the inputs we tested (Figure 14).

Progressive Overdrawing. Some drawing systems update stroke shape in real time when artists add new strokes [Bae et al. 2008; Baudel 1994; Grimm and Joshi 2012]. They rely on absolute distance and drawing order to determine which existing stroke each new stroke is designed to refine. In our experience, order is not a reliable criterion for stroke correlation, as artists often do multiple refinement passes, returning to the same intended curve after editing other portions of the drawing. Similarly, absolute distance is an ambiguous cue for determining correspondence: when an artist draws a similar stroke next to an existing one, they may aim to overdraw it, or they may be delineating a new narrow feature (such as the koala's chin or feet in Figure 4).

Line Drawing Abstraction and Simplification. Drawing abstraction methods [Grabli et al. 2004; Nan et al. 2011b] reduce visual clutter in detailed drawings by removing fine or redundant content. In contrast, our framework seeks to preserve the drawing content while identifying and consolidating groups of raw strokes that jointly represent the same intended curve. Methods for rendering 3D models using contours and other significant surface curves reduce visual clutter and produce cleaner drawings by leveraging 3D information [Kalnins et al. 2003; Wilson and Ma 2004]. Operating on inexact free-hand artist drawings, without any knowledge of their content, requires a very different methodology.

Vector Line Drawing Consolidation. Early research on sketch analysis [Rosin 1994] proposed three perceptual criteria for stroke grouping: proximity (measured with respect to stroke length), parallelism, and good continuation (measured in terms of tangent similarity at stroke end points), but does not provide constructive methods for jointly employing these criteria for stroke consolidation. Barla et al. [2005] cluster strokes incrementally, replacing pairs of strokes by an aggregate stroke if that stroke is within a fixed distance from the two paired strokes and is roughly parallel to both. Shesh and Chen [2008] and Bao and Fu [2012] extend this method to animated drawings. Noris et al. [2012] uses an interactive consolidation process that incorporates user input and drawing order. Chen et al. [2013] use a gradient field based framework to simplify or consolidate raster images. All these methods use absolute distance thresholds or neighborhood size parameters set by the user. As noted by Liu et al. [2015], over-sketched stroke densities vary between artists, and even between different regions within the same drawing (Figures 1, 4), making threshold based decisions problematic.

Orbay and Kara [2011] use a neural network to determine if strokes belong in the same cluster; the features they use are closest point distances and angles, and stroke graph distances. They use a separate step to break clusters which have branch structures (Figure 6). As they acknowledge, employing networks trained on drawings created by one artist to consolidate drawings created by another artist results in non-negligible clustering errors (Figure 4). Simo-Serra et al.[2016; 2017] propose a learning based method for

cleanup or consolidation of raster sketches that can be used as a precursor for subsequent vectorization. Their framework performs poorly on the inputs we tested (see Figure 14).

Liu et al. [2015] introduce contextual angle and proximity metrics defined relative to the size of empty spaces, or regions, enclosed by the input strokes. They threshold the computed values to determine which strokes to cluster. To address region size variation, they repeatedly merge small regions with nearby large ones and repeat the stroke merging step. They manually adjust the thresholds to produce optimal results (in the comparisons shown throughout this paper, we list the author provided value for T_B; for all inputs authors set $T_D = 0.8T_B$). The method's effectiveness diminishes on drawings with different feature scales (see koala's feet and fingers in Figure 4).

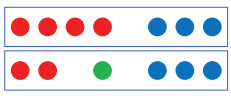
Our method also leverages contextual, or relative, proximity, but does so using region independent proximate distance evaluation. It employs proximity and angle metrics built atop existing perception research with no per-input parameter tuning. Lastly, while prior methods computed aggregate curves by minimizing the distance between these curves and point samples along the input strokes, we focus on minimizing tangent deviation between them and the input strokes. Our choice is motivated by prior research [Xu et al. 2014] and design tutorials [Eissen and Steur 2008] which indicate that the tangents of artist drawn curves are often more accurate than their positions. Consequently, our output curves better conform with viewer expectations than those generated by prior approaches (e.g. the koala's nose in Figure 4 or the eagle's beak in Figure 16). Our comparisons (Section 7) demonstrate that human observers prefer our outputs to those produced by alternative techniques 92% of the time (and judge them as on par 6%).

3 OVERVIEW

3.1 Perception of Oversketched Strokes

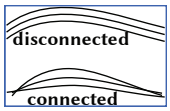
To mimic the mental process viewers apply to consolidate the drawing, we rely on the following observations about human perception of sketches derived from perception literature and sketching tutorials.

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 [Barla et al. 2005; Rosin 1994] (Figure 2a). While viewers mentally group strokes that serve as visible continuation of one another [Bessmeltsev et al. 2015] they do not hallucinate curves absent from the drawing, and thus employ separate corresponding aggregate curves for such strokes (Figure 2c).

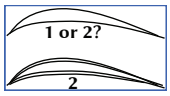


Relative proximity. Perceptual literature strongly suggests that humans group objects based on *relative proximity*, or *relative* distance: given a set of shapes, we visually group objects if the spacing between them is much smaller than the space between them and other objects (see inset, color indicates clustering) [Wagemans et al. 2012]. 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 (inset, 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 the inset 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 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).



Connectedness. The *connectedness* principle highlighted by perception research [Wagemans et al. 2012] 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 inset).



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 inset, top). 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 [Eissen and Steur 2008] 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 (inset, bottom). We refer to this principle as *strength in numbers* and use it to resolve ambiguous configurations, by using stroke number within a cluster as a factor in the final decision making (Figure 5d, Section 4.3).

3.2 Algorithm

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

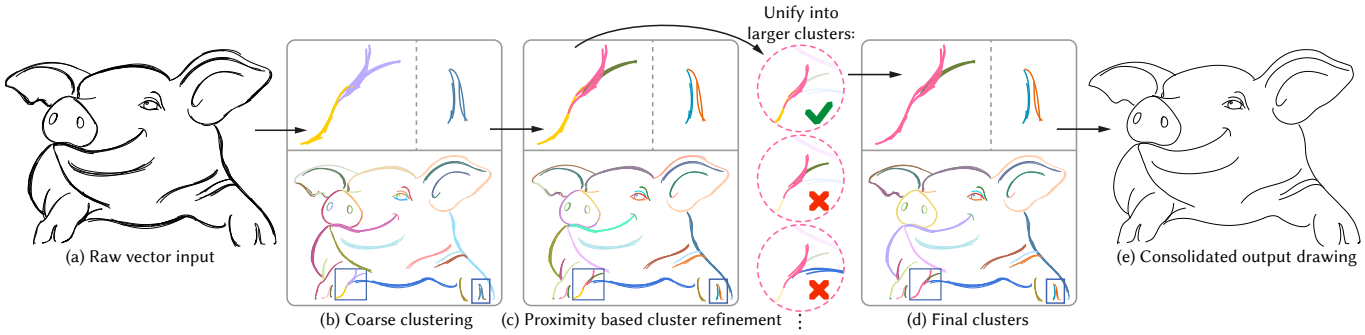


Fig. 5. Given a raw vector sketch (a), our method first clusters based on pairwise compatibilities of angle and relative proximity, resulting in clusters consisting of connected parallel strokes (b, Section 4.1). Our method then analyzes relative proximity within each cluster to separate branches (c, Section 4.2). Given these reliable clusters, our method assesses all pairs of nearby clusters and merges them following the visual grouping rules (d, Section 4.3). Finally, the clusters are consolidated into the cleaned-up sketch (e, Section 5). Raw sketch: © Enrique Rosales.

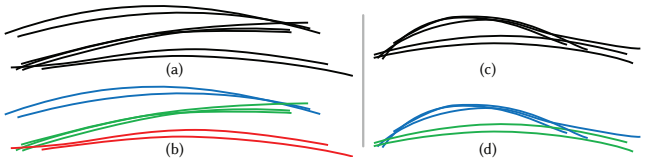


Fig. 6. Local versus global proximity: (a,c) on-average evenly spaced (and connected) strokes may depict multiple aggregate curve branches; (b,d) perceived narrow clusters.

between strokes may vary at different points along them, requiring fine-grained local analysis to separate connected stroke branches that depict different curves (Figure 6), 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 5). We first coarsely cluster strokes based on average, or global, compatibility between their strokes (Section 4.1, Figure 5b). 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 4.1.1). 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 4.1.4). 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 4.2, Figure 6). The output of this stage is a set of stroke clusters that satisfy all our perceptual criteria, and their corresponding aggregate curves (Figures 5c, 6bd).

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 4.3, Figure 5d).

To assess the different properties of the considered clusters throughout the algorithm, we compute their corresponding aggregate curves (Section 5) 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 5e).

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 [Baran et al. 2010; McCrae and Singh 2009]).

Raw strokes captured via a stylus-on-tablet interface are often noisy due to a combination of involuntary hand movement and capture software inaccuracy [Baran et al. 2010; McCrae and Singh 2009]. We pre-process the raw data as described in Appendix A. We do not use this process on previously cleaned data, such as the examples provided by [Liu et al. 2015]. 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.

4 STROKE CLUSTERING

4.1 Coarse Clustering

4.1.1 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 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 [Bansal et al. 2004]. 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. [2015], which provides an efficient approximation of the optimum.

4.1.2 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 [Barla et al. 2005; Liu et al. 2015] or by considering only angles at closest points [Orbay and Kara 2011]. 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 5), 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 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:

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

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 [Hess and Field 1999]. 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:

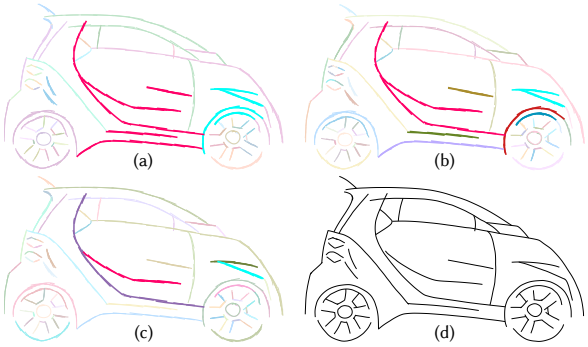


Fig. 7. Clustering stages: (a) angle based clustering output with two clusters (pink and cyan) highlighted; (b) average proximity based clustering breaks these two clusters into roughly evenly spaced distinct components; (c) local refinement separates branches producing uniformly narrow clusters; (d) consolidated output. Input sketch is from [Orbay and Kara 2011].

$\sigma_1 = 9^\circ/3.5, \sigma_2 = 7^\circ/3.5$. 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 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 8). This value is small enough to allow strokes to be grouped together if they share angle compatible intermediate strokes, but pushes them apart otherwise.

4.1.3 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.

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 $\mathcal{I}_1$ (inset, blue). We evenly sample the points $\mathbf{p}$ along $\mathcal{S}_{i,j}^A$ and define the angular distance as

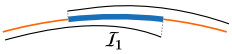


Fig. 8. Stroke pair layouts.

$$D_a(\mathcal{S}_i, \mathcal{S}_{i,j}^A) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} A_i(\mathbf{p}'), \quad (2)$$

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

4.1.4 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 7a). 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 7b). We use the computed correspondences to further refine these clusters using *local* proximity analysis (Section 4.2, Figure 7c).

To measure the proximity, or distance, between two strokes $\mathcal{S}_i$ and $\mathcal{S}_j$ we fit them using an aggregate curve $\mathcal{S}_{ij}^A$ 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 $\mathcal{S}_{ij}^A$. Note that by construction the points $\mathbf{p}', \mathbf{q}, \mathbf{p}$ are colinear and the line connecting them is orthogonal to the aggregate curve. The average distance is then defined as

$$D_{i,j}(\mathcal{I}_1) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} ||\mathbf{p} - \mathbf{q}||. \tag{3}$$

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 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 C, for each stroke, we locate its nearest neighbor based on the inter-stroke distance. We define the internal cluster proximity 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 C and 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) 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 [Campello et al. 2015].

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

$$A_{i,j}(\mathcal{I}_1) = \frac{1}{|\mathcal{I}_1|} \sum_{\mathbf{p}' \in \mathcal{I}_1} \arccos(\mathbf{t}(\mathbf{p}) \cdot \mathbf{t}(\mathbf{q})), \tag{4}$$

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 5) and compare the curve's length to width ratio against the threshold $T_n = 8.5$ established via our study (Appendix B).

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

$$W_c = \max(W_s, \underset{\mathbf{p} \in \mathcal{I}_1}{\text{median}}(||i_l(\mathbf{p}) - i_r(\mathbf{p})||)). \tag{5}$$

4.2 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 6, 7, 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 4.2.2), and cluster narrowness. We assess narrowness as described in Section 4.1.4. For any cluster that fails one of these tests, we perform the split that maximally reduces spacing unevenness (Section 4.2.1). 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 4.2.2). We generate candidate sub-clusters based on local inter-stroke spacing (Section 4.2.1) and then compare their internal spacing to the inter-cluster one to determine if they indeed need to be separated (Section 4.2.2).

4.2.1 Potential Clusters. We compute potential sub-clusters by analyzing local spacing between strokes (Figure 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 9, inset). We order the intersections from leftmost to rightmost (with left and right defined with respect

ALGORITHM 1: Recursive Branch Separation

Input: A set of strokes to separate, C.
Output: *ResultBranches*.
ResultBranches ← $\{C\}$;
PotentialSeparations ← FindPotentialSeparations(C) (Sec. 4.2.1);
$R_{max} \leftarrow 0$; $\{C_L^*, C_R^*\} \leftarrow \{C, \emptyset\}$;
for *each separation* $\{C_L, C_R\}$ *in PotentialSeparations* **do**
  $R \leftarrow$ ComputeGapRatio($\{C_L, C_R\}$) (Sec. 4.2.2);
  if $R > R_{max}$ **then**
    $R_{max} \leftarrow R$; $\{C_L^*, C_R^*\} \leftarrow \{C_L, C_R\}$;
  end
end
if $R_{max} > T_d$ *or* C *violates* Narrowness **then**
  LeftBranches ← RecursiveBranchSeparation(C_L^*);
  RightBranches ← RecursiveBranchSeparation(C_L^*);
  ResultBranches ← *LeftBranches* ∪ *RightBranches*;
end

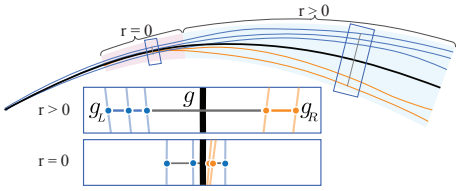


Fig. 9. Local cluster refinement: Pointwise stroke correspondences are defined using intersections between strokes and orthogonal rays emanating from the cluster's aggregate stroke (black). The spacing between lowest top (blue) and highest bottom (orange) intersection points is significantly larger than 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).

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 the right and the strokes they lie on are locally visibly separate (Figure 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 4.1.4.

Given a candidate gap, we assign the strokes to the left and right of it into separate left and right clusters, C_L and 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

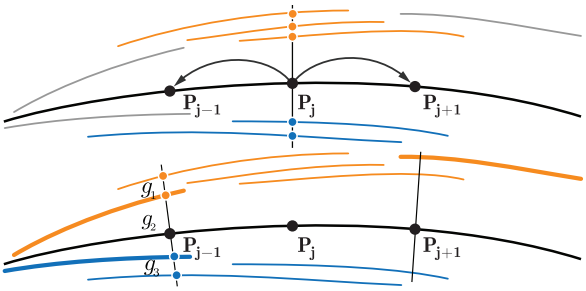


Fig. 10. The growing step for potential separation generation. At position $\mathbf{p_j}$, given the gap across the aggregate curve, the intersection points are labeled into blue and orange, and the strokes are labeled correspondingly. The assignment at $\mathbf{p_j}$ is propagated into $\mathbf{p_{j-1}}$ 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 $\mathbf{p_{j+1}}$.

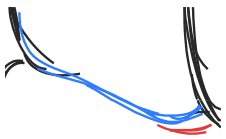
as all currently marked strokes remain on correct sides. At each encountered aggregate curve point, we split the unmarked strokes locally based on the largest gap between the previously marked strokes. Intuitively, the optimal assignment of the remaining strokes is one that maximizes the average gap 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 C_L, or assigning all remaining strokes to C_R.

4.2.2 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 4.1.4, 4.2.1), we use $2W_s$ as the lower bound.



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 inset), we follow the strength in numbers

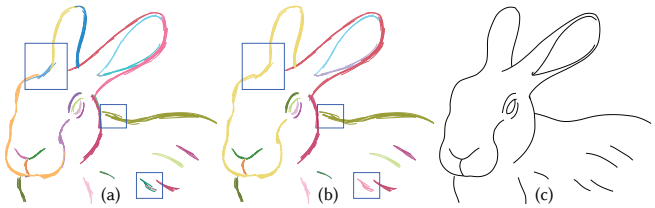
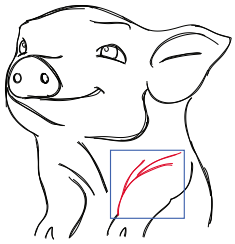


Fig. 11. Final unification: (a) before; (b) after, (c) consolidated result. Input sketch: © Enrique Rosales.

principle and use the smaller bound $\max(d_l, d_r, W_s)$. This choice facilitates 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 6ab). Thus, either of the left or right clusters may consist of more than one branch (see inset, © Enrique Rosales) 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.

4.3 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 4.2) 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.

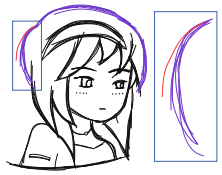
Pairwise Assessment. We determine if a pair of clusters C_l and C_r should be merged based on narrowness, local angular compatibility, and relative proximity. We assess narrowness as before: we compute a common aggregate curve $\mathcal{S}^A_{lr}$ that corresponds to the union of the two clusters. If 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 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.

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 inset) 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 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 cluster'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 (inset, green) as an envelope vertex, otherwise, we use a point along the ray at a half width distance as a vertex (inset, 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 $\mathcal{S}^A_{lr}$.



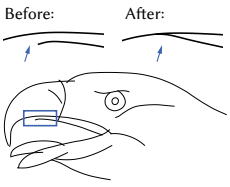
Single-stroke clusters. As noted earlier, relative proximity assessment requires at least three strokes to be meaningful, making assessment of proximity for single-stroke clusters problematic. Moreover, when drawing free-hand, artists do occasionally draw outlier strokes—ones that are intended to depict a target aggregate curve but are sufficiently inaccurate to be visually separate from the other strokes in their intended cluster (see inset, sourced from [Liu et al. 2015]).

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 end-points of aggregate curves $\mathcal{S}_i$ and $\mathcal{S}_j$ with width W_i and W_j respectively as coincident, 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 [Sorkine et al. 2004] ; 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 the inset and comparisons of all data in our supplementary materials. All final results shown in this paper have this optional step turned on.

5 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 12). 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 [Lee 2000; Levin 2004]. The standard MLS formulation does not support tangent optimization, since tangent processing requires point order information which is 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 [Chen et al. 2013]; however, this still requires consistently orienting the resulting tangents.

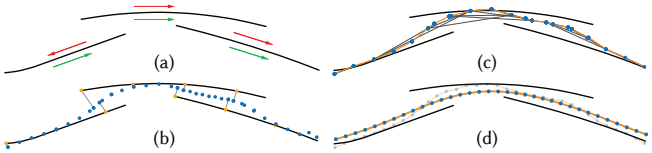


Fig. 12. Aggregate curve fitting: (a) input stroke with original (red) and consistent (green) orientations; (b) MLS fitting output; (c) proximity graph and extracted polyline (e) resampled (thin) and final (thick) optimized polyline curve.

Stroke Orientation. To perform any meaningful operations on point tangents, we require their orientations to be consistent (Figure 12a). 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 [Orbay and Kara 2011].

MLS fitting. Our initial fitting step uses Moving-Least-Squares (MLS) with adaptive neighborhood size [Lee 2000; Levin 2004]. 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) [Levin 2004]. To conduct the MLS projection step, each point needs to be associated with a local neighborhood. Following the method in Lee [2000], 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 12b).

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

$$K(\mathbf{p}^0, T) = \frac{\sum_{p \in N(\mathbf{p}^0)} \mathbf{t} \cdot \theta(||\mathbf{p} - \mathbf{p}^0||)}{\| \sum_{p \in N(\mathbf{p}^0)} \mathbf{t} \cdot \theta(||\mathbf{p} - \mathbf{p}^0||) \|}. \quad (6)$$

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 12c). 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 [Chu 1965; Edmonds 1967] 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

$$d(\mathcal{S}, \mathcal{S}^A) = \sum_{i=1}^{n} \| \frac{\mathbf{p}_{i+1} - \mathbf{p}_i}{||\mathbf{p}_{i+1} - \mathbf{p}_i||} - K(\mathbf{p}_i, T)||^2 + \lambda ||\mathbf{p}_i - \mathbf{p}_i^0||^2, \quad (7)$$

$\mathbf{p}_i^0$ 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 7 using iterated least squares. We define the kth round objective as

$$d(\mathcal{S}^k, \mathcal{S}^{k-1}) \approx \sum_{i=1}^{n} || \frac{\mathbf{p}_{i+1}^k - \mathbf{p}_i^k}{||\mathbf{p}_{i+1}^{k-1} - \mathbf{p}_i^{k-1}||} - K(\mathbf{p}_i^{k-1}, T)||^2 + \lambda ||\mathbf{p}_i^k - \mathbf{p}_i^0||^2 \quad (8)$$

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.

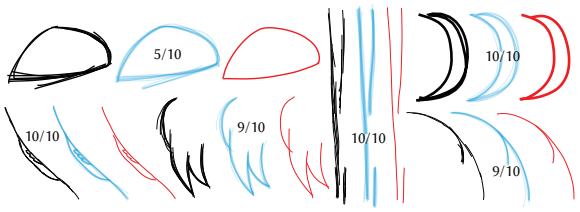


Fig. 13. Examples of manually (blue) and algorithmically (red) traced aggregate curves of different stroke configurations (black). Manual results form multiple participants are overlaid over one another. The ratio shows the number of participants whose results agreed with the plurality consolidated result in terms of output curve number and approximate location. In all cases our result aligns with the plurality response.

6 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. The exact questionnaires used in the evaluations are included in our supplementary material.

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 13); the rest are included in the supplementary. 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 13) 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 1 and 4 show two such artist results side-by-side with our outputs; the rest are included in the supplementary. As these comparisons show, our results are well-aligned with the artist outputs. It took the artist 10 to 30 minutes

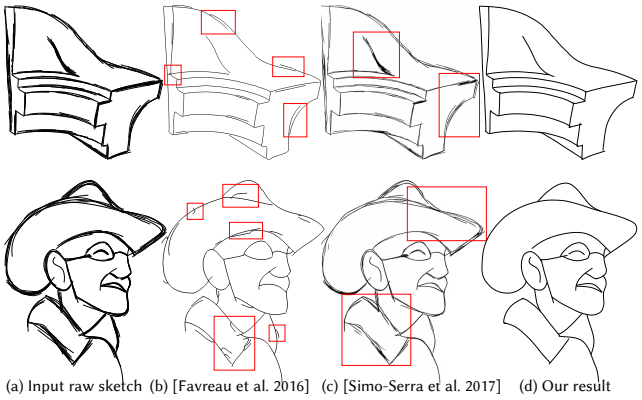


Fig. 14. Comparison with raster cleanup and vectorization methods. The top input is from [Orbay and Kara 2011]; the bottom input is from [Liu et al. 2015].

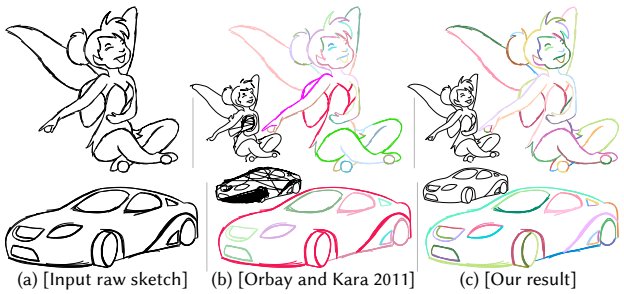


Fig. 15. Comparison (clusters and fitting) with [Orbay and Kara 2011]. Inputs sourced from [Liu et al. 2015]. In column two the examples of wrongly clustered strokes are highlighted.

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 14 compares our output against two raster-space methods for vectorization [Favreau et al. 2016] and cleanup [Simo-Serra et al. 2017]. 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 4, 15, and 16 show comparisons to vector consolidation methods [Liu et al. 2015; Orbay and Kara 2011]. The results in these figures were provided by the authors. As shown in Figures 4 and 15, 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 4 and 16 compare our results with those of Liu et al. Our method achieves better fine input feature preservation, while still correctly

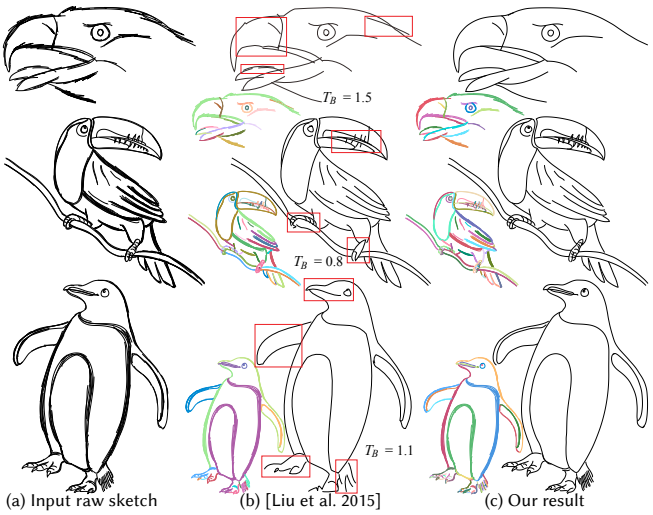


Fig. 16. Comparison (clusters and fitting) with [Liu et al. 2015]. Note the differences in the consolidation of feet and other fine features. The eagle input is sourced from [Orbay and Kara 2011]. Toucan, penguin: © Enrique Rosales.

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 [Favreau et al. 2016; Liu et al. 2015; Orbay and Kara 2011; Simo-Serra et al. 2017] and artists (5 each of [Favreau et al. 2016; Orbay and Kara 2011; Simo-Serra et al. 2017], 6 [Liu et al. 2015], 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 generated 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". (see supplementary material for exact questionnaire). The full statistics and questionnaires are provided in the supplementary material. 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 [2015], 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 with manual consolidation and is far superior to prior art.

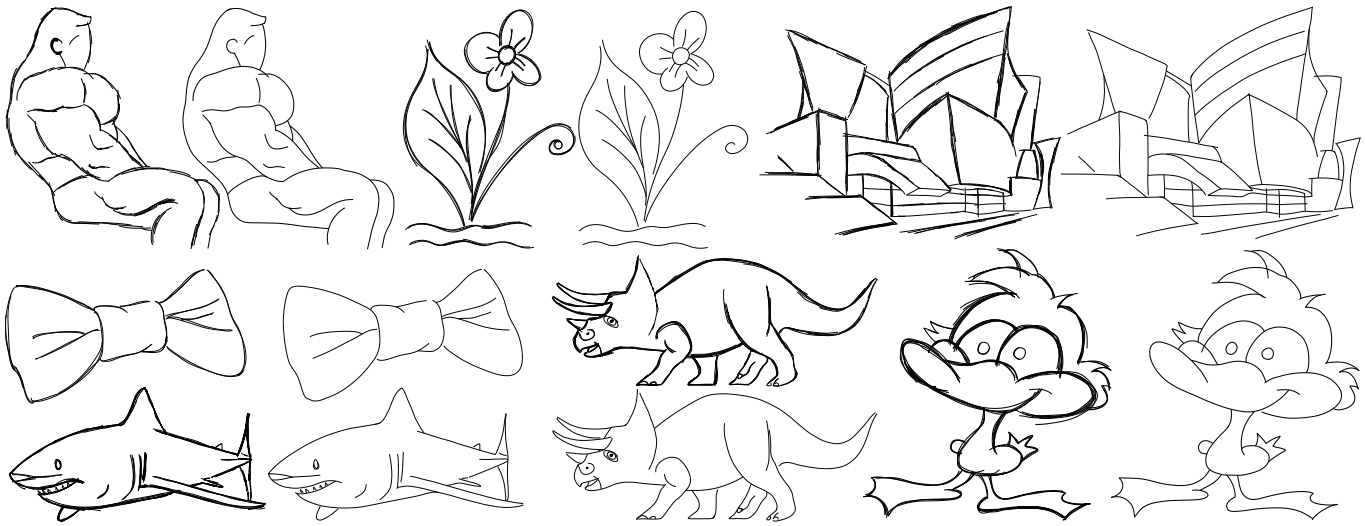


Fig. 17. Additional diversely sourced results. The duck input is sourced from [Liu et al. 2015], the architectural model and man are sourced from [Orbay and Kara 2011], shark and triceratops: © Cristina Arciniega, flower and bow-tie: © Enrique Rosales. Please zoom in online to see image details.

7 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 the paper and others are included in the supplementary material. Our inputs include examples sourced from prior work, e.g. fandisk (Figure 14), eagle (Figure 16), man and opera (Figure 17) are from [Orbay and Kara 2011], and grandpa (Figure 14), duck (Figure 17), and fairy and car (Figure 15) are from [Liu et al. 2015]. They also include new inputs created by two different artists (e.g. Figures 1, 4, shark, bowtie and triceratops in Figures 17). 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 5, 7, 11 show the stages of our progressive cluster refinement process, highlighting the contribution of each stage. The local analysis stage (Section 4.2) is critical for processing clusters with branching structures (Figures 5b, 7b). The narrowness cue is critical in processing features such as the tail of the penguin (Figure 16), the moon shaped windows on the building (Figure 18), or the stripes on the side of the shark (Figure 17).

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 6), 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 Section 2, and is not directly applicable to stylized line drawings (Figure 18ab). 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 18c 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.

8 CONCLUSIONS

We presented a new, robust line drawing consolidation framework that does not require per-input parameter tuning and produces results validated to be consistent with viewer perception. Our method leverages a set of observations about perceptual cues and artist techniques that help viewers parse rough line-drawings and employs those in a coarse-to-fine clustering framework.

The values for the proximity and narrowness thresholds we gleaned from domain specific small-scale human studies were sufficient for our algorithmic needs. The most interesting avenue for future research is an in-depth perceptual study of the interaction of the different perceptual cues humans employ for mental sketch consolidation.

ACKNOWLEDGMENTS

We would like to thank Gunay Orbay, Levent Burak Kara, Xueting Liu and Tien-Tsin Wong for providing comparison data, Cristina

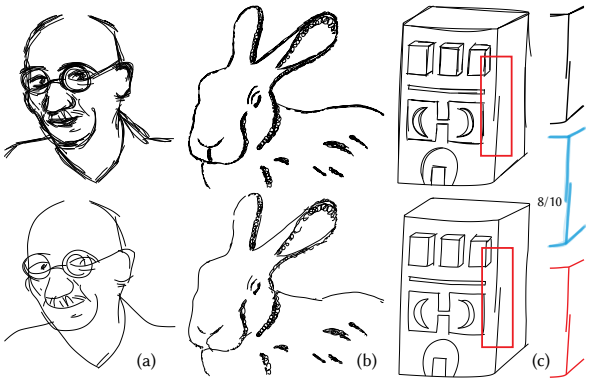


Fig. 18. Our framework relies on local context rather than recognition. Thus its ability to process intentionally sketchy (a) or stylized inputs with unreliable stroke tangents (b) is limited. (c) Our clustering choices on this input are consistent with local human ones (8 out of 10 viewers keep the 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.

Arciniega, Elinor Palomares and Hugo Vargas for their artistic inputs, Nicholas Vining for paper editing, and the reviewers for their insightful suggestions. The authors were supported by NSERC.

REFERENCES

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

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.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation clustering. *Machine Learning* 56, 1-3 (2004), 89–113.

Bin Bao and Hongbo Fu. 2012. Vectorizing line drawings with near-constant line width. In *Proc. International Conference on Image Processing*. 805–808.

Ilya Baran, Jaakko Lehtinen, and Jovan Popović. 2010. Sketching clothoid splines using shortest paths. In *Computer Graphics Forum*, Vol. 29. 655–664.

Pascal Barla, Joëlle Thollot, and François Sillion. 2005. Geometric Clustering for Line Drawing Simplification. In *Proc. EGSR*.

A. Bartolo, K. P. Camilleri, S. G. Fabri, J. C. Borg, and P. J. Farrugia. 2007. Scribbles to Vectors: Preparation of Scribble Drawings for CAD Interpretation. In *Sketch-Based Interfaces and Modeling*.

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

Mikhail Bessmeltsev, Will Chang, Nicholas Vining, Alla Sheffer, and Karan Singh. 2015. Modeling character canvases from cartoon drawings. *ACM Trans. Graph.* 34, 5 (2015), 162.

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

Pengbo Bo, Gongning Luo, and Kuanquan Wang. 2016. A graph-based method for fitting planar B-spline curves with intersections. *Journal of Computational Design and Engineering* 3, 1 (2016), 14–23.

Ricardo J. G. B. Campello, Davoud Moulavi, Arthur Zimek, and Jörg Sander. 2015. Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection. *ACM Trans. Knowl. Discov. Data* 10, 1 (2015), 5:1–5:51.

Jiazhou Chen, Gael Guennebaud, Pascal Barla, and Xavier Granier. 2013. Non-Oriented MLS Gradient Fields. In *Computer Graphics Forum*, Vol. 32. 98–109.

Yoeng-Jin Chu. 1965. On the shortest arborescence of a directed graph. *Science Sinica* 14 (1965), 1396–1400.

Jack Edmonds. 1967. Optimum branchings. *J. Res. Nat. Bur. Standards* 71B, 4 (1967), 233–240.

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 (2016), 120.

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.

Stéphane Grabli, Frédo Durand, and François Sillion. 2004. Density Measure for Line-Drawing Simplification. In *Proc. Pacific Graphics*.

Cindy Grimm and Pushkar Joshi. 2012. Just DrawIt: A 3D Sketching System. In *Proc. SBIM*. 121–130.

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

Xavier Hilaire and Karl Tombre. 2006. Robust and accurate vectorization of line drawings. *IEEE Trans. Pattern Anal. Mach. Intell* 28, 6 (2006), 890–904.

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

Pradeep Kumar Jayaraman, Chi-Wing Fu, Jianmin Zheng, Xueting Liu, and Tien-Tsin Wong. 2017. Globally Consistent Wrinkle-Aware Shading of Line Drawings. *IEEE Trans. Vis. Comput. Graph* (2017).

Robert D. Kalnins, Philip L. Davidson, Lee Markosian, and Adam Finkelstein. 2003. Coherent Stylized Silhouettes. *ACM Trans. Graph.* 22, 3 (2003), 856–861.

Margret Keuper, Evgeny Levinkov, Nicolas Bonneel, Guillaume Lavoué, Thomas Brox, and Bjorn Andres. 2015. Efficient decomposition of image and mesh graphs by lifted multicuts. In *Proc. ICCV*. 1751–1759.

In-Kwon Lee. 2000. Curve reconstruction from unorganized points. *Computer aided geometric design* 17, 2 (2000), 161–177.

David Levin. 2004. Mesh-independent surface interpolation. In *Geometric modeling for scientific visualization*. 37–49.

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.

Xueting Liu, Tien-Tsin Wong, and Pheng-Ann Heng. 2015. Closure-aware sketch simplification. *ACM Trans. Graph.* 34, 6 (2015), 168.

James McCrae and Karan Singh. 2009. Sketching piecewise clothoid curves. *Computers & Graphics* 33, 4 (2009), 452–461.

Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. 2011a. Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM Trans. Graph.* 30, 6 (2011).

Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. 2011b. Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM Trans. Graph.* 30, 6 (2011). 185:1-185:10.

Gioacchino Noris, Alexander Hornung, Robert W Sumner, Maryann Simmons, and Markus Gross. 2013. Topology-driven vectorization of clean line drawings. *ACM Trans. Graph.* 32, 1 (2013), 4.

Gioacchino Noris, Daniel Sỳkora, A Shamir, Stelian Coros, Brian Whited, Maryann Simmons, Alexander Hornung, M Gross, and R Sumner. 2012. Smart scribbles for sketch segmentation. In *Computer Graphics Forum*, Vol. 31. 2516–2527.

Gunay Orbay and Levent Burak Kara. 2011. Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Trans. Vis. Comput. Graph* 17, 5 (2011), 694–708.

Paul L. Rosin. 1994. Grouping Curved Lines. In *Machine Graphics and Vision 7*. 625–644.

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. In *Computer Graphics Forum*, Vol. 27. 537–545.

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.

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2017. Mastering Sketching: Adversarial Augmentation for Structured Prediction. *ACM Trans. Graph.* (2017).

O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. 2004. Laplacian Surface Editing. In *Proc. Symposium on Geometry Processing*. 175–184.

Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger 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.

Brett Wilson and Kwan-Liu Ma. 2004. Rendering complexity in computer-generated pen-and-ink illustrations. In *Proc. NPAR*. 129–137.

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 (2014), 131:1–131:13.

Pengfei Xu, Hongbo Fu, Oscar Kin-Chung Au, and Chiew-Lan Tai. 2012. Lazy selection: a scribble-based tool for smart shape elements selection. *ACM Trans. Graph.* 31, 6 (2012), 142.

APPENDIX A: STROKE PRE-PROCESSING

Raw strokes captured via a stylus-on-tablet interface are often noisy due to both involuntary hand movement and capture software inaccuracy [Baran et al. 2010; McCrae and Singh 2009]. In particular,

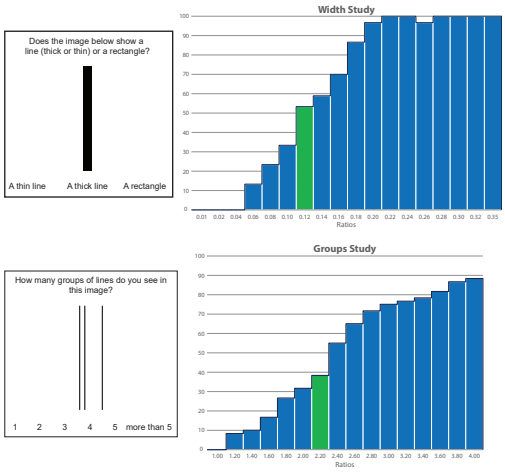


Fig. 19. (top) Narrowness threshold question examples and answer distribution;(bottom) proximity question examples and answer distributions.

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 [2015] 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. We smooth and densely resample the strokes using the Cornucopia algorithm [Baran et al. 2010]. 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).

APPENDIX B: 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 [Hess and Field 1999], 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 19,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 19, 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 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 19,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 19,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 full statistics and questionnaires for the two studies are provided as supplementary material. 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 6).