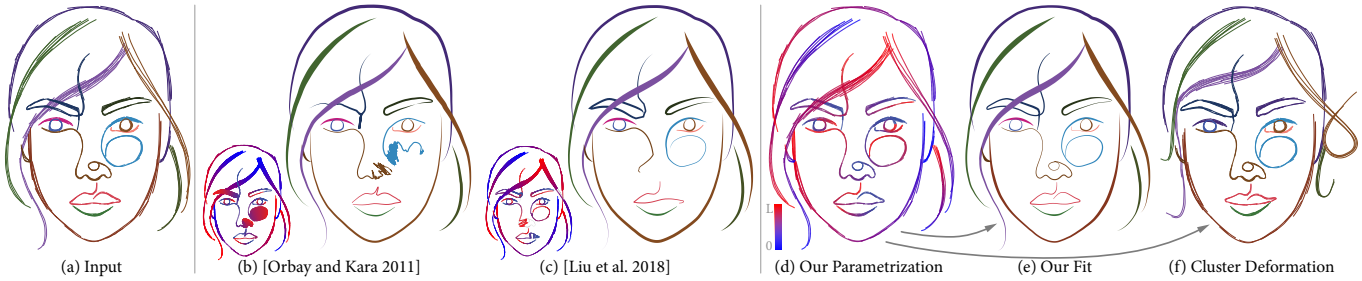
StrokeStrip: Joint Parameterization and Fitting of Stroke Clusters

DAVE PAGUREK VAN MOSSEL, University of British Columbia, Canada
CHENXI LIU, University of British Columbia, Canada
NICHOLAS VINING, University of British Columbia, Canada and NVIDIA, Canada
MIKHAIL BESSMELTSEV, Université de Montréal, Canada
ALLA SHEFFER, University of British Columbia, Canada



(a) Input (b) [Orbay and Kara 2011] (c) [Liu et al. 2018] (d) Our Parametrization (e) Our Fit (f) Cluster Deformation

Fig. 1. Free-hand sketches use clusters of strokes (a, each cluster colorized in different color) to depict intended aggregate curves. Existing methods that aim to fit aggregate curves to vector stroke clusters (b,c) frequently produce outputs inconsistent with artist intent and viewer expectations (e.g. nose, mouth, left eyebrow (b,c); insets show the point orderings they generate and use to compute fits, visualized using a red to blue color scheme). StrokeStrip jointly arc length parameterizes each cluster (d, red to blue color scheme), facilitating aggregate curve fitting consistent with viewer expectations (e) and intuitive cluster level editing operations (f; we deform the hair strands and the lips). Please zoom in to see image details throughout the paper. Drawing ©Enrique Rosales and Sari Pagurek van Mossel.

When creating freeform drawings, artists routinely employ clusters of overdrawn strokes to convey intended, aggregate curves. The ability to algorithmically fit these intended curves to their corresponding clusters is central to many applications that use artist drawings as inputs. However, while human observers effortlessly envision the intended curves given stroke clusters as input, existing fitting algorithms lack robustness and frequently fail when presented with input stroke clusters with non-trivial geometry or topology. We present *StrokeStrip*, a new and robust method for fitting intended curves to vector-format stroke clusters. Our method generates fitting outputs consistent with viewer expectations across a vast range of input stroke cluster configurations. We observe that viewers perceive stroke clusters as continuous, varying-width *strips* whose paths are described by the intended curves. An arc length parameterization of these strips defines a natural mapping from a strip to its path. We recast the curve fitting problem as one of parameterizing the cluster strokes using a *joint* 1D parameterization that is the restriction of the natural arc length parameterization of this strip to the strokes in the cluster. We simultaneously compute the joint cluster parameterization and implicitly reconstruct the *a priori* unknown strip geometry by solving a variational problem using a discrete-continuous optimization framework. We use this parameterization to compute parametric aggregate curves whose shape reflects the geometric properties of the cluster strokes at the corresponding isovalues. We demonstrate StrokeStrip outputs to be significantly better aligned with observer preferences compared to those of prior art; in a perceptual study, viewers preferred our fitting outputs by a factor of 12:1 compared to alternatives. We further validate our algorithmic choices via a range of ablation studies; extend our framework to raster data; and illustrate applications that benefit from the parameterizations produced.

CCS Concepts: • **Computing methodologies** → **Shape modeling**.

ACM Reference Format:
Dave Pagurek van Mossel, Chenxi Liu, Nicholas Vining, Mikhail Bessmeltsev, and Alla Sheffer. 2021. StrokeStrip: Joint Parameterization and Fitting of Stroke Clusters. *ACM Trans. Graph.* 40, 4, Article 1 (August 2021), 18 pages. https://doi.org/10.1145/3450626.3459777

1 INTRODUCTION

When creating freehand sketches, artists ubiquitously employ clusters of nearby, roughly parallel strokes to communicate individual intended, or *aggregate*, curves (Fig. 1a). They form such clusters by repeatedly overdrawing strokes to refine the shape of the curves they aim to convey, to emphasize or thicken these curves, or to add interest and texture to the drawn content [Arora et al. 2017; Eissen and Steur 2008]. When presented with overdrawn stroke clusters, human observers effortlessly imagine the artist's intended aggregate curves [Liu et al. 2018; Yan et al. 2020]. To facilitate downstream digital processing, artists increasingly use drawing tools that store the created raw sketches in vector form (e.g. [Blender 2021]). Most sketch processing applications target clean, overdrawing-free, vector drawings where each stroke is assumed to depict an intended

Authors' addresses: Dave Pagurek van Mossel, University of British Columbia, Canada, dpagurek@cs.ubc.ca; Chenxi Liu, University of British Columbia, Canada, chenxil@cs.ubc.ca; Nicholas Vining, University of British Columbia, Canada, NVIDIA, Canada, nvining@cs.ubc.ca; Mikhail Bessmeltsev, Université de Montréal, Canada, bmpix@iro.umontreal.ca; Alla Sheffer, University of British Columbia, Canada, sheffa@cs.ubc.ca.

0730-0301/2021/8-ART1 $15.00
https://doi.org/10.1145/3450626.3459777

meaningful curve (Fig. 1e, Sec. 2). To process raw freehand vector sketches, these applications rely on sketch consolidation methods which cluster strokes that jointly depict individual intended curves, and then fit aggregate curves to each cluster [Liu et al. 2018, 2015; Orbay and Kara 2011]. Previous fitting methods successfully process clusters that depict relatively simple, low-curvature curves, but often fail to produce results consistent with human expectations on more complex inputs (Sec. 2, Fig. 1bc). We introduce *StrokeStrip*, a new method for fitting intended curves to vector stroke clusters capable of robustly generating aggregate curves consistent with human expectations on arbitrarily complex inputs (Fig. 1e).

We achieve this goal by leveraging insights about human perception of stroke clusters (Fig. 2). We observe that viewers see stroke clusters as narrow, varying-width aggregate strokes, or *strips*, whose paths define the intended aggregate curves (Fig. 2b). Human observers mentally delineate the outlines of the strips depicted by each cluster (Appendix A), distinguishing between points on the strokes that are next to one another *within the strip*, or WTS adjacent, and those adjacent only in Euclidean space (Fig. 3). Manually traced aggregate curve data [Liu et al. 2018; Yan et al. 2020] indicates that viewers expect aggregate curves to follow the direction of the raw strokes, and to smoothly average the geometric properties of strokes that viewers perceive as being WTS adjacent. The main algorithmic challenge in computing such curves is therefore to identify the sets of stroke points whose shape properties jointly define the geometry of each point along the fitted aggregate curve. This problem can naturally be cast as a computation of a 1D cluster parameterization whose isolines define these sets of points (Fig. 2cd). Specifically, the parameterization we look for is a restriction of the natural arc length parameterization of the perceived strip (Fig. 2c) to the strokes in the cluster (Fig. 2d). Given this *joint* newstroke parameterization we can directly compute a parametric fitted curve whose shape at each parameter value u is reflective of the average shape (positions, tangents, and curvatures) of the cluster strokes at u (Fig. 2e, Sec. 6).

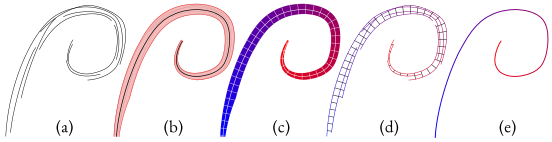


Fig. 2. Viewers perceive stroke clusters (a) as depictions of aggregate, varying width *strips*, (b) whose paths (b, black) depict the intended aggregate curves and follow the average direction of the drawn strokes. The natural arclength parameterization of each strip (c, blue to red) induces a joint parameterization of both the cluster's strokes (d) and the aggregate curve (e). The isolines of the parameterizations (c, d) are orthogonal to this curve.

To produce fitted curves aligned with human perception, the joint cluster parameterization must satisfy several requirements that follow from the properties of the strip parameterizations we seek (Sec. 3.1). To accurately capture the geometry of the cluster strokes, we aim to compute arc length parameterized aggregate curves; we thus aim to produce *arc length preserving* strip and cluster parameterizations. The parameterizations must be continuous, and therefore isovalues at WTS adjacent stroke points should be similar. Aggregate curve tangents, and consequently the gradients of the parameterizations that define them, should be aligned with the stroke tangents; in other words, parameterization isolines need

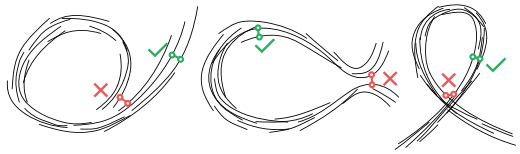


Fig. 3. Viewers distinguish between pairs of stroke points they perceive as adjacent within the aggregate strip, i.e. WTS adjacent (green), and those they see as far apart within it (red), despite similar pairwise 2D distances.

to be approximately orthogonal to the strokes. The isolines should cross the clusters forming *cross-sections* that extend from one side of the strip to the other (Fig. 2d). Finally, since the arc length parameterization of a strip along the gradient direction is strictly monotonic, the restriction to the strokes must be *strictly monotonic* as well.

The core challenge in computing a parameterization that satisfies these requirements is that we do not *a priori* know the strip geometry; in other words, we do not know which points on different strokes that are adjacent in Euclidean space are perceived as WTS adjacent. Purely local analysis of point neighborhoods is insufficient to distinguish between points which are WTS adjacent and ones adjacent only in Euclidean space (Fig. 3). Thus, while human observers can easily envision the aggregate strips and curves corresponding to each cluster, and hence the isoline cross-sections orthogonal to these curves (Fig. 2cd), our algorithm needs to compute these cross-sections given only the raw strokes as input. We consequently face two interconnected challenges: we must compute optimal isovalues along each cluster cross-section while simultaneously computing the cross-sections themselves.

We robustly address both challenges at once by casting the computation of the parameterization and the underlying isoline cross-sections as a constrained variational problem (Sec. 3.2). We then solve it using a combined discrete-continuous optimization framework (Sec. 3.3). We first determine the optimal orientations of parameterization gradients along each stroke, converting monotonicity constraints into more tractable inequality ones (Sec. 4). We then jointly solve for cluster cross-sections and parameter values along them using a variational optimization framework (Sec. 5). Our method starts with an initial set of cross-sections that is expected to approximate the final one, but that might contain outliers. It narrows this set down using an iterated algorithm that solves for a joint cluster parameterization whose isolines are closely aligned with a subset of these initial cross-sections, and which satisfies a relaxed variant of the properties above. Finally, it uses this relaxed parameterization as a close initial guess for computing our desired arc length cluster parameterization. The resulting method robustly handles complex cluster configurations, including self-adjacent and self-intersecting clusters (e.g. the nose or left eyebrow in Fig. 1).

Sec. 7 validates our design choices. We compare our parameterizations and fitted curves to those produced by previous methods, highlighting our method's robustness in handling challenging inputs which cause prior methods to produce severely distorted results. We conduct a perceptual study which shows that observers prefer curves fitted by our method over those produced by the closest competitor 60.6% of the time, judge them as on par 34.6% of the time, and prefer the alternative just 4.8% of the time. We showcase our method's robustness by extending it to handle raster inputs

(Sec. 7.1.1). We demonstrate the versatility of our parameterizations by showcasing their suitability for a range of other applications.

These advancements are made possible by our novel approach to cluster parameterization, which leverages observations about human perception of such clusters and converts those into an actionable joint cluster parameterization method.

2 RELATED WORK

Sketch Processing. User demand has led to the emergence of numerous methods and tools for processing of vector-format artist-drawn sketches, including colorization [Adobe 2020; Orzan et al. 2008], shading [Finch et al. 2011; Shao et al. 2012], style transfer [Freeman et al. 2003], and editing [Igarashi et al. 2005], and has lead to increased use of free-hand vector sketches as inputs for high-level tasks such as 3D modeling [Bessmeltsev et al. 2016; Gryaditskaya et al. 2020; Lipson and Shpitalni 1996; Xu et al. 2014]. These methods typically target *clean* drawings where each stroke corresponds to a complete meaningful geometric curve and cannot be applied as-is to typical raw artist drawings which depict such curves using clusters of overdrawn strokes [Liu et al. 2018, 2015; Yan et al. 2020]; thus processing overdrawn sketches requires consolidating them first.

Vector Sketch Simplification and Consolidation. Vector sketch simplification methods, e.g. [Grabli et al. 2004; Nan et al. 2011], reduce visual clutter in detailed drawings by either removing or combining strokes. Vector sketch consolidation methods identify clusters of strokes that jointly correspond to meaningful intended curves and replace each cluster with its corresponding curve [Barla et al. 2005; Chen et al. 2013; Liu et al. 2018, 2015; Noris et al. 2012; Orbay and Kara 2011; Rosin 1994; Shesh and Chen 2008]. Recent research has made significant advances in detecting clusters that jointly depict intended curves. Once the strokes are clustered, these methods use either generic point-based or dedicated stroke-based fitting methods, discussed below, to fit aggregate curves to each cluster. StrokeStrip provides an alternative to these existing fitting methods. It makes no assumptions on the structure or other properties of the input clusters provided, and can be used in conjunction with any stroke clustering method.

Vectorization and Raster-Space Consolidation. Raster space consolidation methods (e.g. [Simo-Serra et al. 2018, 2016]) generate consolidated raster outputs from raster overdrawn images, and require a separate vectorization step to produce outputs suitable for downstream processing. Most vectorization methods convert raster inputs into vector form [Bao and Fu 2012; Bessmeltsev and Solomon 2019; Bo et al. 2016; Donati et al. 2017, 2019; Najgebauer and Scherer 2019; Noris et al. 2013; Parakkat et al. 2018b] with no attempt to simplify output topology. Recent works [Bartolo et al. 2007; Favreau et al. 2016; Stanko et al. 2020] aim to simultaneously consolidate and vectorize overdrawn raster sketches; however even when given individual clusters as input they frequently produce complex vector graph outputs (Sec. 7.1.1, supplementary material). While our method is designed for vector data, it can be extended to fit single curves to vectorized raster stroke clusters (Sec. 7.1.1).

(a) Input (b) [Liu et al. 2005] (c) [De Goes et al. 2011] (d) [Levin 2004] (e) Ours

Fig. 4. Given points densely sampled on input cluster strokes (a), methods for spline curve fitting [Liu et al. 2005] (b), 2D shape reconstruction [De Goes et al. 2011] (c), and MLS based fitting [Levin 2004] (d) fail to correctly recover the geometry (b) or topology (c,d) of the intended aggregate curves. Our method correctly recovers these intended curves (e).

(a) Input (b) [Orbay & Kara 2011] (c) [Liu et al. 2018] (d) Ours

Fig. 5. StrokeStrip (d) successfully parameterizes (parameter values visualized using a blue to red color scheme) and fits complex stroke clusters (a) on which prior methods ([Orbay and Kara 2011] (b) and [Liu et al. 2018] (c)) fail to compute correct stroke point orderings (visualized using a blue to red scheme) resulting in poor output fit quality. The ordering computed by Liu et al. [2018] fails to include the points colored in gray.

1D Parameterization and Curve Fitting to Points. Fitting curves to *ordered* or 1D parameterized points is a well understood problem [Farin 2014] amenable to many established methods (e.g. [Baran et al. 2010; McCrae and Singh 2009]). Our problem is distinctly different from this setting, as our input is only *partially* ordered. Specifically, while we can trivially order or 1D parameterize points sampled along an individual stroke, we have no predefined ordering between points sampled on different strokes. The isovalues of our joint 1D cluster parameterization define a complete ordering of points across all strokes, allowing for subsequent use of methods that operate on ordered or parameterized points (Sec. 7). Existing research on point ordering or 1D parameterization does so in the context of curve fitting, discussed below.

Multiple methods aim to reconstruct 2D shapes given *unordered* points, which are assumed to lie on or near these shapes (Fig. 4,b-d). Traditional fitting methods deform initial 1D curves [Farin 2014], represented explicitly [Liu et al. 2005; Wang et al. 2006] or implicitly [Yang et al. 2005], to best approximate the input points. The output of these methods is highly dependent on the degree of similarity between the initial and intended curves. Computing suitable initial curves for fitting points sampled from highly bent or self-intersecting shapes remains an open problem [Wang et al. 2006]. Thus, as noted by Orbay and Kara [2011], while these methods can be applied for fitting points sampled from simple low-curvature stroke clusters, they fail on inputs with high curvature or self-intersections (Fig. 4b). Classical 2D or 3D reconstruction methods ([De Goes et al. 2011; Dey et al. 1999; Goshtasby 2000; Kazhdan and Hoppe 2013; Kolluri et al. 2004; Lee 2000; Levin 2004; Parakkat et al. 2018a; Wang et al. 2014]) do not constrain output topology; when presented

with points sampled from our inputs, these methods often produce outputs that contain multiple connected components (e.g. Fig. 4c, bottom) and complex graph connectivity (Fig. 4cd). By arc length parameterizing each input cluster in 1D, we successfully and fully automatically fit each input, no matter how complex, using a *single* curve that is consistent with viewer expectations (Figure 4e).

Fitting Curves to Stroke Clusters. A number of methods specifically target curve fitting for stroke clusters. In contexts where live feedback can be provided to artists, fits can be incrementally corrected as artists add new strokes [Bae et al. 2008; Grimm and Joshi 2012]; however, in offline contexts, drawing order cannot be relied upon. Kara and Shimada [2007] and Orbay and Kara [2011] first order points sampled along the stroke clusters and use the obtained full ordering to fit polylines or smooth curves to these samples. Kara and Shimada [2007] project the points to the dominant principal axis of the cluster and use the order of these projections along the axis as the point order. This approach fails even on relatively simple inputs (see inset). Orbay and Kara [2011] use a Laplacian spectral embedding to order the sampled points (Figs. 1c, 5b). Liu et al [2018] propose a tangent aware Moving Least Squares (MLS) based approach for fitting curves to stroke clusters. To fit a single curve to the MLS output points, they first connect the points using a directed spanning tree, and then extract and smooth the longest path in this tree (Figs. 1d, 5c, points not included in the paths in gray). These methods often produce non-monotonic and counter-intuitive point orderings on clusters with sharp turns and narrow bottlenecks, leading to fitting outcomes inconsistent with viewer expectations (Figs. 1bc, 5bc). Our comparisons (Figs. 1,5, Sec. 7) demonstrate that StrokeStrip robustly computes the desired parameterizations and corresponding fits in complex scenarios where all prior methods fail.


[Kara & Shimada'07]

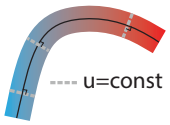
3 OVERVIEW

3.1 Problem Statement

We aim to robustly compute the aggregate curves viewers perceive when presented with input vector stroke clusters. Prior literature [Liu et al. 2018], observations of manually traced curves [Liu et al. 2018; Yan et al. 2020], and our study (Appendix A), suggest that viewers perceive clusters as varying width strips formed by sweeping straight rulings, or *cross-sections*, along paths that follow these aggregate curves. We speculate that observers mentally envision the geometry of these curves by interpolating shape properties along such strip cross-sections (Fig. 2). This observation suggests that our fitting problem can be recast as one of computing a 1D parameterization of the cluster whose isolines define these cross-sections. Given this desired parameterization, we can express the shape of the fitted curve at each parameter value u as a function of the shape properties of the stroke points along the u-isoline cross-section (Sec. 6). In addition to facilitating fitting, our parameterizations provide a natural mechanism for jointly manipulating the strokes within the cluster (Fig. 1f, Sec. 7).

To formalize the desired properties of the parameterization we seek, we first formalize the notion of a strip itself. A strip is well-described by its centerline $\gamma(t) \in \mathbb{R}^2$, parameterized by its arclength $t \in [0, L]$, and width $W(t)$. Reminiscent of ruled surfaces in 3D, the strip is then defined as the set of points swept by a moving line segment of length $W(t)$, centered at and perpendicular to the centerline. The two sides of a strip are then the two curves at the distance of $W(t)/2$ from the centerline in the normal direction, or more formally $\gamma(t) \pm \frac{W(t)}{2} n(t)$.

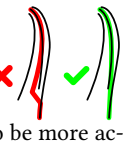
While this definition addresses non-self-intersecting strips, in general strips can and often do self-intersect. This technical difficulty, however, is easy to overcome by considering the curvilinear strip as an image of a locally injective map of an axis-aligned strip, namely the parameterization domain. We omit this for brevity; this reasoning, as well as our algorithm, admits self-intersecting strips.



This construction naturally defines the *arc length parameterization* of a strip as the extension of the arc length parameterization of the centerline to the whole area of the strip via 'extrusion' in the normal direction (see inset). Simply put, it is a function $u(x)$ defined for all points in the strip, such that straight line segments $C(t) = \{\gamma(t) + an(t) | a \in [-\frac{W(t)}{2}, \frac{W(t)}{2}]\}$ are the parameterization's isolines; that is, they share the same parameter value $t = u(x), x \in C(t)$.

The arc length parameterization of a given a strip defined by its centerline and width can be explicitly computed (Appendix B). A trivial, yet important, property of the arc length parameterization of a strip is that, for any point on the isoline $C(t)$, the parameterization gradient is parallel to the centerline tangent $\tau(t)$. The norm of the gradient, while unit at the centerline, changes depending on the distance from the centerline and centerline curvature. The average of parameterization gradients over each isoline has unit length and thus coincides with the centerline tangent (see Appendix B for a proof).

Returning to the problem of stroke cluster parameterization, given the *oriented* strokes $S = \{s_i, i = 1, \ldots, N\}$, we are looking for a map $u(x) : \cup_i s_i \rightarrow [0; L]$, where L is unknown that is the restriction of the arc length parameterization of the viewer-perceived strip.



An important subtlety in parameterizing stroke clusters lies in correct generalization of the notion of strip centerline. Prior sketch processing research [Liu et al. 2018; Xu et al. 2014] repeatedly suggests that viewers perceive the tangents of drawn strokes to be more accurate than their exact locations. This suggests that viewers do not expect the aggregate curves, or strip paths, to follow the cluster's centerline, i.e. to be strictly equidistant from the cluster's outermost side strokes (inset, red), but rather, expect them to align with the stroke tangents (inset, green). Based on these observations, we prioritize tangent alignment over centrality when computing the strip parameterizations and paths.

Thus we look for a stroke cluster parameterization that is a restriction of the arc length parameterization of the aggregate strip. While the strip itself is unknown, the observations above suggest that its restriction to the strokes in the cluster should satisfy the following properties:

Tangent Alignment: The gradient of the arc length parameterization of a strip is parallel to the path tangent; the path tangent is, in turn, expected to be closely aligned with the stroke tangents along each isoline. We thus expect all pairs of WTS adjacent points p, p' belonging to different strokes, where $\overline{p,p'}$ is orthogonal to the path tangent, to have the same iso-value u; in other words, we expect the parameterization isolines to be approximately *orthogonal* to the stroke tangents at stroke points along the isoline.

Arc Length Preservation: The gradient of the arc length parameterization of a strip, averaged over an isoline, has unit norm. We expect the same property from the arc length parameterization of a cluster.

Monotonicity: The arc length parameterization of a strip is, by definition, strictly monotonic along its gradient direction. We thus expect the parameter values u to strictly monotonically increase (or decrease) along each individual stroke. We note that, combined with tangent alignment, monotonicity implies that the gradients of u at all stroke points along each isoline should have similar directions.

Isoline Span: We define pairs of points p, p' on different strokes as *side-by-side* if the line $\overline{p,p'}$ is orthogonal to either $\tau(p)$ or $\tau(p')$. Perception and consolidation research [Liu et al. 2018] suggest that, absent evidence to the contrary, viewers perceive side-by-side stroke points with similar stroke tangents as WTS adjacent . Combined with tangent alignment, the isoline span property suggests that, absent a conflict with the properties above, we expect all side-by-side pairs of points to be adjacent within the strip, and thus to have similar u values in our parameterization.

We formally express these requirements as a combination of an objective function and a set of constraints, described next, and refer to a parameterization satisfying these requirements as *joint*.

3.2 Formulation

We formulate our objective using the following derivation. For a non self-intersecting strip $\mathcal{S} \subset \mathbb{R}^2$, its arc length parameterization $u : \mathcal{S} \to \mathbb{R}$ is the minimizer of the following variational problem:

$$\min_u \int_0^L \left\| \frac{1}{W(t)} \int_{C(t)} \nabla u(x) dx - \tau(t) \right\|^2 dt, \tag{1}$$

where the outer integral is with respect to t, the arc length of the strip centerline; the inner integration is over each straight line cross-section $C(t)$ with length $W(t)$, crossing the strip perpendicular to the centerline, as defined in Sec. 3.1 (see Appendix B for a proof). However, applying this formulation as-is to strokes is problematic since ∇u is not clearly defined. We recall that the arc length parameterization satisfies the following properties: its gradient is always parallel to the centerline tangent, and its average over each gradient orthogonal cross-section has unit length. Thus instead of minimizing Eq. (1) we can equivalently minimize the projections of the expression inside the norm onto the centerline tangent and normal. Since for the arc length strip parameterization $\nabla\, u(x) \cdot n(t) = 0$ at every point x and its corresponding centerline parameter t, and $(\frac{1}{W(t)} \int_{C(t)} \nabla u(x) dx - \tau) \cdot \tau = \frac{1}{W(t)} \int_{C(t)} \nabla u(x) \cdot \tau dx - 1$, the expression for u in Eq. 1 has the same minimizer as the following:

$$\int_0^L \left| \frac{1}{W(t)} \int_{C(t)} \nabla u(x) \cdot \tau(t) dx - 1 \right|^2 dt + \int_0^L \int_{C(t)} |\nabla u(x) \cdot n(t)|^2 dx dt$$

These terms are much more amenable to computation on stroke clusters, since $\nabla u(x) \cdot \tau(t)$ can be well approximated by the directional derivative of u along each stroke, and $\nabla u(x) \cdot n(t)$ can be approximated using samples on different strokes that are WTS adjacent along the normal direction.

This formulation relies on the notion of a path tangent τ and a normal n, which are unknown. However, if we were given the cross-sections $C(t)$, we could approximate τ as the averages of stroke tangents $\overline{\tau}$ along each cross-section and replace the normal n with the vector perpendicular to $\overline{\tau}$, denoted $\overline{n}$. Our formulation then becomes,

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

$$E_{\text{align}} = \int_0^L \int_{C(t)} |\nabla u(x) \cdot \overline{n}(t)|^2 dx dt \tag{3}$$

$$\min_u E_{\text{arc}} = \min_u (E_{\text{length}} + E_{\text{align}}) \tag{4}$$

Here E_{length} promotes arc length preservation, while E_{align} promotes tangent alignment.

3.2.1 Discretization. We discretize the combined energy function E_{arc} and solve for the parameter values at densely sampled points along the cluster strokes that jointly minimize it as follows.

Notation: We discretize input strokes as densely-sampled polylines (Appendix F). A sample on an input stroke s then can be described as $\{s, i\}$, where i refers to the sample index. We denote the sample position as $p_{s,i}$, its unit tangent as $\tau_{s,i}$, and the unknown parameter value as $u_{s,i}$. For brevity, when referring to generic point samples, we drop the indexing and use p to describe positions and $\tau(p)$ to describe unit tangents. Finally, we encode a cross-section $C(u)$ as a sequence of point samples $C = \{a\} = \{(s, i)\}$.

We convert the integral formulation above into a finite summation by using a finite set of isoline cross-sections $C(u)$, defined for a dense set of parameter values u. We define the cross-section tangent as:

$$\overline{\tau}(C) := \frac{\sum_{(s,i)\in C} d_{s,i}\ o_s \tau_{s,i}}{\left\| \sum_{(s,i)\in C} d_{s,i}\ o_s \tau_{s,i} \right\|} \tag{5}$$

Here $o_s \in \{1, -1\}$ is a per-stroke direction variable indicating whether the parameterization gradient at stroke s is expected to be ascending ($o_s = 1$) or descending ($o_s = -1$) with respect to the original stroke orientation. Each tangent in this sum is weighted by the size of the dual cell of the corresponding point along the cross-section: If $(s', i'), (s, i), (s'', i'')$ are three consecutive points along the cross-section, then $d_{s,i} = \frac{1}{2}(\|p_{s',i'} - p_{s,i}\| + \|p_{s,i} - p_{s'',i''}\|)$.

The finite difference discretization of Eq. 2 then yields:

$$E_{\text{length}}(u) = \sum_C \left(\sum_{(s,i)\in C} \frac{d_{s,i}}{D_C} \frac{(u_{s,i} - u_{s,i-1})\tau_{s,i}}{l_{s,i}} \cdot \overline{\tau}(C) - 1 \right)^2, \tag{6}$$

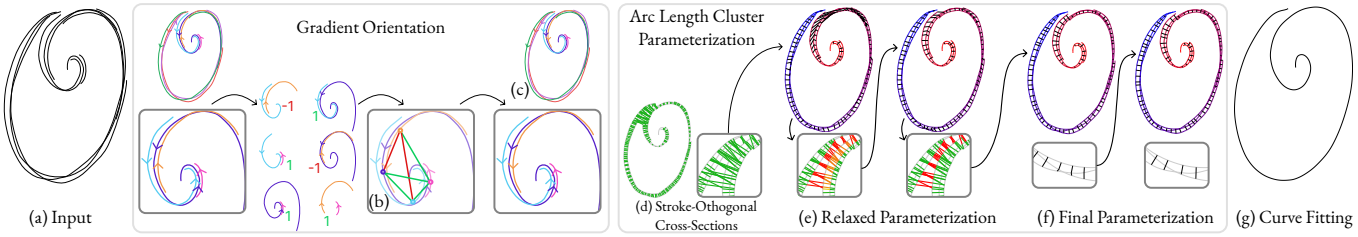


Fig. 6. Overview: StrokeStrip first computes compatible parameterization gradient directions (c) for all cluster strokes (a) by formulating direction choice as a binary labeling problem on a graph (b) whose nodes correspond to strokes, and whose positive and negative edge scores reflect the degree of affinity between them. It then jointly parameterizes the cluster using a variational framework (d-f); starting with a initial set of stroke-orthogonal cross-sections (d) it gradually obtains both a *relaxed* parameterization and pairwise likelihoods (green to red) for side-by-side points on these cross-sections of being WTS adjacent. It uses this relaxed parameterization as a starting point to compute our final arc length parameterization (f), used to compute our fitted curves (g).

where we normalize the sum by the length of the cross-section $D_C = \sum_{(s,i)\in C} d_{s,i}$, i.e. a discretization of $W(t)$, and $l_{s,i}$ is the length of a segment along the polyline: $l_{s,i} = \|p_{s,i} - p_{s,i-1}\|$.

The discretization of Eq. 3 yields:

$$E_{\text{align}}(u) = \frac{1}{2}\sum_{C}\sum_{a,b\in C}(u_a - u_b - \overline{\tau}(C)\cdot(p_a - p_b))^2 \tag{7}$$

We solve for the u values that minimize

$$E_{\text{arc}} = E_{\text{length}} + E_{\text{align}} \tag{8}$$

subject to monotonicity constraints, expressed as

$$(u_{s,i} - u_{s,i-1})o_s \geq \frac{l_{s,i}}{2} \;\forall\, s, i. \tag{9}$$

Our choice of a right-hand side value that reflects the local sampling density along each stroke makes the constraints robust to uneven sampling density, and encourages more uniform velocity.

We fix the parameter interval in place, by constraining the parameter value of a single stroke point $u_{0,0} = 0$.

3.3 Solution Overview

Computing the minimizer of the constrained optimization problem above requires solving for three sets of variables: the parameter values at all stroke points, the gradient orientations along strokes, and the sets of stroke points on each isoline cross-section. Accounting for the two latter sets of discrete variables while computing the parameterization gives rise to a difficult mixed discrete-continuous optimization problem which cannot be addressed using standard optimization strategies. We obtain the desired minimizer by employing a tailored solution mechanism that leverages the unique properties of our problem.

We recall that tangent alignment requires stroke gradient orientations along each isoline to be similar, and that the isoline span property suggests that *most* side-by-side points on adjacent strokes should have similar isovalues. Since stroke tangents are expected to change gradually, these properties suggest that the oriented tangents at most side-by-side points should point in the same direction (Figs. 6b, 7). We use these observations to compute gradient orientations o_s along all strokes as a first step toward computing the parameterization (Sec. 4, Fig. 6b). We obtain the orientation choices that optimize directed tangent similarity across all pairs of side-by-side points, and are therefore likely to allow for maximally wide isoline span. We formulate the computation of the desired orientations as a binary graph labeling problem, where the edge weights reflect the likelihood of pairs of strokes to share side-by-side WTS adjacent points . With the orientations computed, our monotonicity constraints (Eq. (9)) become simple inequalities, simplifying the rest of the processing.

Our core parameterization step computes the desired arc length cluster parameterization by solving for the cross-section isolines $C(u)$ and the isovalues along them (Sec 5). Specifically, we first identify an initial set of side-by-side points likely to be WTS adjacent by forming *stroke-orthogonal cross-sections* (Fig. 6d, Sec. 5.1). These cross-sections join together points that are largely expected to have similar isovalues in our target parameterization, but that may not satisfy tangent alignment. We solve for a *relaxed* joint parameterization that is arc length preserving and assigns similar values to WTS adjacent points on these cross-sections, subject to monotonicity constraints (Sec. 5.2). Specifically, we associate WTS adjacency likelihood variables with each pair of points along the cross-sections and simultaneously solve for parameter values and likelihoods. The minimizers of our relaxed formulation closely approximate the optimal solutions we seek, and we therefore use them as initial guesses for a local optimization step that generates our desired final parameterizations (Sec. 5.3).

4 GRADIENT ORIENTATION

The first stage of our algorithm assigns a gradient orientation $o_i \in \{-1, 1\}$ to each stroke i in the cluster. Intuitively, we seek orientations that maximize tangent alignment, i.e. result in similar tangent directions $o_i\tau(p_i)$ and $o_j\tau(p_j)$ at side-by-side points p_i and p_j on strokes i and j that viewers perceive as WTS adjacent (Fig. 7). While we do not a priori know which side-by-side points are WTS adjacent, we aim to maximize isoline span by maximizing the *overall* number of side-by-side points with similar tangent directions. Maximizing tangent direction similarity at such points increases the likelihood of these points having similar isovalues in the output parameterization (Figs. 7, 8).

We note that any strip allows for two equal quality parameterizations with inversely oriented gradients. We address this degree

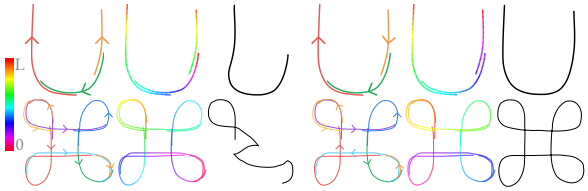


Fig. 7. Gradient orientations along individual strokes drastically impact the properties of the resulting parameterization: Forcing the parameterization gradients to align with arbitrary input stroke orientations (indicated by arrows) (a) leads to parameterizations with distinctly different u values at points viewers perceive as WTS adjacent (b) , and leads to fitting outcomes inconsistent with viewer expectations (c). Our joint orientation method consistently orients strokes perceived as WTS adjacent (d), leading to better parameterization (e) and fits (f).

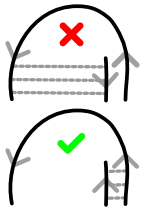
of freedom by arbitrarily setting $o_0 = 1$. Our problem can now be recast as one of solving for $o_1, \ldots, o_N$ that, jointly with o_0, best satisfy the requirements above. In the following, we first consider the optimal choice of orientations for a cluster consisting of a single stroke pair (Sec. 4.1, Fig 8), then generalize the problem formulation and solution method to clusters with multiple strokes (Sec. 4.2).

4.1 Optimal Pairwise Orientation.

Given a pair of strokes s_i and s_j, we keep o_i fixed and cast the problem of computing o_j as one of finding the pairwise orientation compatibility $o_{ij} \in \{-1, 1\}$ (where 1 indicates similar and -1 indicates opposite) such that the j-th stroke orientation $o_j = o_{ij} o_i$ maximizes the quality of the parameterization of a cluster that consists of only these two strokes . We choose the optimal solution by evaluating the properties of the parameterizations that each of the two choices induces, accounting for the following considerations, illustrated in Fig. 8:

Tangent Compatibility: We consider a parameterization whose isoline cross-sections include points on both s_i and s_j to be *compatible* if the angle between the oriented tangents $o_i \tau_i(p)$ and $o_{ij} o_i \tau_j(p')$ along all cross-sections $C = (p, p')$, where $p \in s_i$ and $p' \in s_j$, is below 90° (Fig. 8ac). This requirement follows from the combination of our monotonicity and tangent alignment properties.

Isoline Span: To best satisfy the isoline span property, we prioritize tangent-compatible parameterizations with isolines that span both strokes over those that do not (Fig. 8a).



Narrowness: Given two tangent compatible parameterizations induced by the two choices of o_{ij} that include cross-sections that span both strokes (see inset), we prioritize the parameterization with the shortest cross-section. This preference is motivated by the expectation that human observers interpret clusters as narrow strips [Liu et al. 2018]; thus, given two alternatives, they prefer a more narrow interpretation (Fig. 8b).

As we prefer parameterizations that satisfy the isoline span requirement, we first assess if each orientation choice o_{ij} allows for a tangent compatible parameterization whose cross-sections span

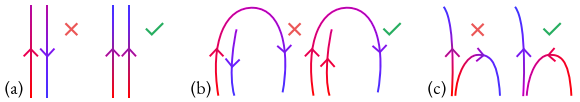
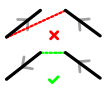


Fig. 8. Stroke pair configurations with preferred (right) and sub-optimal (left) orientation choices (arrows show orientation). Coloring visualizes the joint parameterization imposed by each order. (a) the preferred orientation results in a parameterization that is tangent compatible and better satisfies isoline span; (b) the preferred orientation induces a more narrow strip; (c) the preferred orientation results in a tangent compatible parameterization.

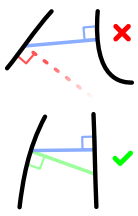
both strokes. If both allow for compatible parameterizations, we choose the orientation that induces more narrow strips (Fig 8b). If only one orientation allows a compatible parameterization that satisfies the isoline span property, we use this orientation choice (Fig 8a).



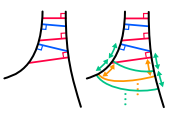
If no orientation choice allows for a parameterization that satisfies both properties, we locate the closest pair of endpoints on the two strokes i and j, and compute o_{ij} that minimizes the difference between the oriented tangents at these endpoints (see inset). This choice induces a strip geometry in which one stroke is the continuation of the other (Fig 8c).

Proxy Parameterization. A possible approach to evaluate the impact of each choice of o_{ij} is to directly compute the parameterization it induces (Sec. 5), and then assess the result with respect to the properties above. Such a computation is time consuming, and can be replaced by a much faster proxy parameterization computation that is sufficient for our needs. Specifically, we note that the criteria above are based only on cross-section width and span, and tangent similarity at cross-section endpoints. Accordingly, instead of computing a full parameterization, we form a set of tentative cross-sections and use them for assessing both properties.

For each choice of o_j, we first compute tangent compatible, stroke-tangent-aligned cross-sections. We then extend the proxy parameterization they induce along the rest of the strokes by introducing additional well-spaced cross-sections. Since tangent alignment advocates for cross-sections to be orthogonal to the strokes, we form our first set of cross-sections by shooting orthogonal left and right rays from evenly sampled points p on each stroke $s \in s_i, s_j$ and recording the intersections



p' of these rays with the other stroke. We add p, p' to the set of cross-sections if the following is satisfied: (1) the oriented tangents at the two points p and p' satisfy tangent compatibility, and (2) the orthogonal to the second stroke at p' intersects the stroke s. If this test fails (see inset, p, p' in blue), p is not viewed by observers as being next to p'. If the set of cross-sections computed this way is empty, the assessed orientation does not allow for a tangent-compatible parameterization.

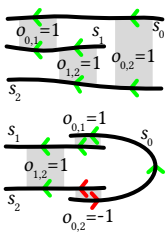


Given a non-empty set of stroke-orthogonal cross-sections (inset, blue and red) we leverage the expectation that output parameterization should be arc length preserving to introduce cross-sections through all points p on the two strokes than are not currently part of a cross-section (orange and cyan in the inset). For each such point p, we locate the nearest cross-section endpoint $\overline{p}$ along its underlying stroke s. We use the cross-section $\overline{p}, \overline{p}'$ to pair the point p with a point p' located at the same signed arclength distance

from $\overline{p}'$ as p is from $\overline{p}$ (see inset, green and orange,right). If cross-sections added this way violate tangent compatibility, the assessed orientation does not allow for a compatible parameterization.

4.2 Global Solution

A brute force approach for obtaining the orientations resulting in the best parameterization for larger clusters would be to compute proxy parameterizations for each possible orientation combination and to select the best combination using similar criteria to ones above. However, this approach is clearly too computationally expensive. We compute a set of orientations that leads to high quality parameterizations by noting that, given a cluster with more than two strokes, pairwise preferences, computed independently for each pair of strokes in the cluster, are strongly suggestive of the globally optimal orientation choice. More specifically, when all pairwise preferences are compatible (inset, top), using these preferences as-is leads to the desired outcome; however, when the preferences are globally contradictory (inset, bottom), we can still reliably predict which pairwise choices are more likely to impact the global outcome by assessing the geometry of each pair as detailed below. These observations motivate our solution approach: we first compute compatible orientations for each pair of strokes in the cluster (Sec. 4.1). We then use these pairwise choices and their predicted impact on the overall parameterization quality to compute the orientation choices across the entire cluster.



Formulation. For each pair of strokes, we compute pairwise orientation compatibilities $o_{ij} \in \{-1, 1\}$ as described in Sec. 4.1. We associate with each pair an impact score w_{ij}, indicating how critical satisfying orientation compatibility for this pair is for optimizing the quality of the overall output parameterization. Using these values, the optimal set of orientations is given by the solution to the following mixed-integer quadratic programming (MIQP) problem:

$$\min_{o_i \in \{-1,1\}} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} o_{ij} w_{ij} (o_i - o_j)^2 \tag{10}$$

We compute the optimal variables o_i using the integer programming solver provided by the Gurobi optimization toolbox [2020]. For simplicity, instead of keeping track of the per-stroke orientations in the rest of the computation, for each stroke with $o_i = -1$, we flip the stroke orientation. In the rest of the computation, we use the directed stroke tangent as-is for all computations and require the gradient of u to be **positive** along all strokes.

Pairwise Impact Scores. When determining the importance of each individual pair satisfying the pairwise orientation relation computed in Sec. 4.1 within the global parameterization, we consider similar factors to those used in consolidation literature when assessing how likely strokes are to belong to the same cluster [Liu et al. 2018]: tangent similarity, distance, and the relative length of their side-by-side sections compared to their overall length. As we aim for a globally tangent-aligned parameterization, we prioritize pairs which demonstrate stronger tangent alignment along shared cross-sections; since we aim for narrow clusters, we prioritize pairs with shorter cross-sections; and lastly, we prioritize pairs which have a larger percentage of points on shared cross-sections. Given all cross-sections computed in Sec. 4.1, which we denote by Ω, we compute the average angle between cross-section endpoint tangents as

$$\phi_{ij} = \frac{1}{|\Omega|} \sum_{(a,b)\in\Omega} \arccos\left(o_{ij} \tau_{i,a} \cdot \tau_{j,b}\right).$$

We set the alignment score $\widetilde{w}_{i,j}$ as,

$$\widetilde{w}_{i,j} = \begin{cases} 1, & \phi < \alpha_{min} \\ e^{\frac{-(\phi-\alpha_{min})^2}{2\sigma^2}}, & \alpha_{min} \le \phi < \alpha_{max} \\ 0, & \alpha_{max} \le \phi \end{cases} \tag{11}$$

Following perception studies that indicate that observers see lines as parallel when the angle between them is under 20° [Hess and Field 1999], we set $\alpha_{min} = 10°$ and $\alpha_{max} = 20°$, and use the three sigma rule to set $\sigma = (\alpha_{max} - \alpha_{min})/3$ to allow a smooth fall off. If $|\Omega| = 0$, we set $\widetilde{w}_{i,j} = 0$.

We define the impact score w_{ij} as

$$w_{ij} = \frac{\widetilde{w}_{i,j} + \varepsilon}{1 + \min_{(a,b)\in\Omega} \|a - b\|^2} \max\left\{\frac{|\Omega|}{\min(N_i, N_j)}, \frac{1}{2}\right\}$$

where $\varepsilon = 10^{-2}$ and N_i, N_j are the total numbers of samples on the respective strokes used in our parameterization computation above.

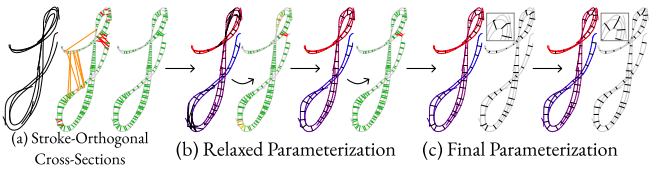


Fig. 9. Arc Length Cluster Parameterization: (a) Initial stroke-orthogonal cross-sections (green), sub-sampled for visual clarity; red and orange cross-sections filtered out due to violating tangent similarity and inter-stroke spacing respectively; (b) relaxed parameterizations and pairwise WTS adjacency likelihoods (red low, green high) across solution iterations; (c) final isolines and parameterization.

5 ARC LENGTH CLUSTER PARAMETERIZATION

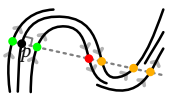
The energy function we aim to minimize (Eq. 8) is non-linear and, in the general case, non-convex. Most importantly, this function is defined via the cross sections which are *a priori* unknown; consequently, minimizing it — or even assessing it — requires identifying points on different strokes that belong to the same cross-section. To identify cross sections, we face two chicken and egg problems. First, we expect the cross-sections to be orthogonal to the parameterization gradient, but we need a parameterization to compute this gradient. Second, we expect our cross-sections to lie entirely inside the envisioned strips; however, we do not *a priori* know the strip geometry, and thus the strip outlines.

We address these challenges with a three-step process. We first note that tangent alignment suggests that stroke tangents can often serve as a plausible proxy for parameterization gradients. To this end, we initialize our parameterization computation using *stroke-orthogonal cross-sections*, rather than gradient orthogonal

ones (Sec. 5.1, Fig. 9a). To maximize isoline span, we form maximally long cross-sections, absent evidence of them connecting points which are highly unlikely to be WTS adjacent (Fig 9a). We expect most, if not all, points on these cross-sections to be WTS adjacent, and thus aim for them to have similar isovalues; we do not, however, expect these cross-sections to be strictly gradient orthogonal. Motivated by these observations, we introduce a *relaxed*, outlier-robust parameterization formulation that aims to jointly minimize both the arc length distortion along the cluster strokes and a relaxed alignment term, which seeks to assign similar values to a maximal subset of point pairs on common cross-sections which are deemed as likely to be WTS adjacent (Fig. 9b, Sec. 5.2). The parameterization that minimizes this relaxed energy serves as a suitable initial guess for computing our target joint parameterization which strictly satisfies tangent alignment (Sec. 5.3, Fig. 9c).

We extend the method to detect and parameterize clusters that jointly depict closed curves in Appendix D.

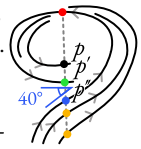
5.1 Stroke-Orthogonal Cross-Sections



To form stroke-orthogonal cross-sections, we densely sample points p along each stroke (black in the inset, gray arrows indicate tangent directions), shoot stroke-orthogonal left and right rays from each sampled point, and compute the intersections of these rays with all cluster strokes. Notably, some of these intersections may not be WTS adjacent to the source point p . We filter such points out using a combination of local and contextual filters.

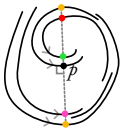
Specifically, we form initial stroke-orthogonal cross-sections $C(p)$ comprised of the sample p and all ray-stroke intersections p' that satisfy the following three criteria (green in the inset):

Tangent Compatibility: As in Sec. 4.1, we only include ray-stroke intersections p' in $C(p)$ if the angle between the oriented stroke tangents at p and p' is smaller than 90° (in the insets, the red intersections do not satisfy this property).



Tangent Similarity: Perceptual literature [Hess and Field 1999] and consolidation research [Liu et al. 2018] indicate that human observers are unlikely to see lines as parallel if the angle between them exceeds 20°. We therefore only include ray-stroke intersections p' in $C(p)$ if the angle between the oriented stroke tangents at consecutive points p' and p'' is below this threshold (in the inset, the blue intersection does not satisfy this property).

Connectedness: We expect strips and hence cross-sections to be continuous. Thus, given three consecutive points a, b, c along the ray, if a and b are *not* WTS adjacent, then neither are a and c. We therefore only include intersections p' in C if all ray-stroke intersections along the segment $\overline{pp'}$ satisfy both alignment and similarity.



Spacing: This local filtering resolves the most common scenarios of cross-sections extending outside the intended strip, but does not address cases where a ray intersects two or more separate similarly oriented sections of an intended strip (see inset). We require a principled way to identify and resolve such configurations. We note that human observers are likely to separate parallel stroke groups from one another when the distance between these groups along the stroke orthogonal direction is significantly bigger than the inter-stroke distances within the groups [Liu et al. 2018; Wagemans et al. 2012].

In our context, this observation argues for separating distinct side-by-side strip sections by assessing distances between pairs of consecutive intersections $\{p', p''\}$ along the rays; if the distance $d_{p',p''} = \|p' - p''\|$ between one pair of points is much larger than the distances between other consecutive pairs in its vicinity, this suggests that the segment $\overline{p', p''}$ is likely to be outside the strip. We employ a conservative version of this criterion, and compare the distance between consecutive ray intersections to the widths of entire adjacent cross-sections. We define the *neighborhood* of a cross-section C of width W as the set of all cross-sections with points at a distance less than W from any $c \in C$. We compute the median cross-section width W_c within this neighborhood. For each pair of consecutive intersections $\{p', p'\}'$ in C where p' is closer to the ray origin p, if $d_{p',p''} > W_c(1 + S_{max})$ we remove p'' (pink in the inset) from the cross-section (we set $S_{max} = .2$). We satisfy connectedness by removing from C all points along the ray that are further away from the origin than p''.

5.2 Relaxed Joint Parameterization

Our initial set of stroke-orthogonal cross-sections defines, for each stroke point, a set of side-by-side potentially WTS adjacent points on neighboring strokes. We use these sets to compute a *relaxed* joint parameterization, which promotes arc length preservation along each strokes and minimizes parameter value differences between side-by-side points on different strokes which are likely to be adjacent. This then serves as a good initial guess for our target joint parameterization (Sec. 5.3).

We formulate the goal of our relaxed joint parameterization u as:

$$\min_u E_{\text{relaxed}} = E_{\text{length}}(u) + E_{\text{similar}}(u) \tag{12}$$

$$E_{\text{similar}}(u) = \frac{1}{2} \sum_C \sum_{a,b \in C} L(a,b)(u_a - u_b)^2, \tag{13}$$

where E_{length} is defined by Eq. 6, $L(a,b)$ measures the likelihood of the points a, b to be WTS adjacent, and $E_{\text{similar}}(u)$ promotes assigning similar isovalues to side-by-side points which are likely to be WTS adjacent.

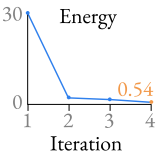
Our formulation of $L(a,b)$ is motivated by the following observations. Since strips allow for a strictly arc length preserving parameterization, we expect the values of both our target and relaxed energies to evaluate to near zero given the optimal parameter values u. Thus, given either the target or relaxed parameterizations, we expect the parameter space distance between nearby WTS adjacent points to be close to zero; ergo the farther apart such points are in parameter space, the less likely they are to be WTS adjacent . Thus, the difference between u values at points along the same stroke-orthogonal cross-section can be used as a predictor of WTS adjacency between them. Based on this observation, given a pair of points a and b along a common stroke-orthogonal cross-section C, we define the likelihood that these points are WTS adjacent as:

$$L(x_a, x_b) = e^{\frac{-(u_a - u_b)^2}{2\sigma^2}}. \tag{14}$$

We set $\sigma = L/30$, using the three-sigma rule: we consider points with u values more that 1/10th of the entire parameter space length apart as unlikely to be adjacent.

Solution. We compute $\tau(C)$ on the initial cross-sections and keep it fixed throughout the iterations. We then find the parameterization u that minimizes E_{relaxed} by using alternating local-global optimization iterations. Our local step computes the likelihoods $L(a, b)$ for every pair of points $(a, b) \in C$ based on the current parameter values u. Our global step keeps these likelihoods fixed, and uses them to compute the parameter values u that minimize E_{relaxed} (Eq. 12). With $\tau(C)$ and $L(a, b)$ fixed, the energy function minimized in each global iteration becomes a simple quadratic function, which needs to be minimized subject to linear inequality constraints. We solve the QP problems posed by each iteration using the interior-point method implementation in the CVX optimization toolkit [Grant and Boyd 2008, 2014].

The steps are iterated until convergence; specifically, we terminate once the largest change in parameter values u, across all participating stroke points, drops below 1/2 of the stroke's width. Our method typically takes at most seven iterations to converge. The average per-cross-section energy in each step for the cluster in Fig. 9b is shown in the inset.



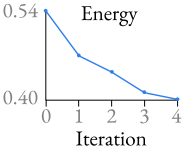
Initialization. To obtain an initial outlier-robust guess for u, we minimize E_{relaxed} using a set of fixed small likelihood values $L(a, b) = L^0 = 10^{-5}$, and rewriting E_{similar} (Eq. 7) to use the L_1-norm, which is known to be less sensitive to outliers:

$$E_{\text{similar}}(u) = \frac{1}{2} \sum_{C} \sum_{a,b \in C} L^0 |u_a - u_b|$$

This formulation strongly prioritizes arc length preservation (Figs 6e, 9b). Notably since the minimizer of E_{relaxed} is expected to minimize both E_{length} and E_{similar} independently, we expect the values u_a and u_b in the output to only diverge in the vicinity of stroke-orthogonal cross-sections that join points which are not WTS adjacent.

5.3 Final Parameterization

We use the relaxed joint parameterization computed in the previous step as an initial guess for minimizing E_{arc} (Eq. 8) subject to monotonicity. At each iteration, we form cross-sections $C(u)$ by evenly sampling the current parameterization u and grouping all stroke points with the corresponding isovalues. We solve the resulting QP problems using the mechanism described in Sec. 5.2. We require at most two iterations to converge to a desired solution, using the same convergence criterion as in Sec. 5.2 (Figs 6c, 9, 10). The average per-cross-section energy for the cluster in Fig. 9c is shown in the inset, with the result of Sec. 5.2 as iteration 0.



6 AGGREGATE CURVE FITTING

We aim to fit aggregate curves $\widetilde{X}(u)$, $u \in [0, L]$ to input clusters such that the curve shape at a parameter value u is reflective of the average shape of the cluster's strokes at the u iso-value. Prior literature [Liu et al. 2018; Xu et al. 2014] suggests that viewers perceive

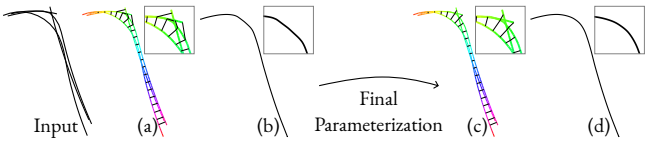


Fig. 10. Starting with a near-optimal parameterization, (a) we minimize E_{arc} reducing arc length distortion and producing straight, gradient orthogonal isolines (c), leading to improved fitting outcomes (b,d).

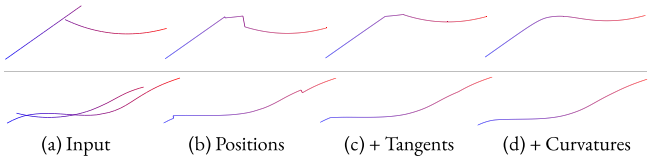


Fig. 11. Fitting alternatives: accounting for curvature, positions and tangents (d) produces fits more consistent with viewer expectations than when accounting for only positions (b) or positions and tangents (c).

the tangents of artist drawn strokes as more accurate than their exact location, motivating us to explicitly account for tangents in our framework, and to prioritize tangent alignment over average position preservation. Our observations further indicate that viewers expect the curvature of the fitted curve to be reflective of that of the underlying strokes. In particular, while the average tangent orientation can change abruptly near stroke endpoints (Fig. 11a), human observers expect the change in the tangent of the fitted curve (its curvature) to be more gradual when the curvature of these strokes is low (Fig. 11d).

We consequently formulate curve fitting as an optimization balancing three terms: curvature preservation, tangent alignment, and average position preservation (Fig. 11d). In other words, we seek a curve $\widetilde{X}(u)$ that minimizes

$$\widetilde{X}(u) = \underset{\widetilde{X}}{\operatorname{argmin}} \int_0^L \left(\left\| \widetilde{X}(u) - \overline{p}(u) \right\|^2 + \lambda_1 \left\| \widetilde{T}(u) - \overline{\tau}(u) \right\|^2 + \lambda_2 \left\| \widetilde{K}(u) - \overline{k}(u) \right\|^2 \right) du \qquad (15)$$

Here, $\widetilde{T}(u)$ and $\widetilde{K}(u)$ are the unit length tangent and curvature of the curve $\widetilde{X}(u)$ at u, λ_1 and λ_2 are weights controlling the relative impact of the different terms, $\overline{\tau}(u)$ is the average of the tangents of stroke samples along the u isoline $C(u)$, computed per Eq. (5), $\overline{p}(u)$ is the average of the positions of stroke samples along the u isoline,

$$\overline{p}(u) = \frac{1}{|C(u)|} \sum_{(s,i) \in C(u)} p_{s,i},$$

and $\overline{k}(u)$ is the average of the curvature values at these samples,

$$\overline{k}(u) = \frac{1}{|C(u)|} \sum_{(s,i) \in C(u)} \frac{(\tau_{s,i+1} - \tau_{s,i-1}) \cdot \tau_{s,i}^{\perp}}{\|p_{s,i} - p_{s,i-1}\| + \|p_{s,i+1} - p_{s,i}\|}$$

where $p_{s,i}$ are sample positions, and $\tau_{s,i}$ are the unit tangents at these samples.

We discretize $\widetilde{X}(u)$ by using a finite set of parameter values $u_j \in [0, L]$ and represent it as a polyline with edges $\{\widetilde{X}(u_j), \widetilde{X}(u_{j+1})\}$:

$$\widetilde{X}(u_j) = \underset{\widetilde{X}}{\operatorname{argmin}} \sum_j \left\| \widetilde{X}(u_j) - \overline{x}(u_j) \right\|^2 + \lambda_1 \left\| \widetilde{T}(u_j) - \overline{\tau}(u_j) \right\|^2 + \lambda_2 \left\| \widetilde{K}(u_j) - \overline{k}(u_j) \right\|^2. \quad (16)$$

Unit length tangents and curvatures are non-linear functions of point positions, and therefore directly solving for $\widetilde{X}$ is computationally expensive. We solve for the position $\widetilde{X}(u_j)$ by approximating the minimization above via two linear solves. Instead of directly computing the positions $\widetilde{X}(u_j)$, we first compute the *joint* tangents $\widetilde{\tau}_j$ that balance curvature preservation and tangent alignment. We then use these tangents to compute the positions.

Assuming arc length parameterization of the input strokes, we apply the Frenet-Serret formulas to get $\tau'(u, p) = k(u, p)n(u, px)$, where $n(u, p)$ is the unit normal and $k(u, p)$ is the curvature at a point, to solve for tangents at u_j:

$$\min_{\widetilde{\tau}} \sum_j \left\| \widetilde{\tau}_j - \overline{\tau}(C_j) \right\|^2 + \lambda_C \left\| \frac{\widetilde{\tau}_j - \widetilde{\tau}_{j-1}}{\left\| \overline{\tau}(C_j) \cdot (\overline{p}_j - \overline{p}_{j-1}) \right\|} - k_j \overline{\tau}^{\perp}(C_j) \right\|^2 \quad (17)$$

We then normalize the computed joint tangents to have unit length, and compute the point positions $\widetilde{X}(u_j)$ as the minimizers of

$$\min_{\widetilde{X}(u_j)} \sum_j (\widetilde{X}(u)_j - \overline{p}_j)^2 + \lambda_T \left\| \frac{\widetilde{X}(u)_j - \widetilde{X}(u)_{j-1}}{\left\| \overline{\tau}(C_j) \cdot (\overline{p}_j - \overline{p}_{j-1}) \right\|} - \widetilde{\tau}_j \right\|^2. \quad (18)$$

We set $\lambda_C = 10^3$ and $\lambda_T = 10^4$, prioritizing curvature preservation over tangent alignment and deemphasizing position preservation.

7 RESULTS AND VALIDATION

Throughout the paper we demonstrate our fitting and underlying parameterization methods on 9 vector format drawings containing 214 clusters, and 31 additional single-cluster inputs. An additional 71 drawings totaling 1,618 clusters are included in the supplementary material. In multi-cluster drawings, clusters were labeled semi-manually using the clustering of [Liu et al. 2018] as a starting point, but without employing their preprocessing step which cuts strokes into smaller clusters for easier fitting. These examples include self-intersecting clusters (e.g. Figs. 1, 17), narrow U-shapes (e.g. Fig. 14, col 3 and Fig. 15, right) spirals (e.g. Figs 14, right; 6), clusters with sudden curvature changes (e.g. eyebrows in Figs. 1), and ones with large gaps and tangent differences between participating strokes (Fig 18). In all cases we successfully compute arc length parameterizations that satisfy our core criteria (Sec 3.1), leading to fitting outputs well aligned with viewer expectations.

Comparison to Prior Art. As Sec. 2 and Fig. 4 demonstrate, methods that target fitting to 2D point clouds are poorly suited for the task of fitting intended curves to stroke clusters. We thus focus our comparisons on recent approaches specifically designed for this task [Liu et al. 2018; Orbay and Kara 2011]. Figs. 1, 5, 12-15 compare the fitted curves produced by StrokeStrip to those produced by these earlier approaches. The comparisons use the fitting output of Liu et al.'s code as-is. Orbay and Kara provided us their point-sample ordering code, but were unable to locate the original code they used for fitting curves to these ordered samples. To visually compare the fits, we therefore fit curves to their output ordered samples using the tangent-aware fitting of Liu et al., which we expect to provide results on par or superior to the B-spline fitting described in the original Orbay and Kara paper.

As these figures demonstrate, prior methods perform well on simple, low-curvature inputs, but frequently fail on more complex data. Both methods rely on point ordering as a precursor to fitting, and as the ordering visualizations in these figures demonstrate, the key reason for their fitting failures is incorrect ordering. The orderings they produce frequently violate monotonicity and other key properties identified in Sec. 3.1; our fitting leverages quality parameterizations which avoid these pitfalls.

Qualitative Comparison. We validate the quality of our results by comparing them to these alternatives via a comparative perceptual study. Study participants were shown input drawings, together with our fitting result and an alternative fitting result, using the following layout (Appendix C). We include 38 inputs sourced from earlier papers [Barla et al. 2005; Liu et al. 2018, 2015; Orbay and Kara 2011; Pusch et al. 2007], and 26 drawings commissioned from 4 artists. As Orbay and Kara [2011] and Liu et al. [2018] each commissioned drawings from at least two artists, our inputs include drawings from at least 11 artists. The input was shown at the top and marked as 'A', and the two fitting results were placed at the bottom in random order and marked as 'B' and 'C' . Participants were then asked: "Which of the clean drawings below, B or C, more accurately depicts the rough drawing A on top? If both are equally accurate then select 'Both'. If neither select 'Neither' ". The answer options were "B", "C", "Both", and "Neither". We included 80 questions comparing our results against Orbay and Kara [2011] and 76 against Liu et al [2018]; we collected answers for each query from 6 different participants using the protocol described in Appendix C. The study had a total of 36 participants, with each answering at most 32 questions. Participants included 1 artist; 8 participants familiar with vector drawing; and 27 participants with no art or graphics experience. All study data is provided in the supplementary.

Fig. 16 summarizes the study results. Participants preferred our method over the one of Liu et al [2018] 77% of the time, and preferred the alternative only 3% of the time. They preferred our method over the closest alternative [Orbay and Kara 2011] 61% of the time, ranked the methods as on par 29% of the time, and preferred the alternative only 5% of the time. Our t-test on the study data indicated that both results are highly significant; this study convincingly demonstrates that the curve fitting outputs we produce are more consistent with viewer expectations than those produced by prior approaches.

We note that on 5 of 156 prior-art comparisons a plurality of respondents answered 'Neither'; these occurred on synthetic wide single clusters used to stress test fitting methods. We believe that users perceived the input as multiple distinct clusters, and hence wanted a solution that contains multiple strokes. We fitted these examples as a single cluster, and suspect users ranked their preference

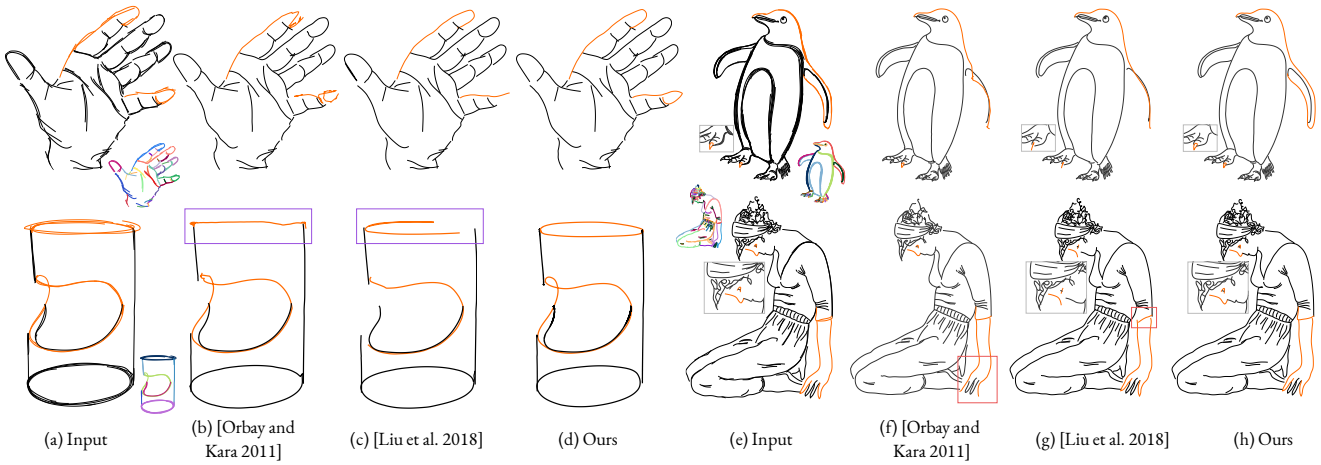


Fig. 12. Our method successfully fits aggregate curves (d,h) to inputs (a,e) that prior methods of Orbay and Kara [2011] (b,f) and Liu et al. [2018] (c,g) fail on. Input clusters highlighted in insets (a,e); visibly differently fitted clusters and curves highlighted in orange. Drawings, clockwise from top left, ©Silver Burla, Enrique Rosales, [Barla et al. 2005], Akshay Sharma (CC-BY-SA).

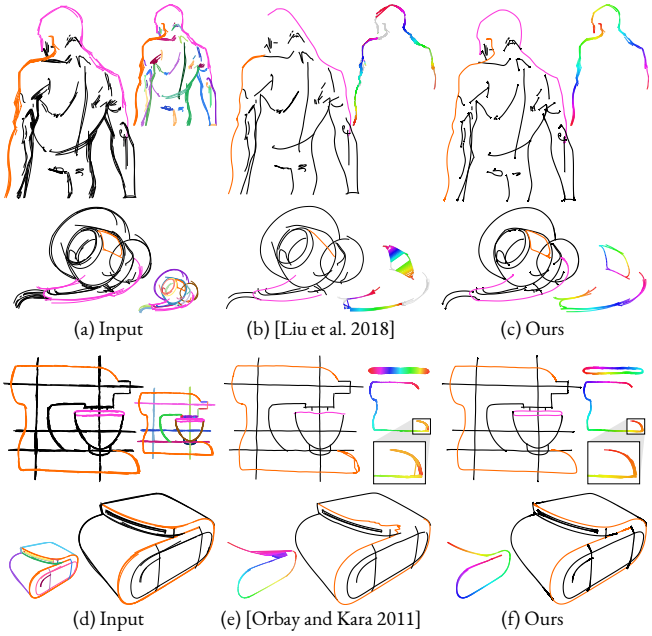


Fig. 13. Additional comparisons on complete drawings from the Yan et al. [2020] dataset against Liu et al. [2018] (top) and Orbay and Kara [2011] (bottom). Insets show ordered points for prior work and cluster parameterization for ours rendered using a rainbow color palette. Drawings, top to bottom, ©Anton Gulic, Patrick Murphy, [Favreau et al. 2016], Akshay Sharma (CC-BY-SA).

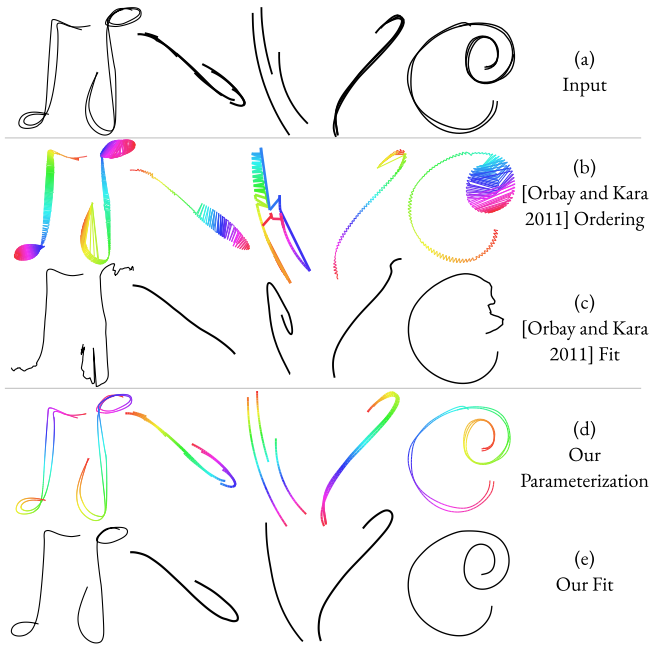


Fig. 14. While the method of Orbay and Kara [2011] fails to produce a correct order (b) on the input clusters (a) resulting in poorly fitted curves (c), our framework parameterizes (d) and fits (e) these clusters consistently with viewer expectations.

as 'Neither', as the final clean drawings from both methods did not contain the number of fitted strokes they expected.

On 3 of 156 comparisons against prior art, a plurality of respondents preferred the other method to our outputs (Questionnaire 4 #13; Questionnaire 6 #15; Questionnaire 5 #25). The fits on the first two look nearly identical – we speculate that different viewers balance fit smoothness vs accuracy differently. For Q5 #25, we speculate viewers perceive the bumps on the bee's body as pairs of separate curves, rather than single curves, and thus prefer the hooked fit of Orbay and Kara [2011]. While viewers may have varying attention to detail between inputs, and may differ in their preferred balance of smoothness and accuracy, having multiple participants evaluate each input and having many inputs helps compensate for this noise.

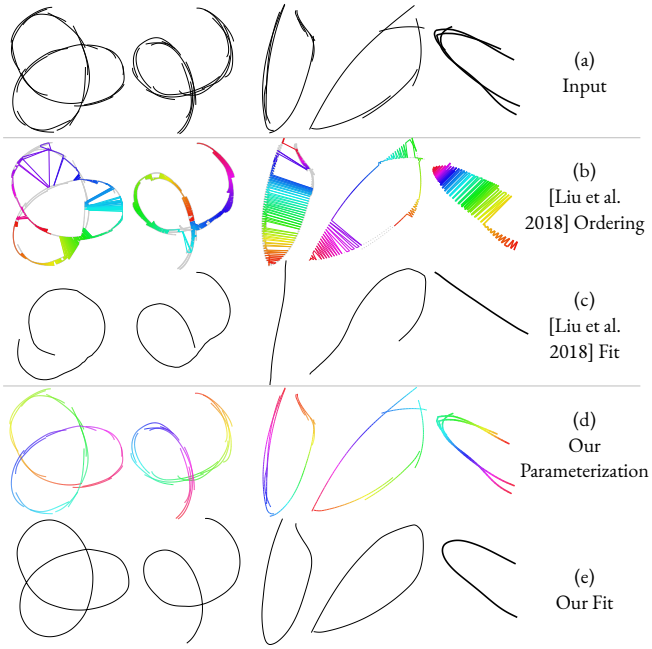


Fig. 15. While the method of Liu et al. [2018] fails to produce a correct order (b) on the input clusters (a) resulting in poor fitted curves (c), our framework parameterizes (d) and fits (e) these clusters consistently with viewer expectations.

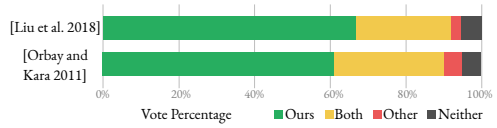


Fig. 16. Summary of comparative preferences in our perceptual study. Participants strongly preferred our results over algorithmic alternatives.

Comparison to Manual Fitting. Manually fitting curves to stroke clusters is highly time consuming [Liu et al. 2018; Yan et al. 2020]. We compare the curves fitted by our method to those produced by manual fitting using a comparative qualitative study that follows the same protocol as above. Specifically, we collected 27 overdrawn sketches with corresponding manually traced aggregate strokes (five from the dataset of [Liu et al. 2018] and the rest from [Yan et al. 2020]). Participants preferred our results 28% of the time, ranked both our and manual results as equally good 35% of the time, and preferred the manually traced outputs 37% of the time. Overall, these results indicate that observer view our results as having comparable quality to manual outputs. We note that comparisons against human consolidations are impacted by factors beyond fitting – artists may modify, suppress or add details during the clustering process, or may make different clustering choices than our annotator. Observations of inputs where one result was more strongly preferred suggests that viewer preferences may originate in such external factors, outside the scope of our work.

Stress Tests. Figs 17 and 18 show stress-test inputs we tested our method on. Fig 17 demonstrates our ability to fit intended curve to highly complex clusters which include multiple intersections and

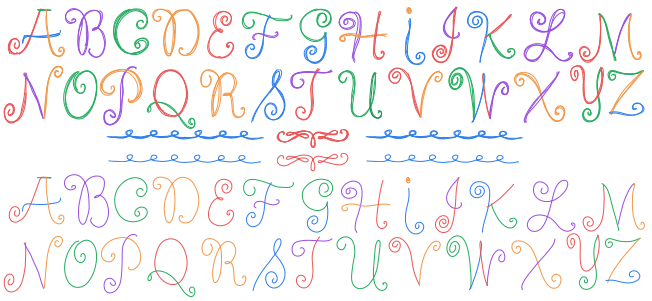


Fig. 17. (Top) Overdrawn letters of the alphabet and single clusters with multiple self-intersections, (bottom) correctly fitted by our method.

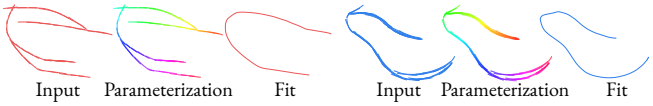


Fig. 18. Our outputs on clusters with complex branching structures.

turns. We expect our method to be most frequently used on input clusters that viewers perceive as conveying single intended curves; however, our framework makes no explicit assumptions about the cluster structure and, as demonstrated in Fig. 18, can be applied as-is to general clusters containing strokes which are neither proximate nor even roughly parallel. Another such example is included in the supplementary material and the aforementioned comparative study.

Implementation and Runtime. We implemented our parameterization method using MATLAB. StrokeStrip takes a median time of 1.6 seconds to fit an aggregate curve per cluster. The runtime increases as a function of the number of samples and overall cluster complexity. The latter impacts the rate of convergence of our iterative topology refinement step (Section 5.2). Additional implementation details are provided in Appendix F.

Fig. 19. Both vectorization [Adobe 2020] (b) and joint vectorization and simplification [Favreau et al. 2016] (c) methods frequently fail to generate clean curves when presented with single *raster* stroke clusters (a). Using the basic vectorizations in (b) as input, we successfully fit single intended curves to this data.

7.1.1 *Extensions and Applications.*

Raster Data. Our framework targets artist drawn vector stroke clusters. However, it can typically be successfully applied to vectorized raster stroke clusters (Fig 19) with the following minor changes that account for the differences between the two data sources. Specifically, vectorization methods break strokes at cluster intersections and vectorized raster strokes have more noisy tangents than human-drawn inputs. We account for these artifacts as follows. During

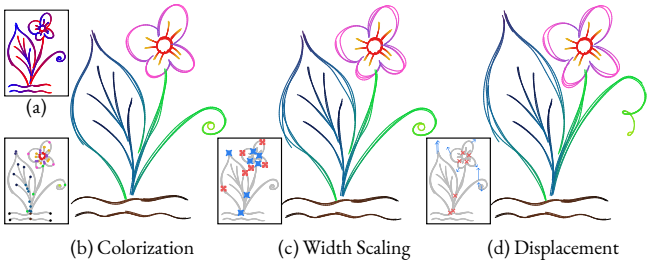
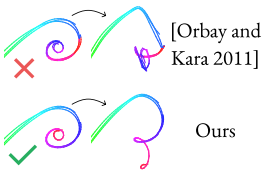


Fig. 20. Our parameterization (a) enables cluster level stroke manipulation tasks such as color interpolation (b), width scaling (c), and deformation (d). Drawing ©Enrique Rosales.

gradient direction computation (Sec. 4), given pairs of strokes with no common cross-sections we set the alignment score $\widetilde{w}_{i,j}$ to be a function of the angle between the stroke tangents at the closest stroke endpoints using Eq. 11. This change promotes assigning compatible gradient orientations for such pairs. Second, when solving for a parameterization, we force the parameter values across such pairs of endpoints to satisfy monotonicity constraints. Finally, during fitting, we compensate for unreliable tangents and curvatures by setting the expected curvature k_j to 0 in Eq. 17.

Parameterization-Guided Cluster Manipulation. The benefits of computing joint stroke cluster parameterizations extend beyond aggregate curve fitting. Our parameterization can be used for jointly manipulating different properties of the individual strokes in the cluster (Fig. 20, Fig. 1f). To manipulate the strokes jointly we first extend our 1D parameterization to a 2D (u, v) one by arc-length parameterizing each isoline across the interval $[-W/2, W/2]$ where W is the isoline width; we set $v = 0$ for isolines consisting of a single point. We use this parameterization to propagate user specified property values at a sparse set of stroke samples across all cluster strokes by minimizing either Dirichlet or Laplacian energy (finding harmonic or biharmonic interpolating functions respectively) along the u and v parameter intervals. We demonstrate this approach by shading strokes via biharmonic color interpolation starting from fixed user specified colors at selected points (Fig. 20b). Using our arc length parameterization produces smooth color change along and across the clusters. We use the same framework to modify cluster width - scaling the v isolines by a user given factor (Fig. 20c). Finally, we use biharmonic interpolation to intuitively deform stroke clusters (Fig. 20d, Fig. 1f); we compute per-point displacements as a function of user-specified anchor displacements, and then solve for new point positions consistent with these changes. The same framework can be used to compute anchor tangent or normal rotations, in line with Poisson deformation schemes. As the inset demonstrates, the quality of the parameterization is critical to the success of the editing task — using the parameterization induced by the sample ordering of Orbay and Kara [2011] (top) leads to visible artifacts during deformation.



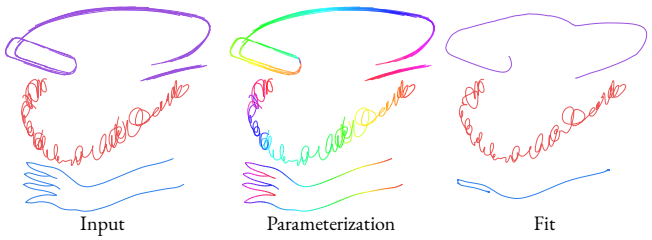
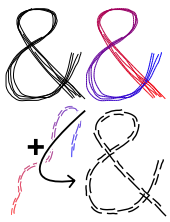
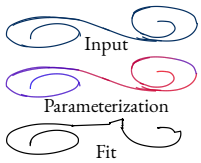


Fig. 21. Our method produces meaningful results on ambiguous inputs, but these may diverge from viewer expected ones.

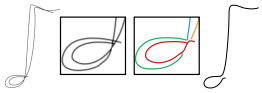


The inset demonstrates the use of our (u, v) parameterization to transfer the style of an exemplar cluster to a target cluster, generalizing the notion of a decorative brush [Lu et al. 2014]. Here we first map the path of the exemplar cluster to the target curve using arc-length parameterization. We then use this path as a deformation handle, to which all other strokes are 'rigged'.

7.1.2 Limitations. As demonstrated, our framework robustly handles a large and diverse set of inputs. However, given inherently ambiguous inputs (Fig. 21) it lacks the higher-order reasoning that may guide viewers when interpreting such data. In these examples observers may perceive the input on the top as a convoluted spiral, or the one on the bottom as a messy approximation of a low-curvature curve. Instead, our framework chooses interpretations more consistent with tangent alignment and maximal isoline span.



Our parameterization may converge on a local minimum instead of the global minimum. In the left most inset (a cluster from the Yan et al. [2020] dataset), stroke-orthogonal connections between two sides of a spiral are not filtered out in Sec. 5.2. As such cases are uncommon, we leave their detection and resolution to future work.



Our ability to fit raster inputs is dependent on the topology of the intermediate vector representation; we cannot correct cluster self-intersections incorrectly resolved by the vectorization (see inset). An interesting area for future work would be to either improve this intermediate data or develop a parameterization method that selectively relaxes monotonicity to overcome such input imperfections. Finally, our current MATLAB implementation does not allow for real-time parameterization; however we expect performance to be improved once the method is re-implemented using a language geared toward real-time performance.

8 CONCLUSION

We presented a robust method for fitting and parameterizing stroke clusters that dramatically outperforms prior art. We achieve this goal by casting stroke cluster parameterization as the computation of the restriction to the strokes of the natural arclength parameterization of the intended strips these clusters depict.

Our method has potential applications outside sketch processing, such as reconstructing road maps from GPS traces and processing

of astronomical data, where image processing techniques extract multiple approximately parallel and nearby paths which need to be jointly fitted by singe curves [Bo et al. 2016; Chai et al. 2013; Türetken et al. 2013].

ACKNOWLEDGMENTS

We would like to thank Enrique Rosales, Sari Pagurek van Mossel, and Silver Burla for creating new art assets, Jerry Yin and Enrique Rosales for their assistance, and the reviewers for their editing and suggestions. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2019-05097 ("Creating Virtual Shapes via Intuitive Input"), NSERC grant RGPIN-2018-03944 ("Broad-Based Computational Shape Design"), NSERC Canada Graduate Scholarship - Master's, and the Fonds de recherche du Québec - Nature et technologies (FRQNT) grant 2020-NC-270087.

REFERENCES

Adobe. 2020. Illustrator. https://www.adobe.com/ca/products/illustrator.html

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.

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 *Proc. SBIM*.

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

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

Blender. 2021. Grease Pencil. https://www.blender.org/features/grease-pencil/.

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.

Dengfeng Chai, Wolfgang Förstner, and Florent Lafarge. 2013. Recovering Line-Networks in Images by Junction-Point Processes. In *Proc. CVPR*. 1894–1901.

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

Fernando De Goes, David Cohen-Steiner, Pierre Alliez, and Mathieu Desbrun. 2011. An Optimal Transport Approach to Robust Reconstruction and Simplification of 2D Shapes. *Computer Graphics Forum* 30, 5 (2011), 1593–1602.

T. Dey, K. Mehlhorn, and E. Ramos. 1999. Curve reconstruction: connecting dots with good reason. In *SCG '99*.

Luca Donati, Simone Cesano, and Andrea Prati. 2017. An Accurate System for Fashion Hand-drawn Sketches Vectorization. *Proc. ICCV* (2017).

Luca Donati, Simone Cesano, and Andrea Prati. 2019. A complete hand-drawn sketch vectorization framework. *Multimedia Tools and Applications* (2019), 1–31.

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

Gerald Farin. 2014. *Curves and surfaces for computer-aided geometric design: a practical guide.*

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.

William T Freeman, Joshua B Tenenbaum, and Egon C Pasztor. 2003. Learning style translation for the lines of a drawing. *ACM Trans. Graph.* 22, 1 (2003), 33–46.

A. Goshtasby. 2000. Grouping and Parameterizing Irregularly Spaced Points for Curve Fitting. *ACM Trans. Graph.* 19, 3 (2000), 185–203.

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

Michael Grant and Stephen Boyd. 2008. Graph implementations for nonsmooth convex programs. In *Recent Advances in Learning and Control*. 95–110.

Michael Grant and Stephen Boyd. 2014. CVX: Matlab Software for Disciplined Convex Programming, version 2.1. http://cvxr.com/cvx.

Cindy Grimm and Pushkar Joshi. 2012. Just DrawIt: A 3D Sketching System. In *Proceedings of the International Symposium on Sketch-Based Interfaces and Modeling (SBIM '12)*. Eurographics Association, Goslar, DEU, 121–130.

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

LLC Gurobi Optimization. 2020. Gurobi Optimizer Reference Manual. http://www.gurobi.com

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

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

L. B. Kara and K. Shimada. 2007. Sketch-Based 3D-Shape Creation for Industrial Styling Design. *IEEE Computer Graphics and Applications* 27, 1 (2007), 60–71.

Michael Kazhdan and Hugues Hoppe. 2013. Screened Poisson Surface Reconstruction. *ACM Trans. Graph.* 32, 3, Article 29 (2013).

Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. 2004. Spectral Surface Reconstruction from Noisy Point Clouds. In *Proc. SGP*. 11–21.

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.

Chenxi Liu, Enrique Rosales, and Alla Sheffer. 2018. StrokeAggregator: consolidating raw sketches into artist-intended curve drawings. *ACM Trans. Graph.* 37, 4, Article 97 (2018).

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

Y. Liu, H. Yang, and W. Wang. 2005. Reconstructing B-spline Curves from Point Clouds–A Tangential Flow Approach Using Least Squares Minimization. In *Proc. Shape Modeling and Applications*. 4–12.

Jingwan Lu, Connelly Barnes, Connie Wan, Paul Asente, Radomir Mech, and Adam Finkelstein. 2014. DecoBrush: Drawing Structured Decorative Patterns by Example. In *ACM Trans. Graph (Proc. SIGGRAPH)*.

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

Patryk Najgebauer and Rafał Scherer. 2019. Inertia-based Fast Vectorization of Line Drawings. *(Proc. Pacific Graphics)* 38, 7 (2019), 203–213.

Liangliang Nan, Andrei Sharf, Ke Xie, Tien-Tsin Wong, Oliver Deussen, Daniel Cohen-Or, and Baoquan Chen. 2011. Conjoining Gestalt Rules for Abstraction of Architectural Drawings. *ACM 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 TVCG* 17, 5 (2011), 694–708.

Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. 2008. Diffusion Curves: A Vector Representation for Smooth-Shaded Images. In *ACM Trans. Graph.*, Vol. 27.

Amal Dev Parakkat, Subhasree Methirumangalath, and Ramanathan Muthuganapathy. 2018a. Peeling the longest: A simple generalized curve reconstruction algorithm. *Computers and Graphics* 74 (2018), 191 – 201.

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

Richard Pusch, Faramarz Samavati, Ahmad Nasri, and Brian Wyvill. 2007. Improving the Sketch-Based Interface: Forming Curves from Many Small Strokes. *Vis. Comput.* 23, 9 (Aug. 2007), 955–962. https://doi.org/10.1007/s00371-007-0160-5

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.

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, and Hiroshi Ishikawa. 2018. Mastering sketching: Adversarial augmentation for structured prediction. *ACM Trans. Graph.* 37, 1 (2018).

Edgar Simo-Serra, Satoshi Iizuka, Kazuma Sasaki, and Hiroshi Ishikawa. 2016. Learning to simplify: Fully convolutional networks for rough sketch cleanup. In *ACM Trans. Graph.*, Vol. 35. 1–11.

T. Stanko, M. Bessmeltsev, D. Bommes, and A. Bousseau. 2020. Integer-Grid Sketch Simplification and Vectorization. *Comput. Graph. Forum* (Aug. 2020).

Fig. 22. Top: stroke clusters, bottom: Overlay of aggregate strips participants traced for each cluster in our WTS adjacency study. Participants demonstrated consistency across all inputs, confirming our hypothesis.

Engin Türetken, Fethallah Benmansour, Bjoern Andres, Hanspeter Pfister, and Pascal Fua. 2013. Reconstructing Loopy Curvilinear Structures Using Integer Programming. In *Proc. CVPR*. 1822–1829.

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.

Jun Wang, Zeyun Yu, Weizhong Zhang, Mingqiang Wei, Changbai Tan, Ning Dai, and Xi Zhang. 2014. Robust reconstruction of 2D curves from scattered noisy point data. *Computer-Aided Design* 50 (2014), 27 – 40.

Wenping Wang, Helmut Pottmann, and Yang Liu. 2006. Fitting B-Spline Curves to Point Clouds by Curvature-Based Squared Distance Minimization. *ACM Trans. Graph.* 25, 2 (2006), 214–238.

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.

Chuan Yan, David Vanderhaeghe, and Yotam Gingold. 2020. A Benchmark for Rough Sketch Cleanup. *ACM Transactions on Graphics (TOG)* 39, 6 (2020).

Zhouwang Yang, Jiansong Deng, and Falai Chen. 2005. Fitting unorganized point clouds with active implicit B-spline curves. *The Visual Computer* 21 (09 2005), 831–839.

A STUDY OF PERCEIVED WTS ADJACENCY

We performed a small scale perceptual study to validate our intuition that human observers are consistent in their perception of cluster outlines, or sides, and thus consistently differentiate between pairs of points that are WTS adjacent and those that are not (ones separated by outlines). Participants were shown individual stroke clusters (Fig. 22, top) and asked to "Please trace the sides of the stroke that is jointly conveyed by the curves below". The study included 5 participants and 7 representative clusters of varying complexity. As the overlays of the traced outputs (Fig. 22,bottom) show, participants traced the same sets of sides, which satisfied both the tangent alignment and the isoline span properties. These findings confirm our key hypotheses: that viewers are consistent in perceiving stroke clusters as strips, and that the strips they perceive satisfy the key properties we identified (Sec. 3.1).

B STRIP GEOMETRY

We first note that for a strip with an arc length parameterized centerline $\gamma(t)$ and variable width $W(t)$, to avoid local self-intersections, the centerline's curvature $k(t)$ has an upper bound as a function of strip width (a straightforward computation shows that $|k(t)| < 1/W(t)$). Here we assume strips do not have local self-intersections.

For such a strip, we formally define its arc length parameterization $u(x)$ as coinciding with the arc length parameterization at the centerline, and constant along each normal cross-section $C(t) = \{\gamma(t) + an(t) | a \in [-\frac{W(t)}{2}, \frac{W(t)}{2}]\}$. Hence, for some point $x = \gamma(t) + an(t)$ at the distance of a from the centerline, the gradient of its arc length parameterization is tangent to $\tau(t) = \gamma'(t)$ and is simply

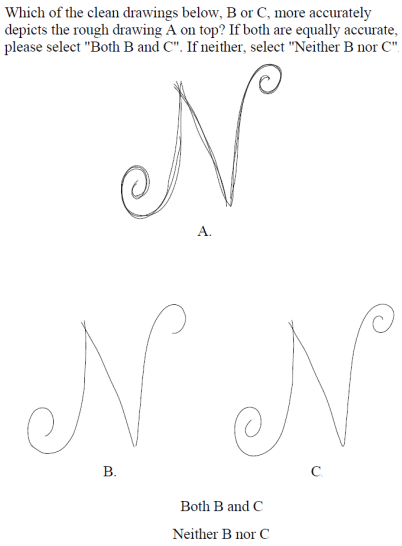


Fig. 23. Question layout in our comparative study.

$\nabla u(x) = (\gamma(t) + an(t))' = \tau(t)(1 - ak(t))$. (Here we used the Frenet-Serret formula $n'(t) = -k(t)\tau(t)$). We now prove:

Lemma B.1. *For a strip as defined above, the average of the arc length parameterization gradients along each normal cross-section $C(t)$ is the centerline unit tangent; i.e.:*

$$\frac{1}{W(t)} \int_{x \in C} \nabla u(x) dx = \tau(t).$$

Proof. Direct computation shows (denoting $W = W(t), C = C(t)$ for brevity):

$$\frac{1}{W} \int_{x \in C} \nabla u(x) dx = \frac{1}{W} \int_{-W/2}^{W/2} \tau(t)(1 - k(t)a) da$$
$$= (1/W) \left(\tau a - \frac{k(t)\tau(t)a^2}{2}\right)\Big|_{-W/2}^{W/2} = \tau(t).$$

□

C COMPARATIVE STUDY SETUP DETAILS

The questions (Sec. 7) comparing our results to prior art and manually consolidated images were randomly split into six questionnaires, randomly ordered; the two alternative outputs were placed in random order with the question layout shown in Fig. 23. Each questionnaire consisted of at most 32 questions. We collected six answers for each questionnaire. Participants received no training of any kind, and were provided PDF questionnaires remotely via email; this allowed them to zoom in to see fine details. All questions are included in the supplementary material.

D PERIODIC PARAMETERIZATION

The framework described so far produces open curve parameterizations, whose parameter extrema are typically located far apart. While individual strokes drawn by artists in standard drawing software are usually open, clusters of strokes can and do frequently depict closed intended curves (e.g. eye pupils in Fig. 1). We support optional periodic parameterization of such clusters by first

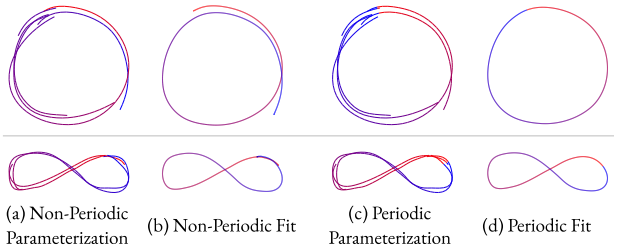


Fig. 24. (a) Initial non-periodic parameterizations of closed-curve clusters; (b) fitting using these parameterizations; (c) periodic parameterizations; (d) fitting using periodic parameterizations.

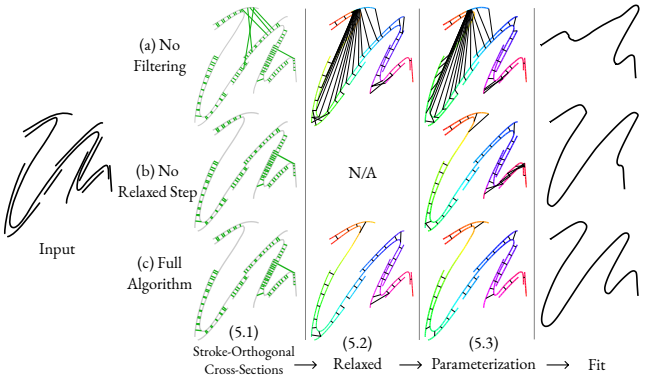


Fig. 25. Parameterization and fitting output generated by (a) skipping the initial cross-section filtering step (Sec. 5.1), (b) using the stroke-orthogonal cross-sections (Sec. 5.1) to initialize our final solve (Sec. 5.3), and (c) our method. These results demonstrate the necessity of the core steps of our method.

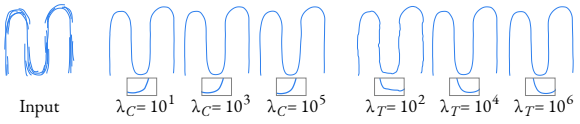


Fig. 26. Impact of changing fitting weights λ_C (left) and λ_T (right). In each example the central result is generated with the default values and the left and right are lower and higher respectively.

identifying them, and then by reparameterizing each such *closed curve* cluster by enforcing periodicity and selectively relaxing monotonicity requirements. The benefit of such a parameterization for aggregate curve fitting is illustrated in Figure 24.

Closed-Curve Clusters. In detecting closed-curve clusters, we leverage the observation that, for a directed graph with closed cycles, one can reach any cycle vertex from any other by following the edge directions. Extending this logic to clusters, if a cluster depicts a closed curve, then once the periodic parameterization is computed, we expect to be able to reach each stroke point from another by following the parameterization gradient. In practical terms, this means that given a graph that consists of directed edges connecting points with consecutive isovalues along the same stroke, and connecting points on different strokes that share the same isovalue, a cluster is a closed curve if, and only if, we can compute a directed path in this graph from each point sample to another. Notably, given a nonperiodic parameterization, such paths only exist from samples with a lower parameter value to those with a higher one.

We assess the potential for a parameterized cluster to be a closed-curve cluster by looking for paths in a graph that, in addition to the edges above, contains *orthogonal edges*: edges that connect immediately adjacent points p and p' that have similar parameterization gradients but different parameter values and where the line $\overline{p, p'}$ is approximately orthogonal to the gradients at both points. We note that this property is well satisfied by edges connecting ray seed points on our initial set of cross-sections and their immediate neighbors within these cross-sections (Sec. 5.1). We thus add these edges to the graph.

We classify a cluster as closed-curve if the resulting graph allows for a directed path from a point with the maximal parameter value to one with the minimal one. Note that once this is the case, this property will hold for all other pairs of points.

Periodic Reparameterization. If a cluster is deemed closed, we parameterize it once more (Sec. 5.3) with the following adjustments.

We estimate the parameterization period P as the biggest difference in parameter values between the end-points of the computed orthogonal edges. We then recompute a set of isolines such that for each value u they contain all points with this isovalue modulo P. We locate the isoline with the largest u value, and relax the monotonicity constraints between points on this isoline and the consecutive sample points on the strokes they are on. We then perform one iteration of minimizing E_{arc} (Eq. 8). All subsequent iterations proceed as described in Sec. 5.3.

E ABLATION STUDIES

We demonstrate the impact of our design decisions on the output parameterizations and fits throughout the paper (Figs. 6,7, 9, 10, 24, and 11). Fig. 25 further demonstrates the importance and impact of our initial stroke-orthogonal cross-section filtering (Fig. 25a) and our relaxed parameterization (Fig. 25a). Skipping either one of those results in poor quality parameterizations and fits.

Our parameterization method has just two empirical parameters: the distance ratio threshold S_{max} (Sec. 5.1) and the σ in Eq. 14 used to set adjacency likelihoods (Sec 5.2). Running our algorithm on all the letters in Fig. 17, we observed no change in fitting output when doubling or halving either parameter; the impact on runtime was under 10% in all cases. Our fitting method uses two additional weight parameters: one for curvature λ_C and one for tangents λ_T. Fig. 26 demonstrates the impact of decreasing or increasing each by a factor of 100 on our fitted outputs; predictably, reducing these weights makes the fitted curves more centered with respect to the clusters but less smooth, while increasing them increases smoothness.

Impact of Clustering. We run our fitting method on different clusterings for the same inputs, provided by different people (Fig. 27). Despite differing levels of cluster granularity, the resulting curves for each set of clusters are consistent with viewer expectations.

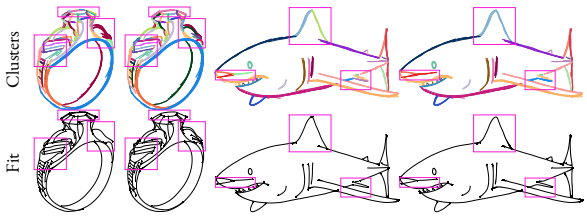
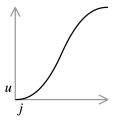


Fig. 27. Impact of alternative clusterings on fitting results. Inputs are from the Yan et al. [2020] dataset. Drawings ©[Liu et al. 2015] (left), Cristina Arciniega (right).

F IMPLEMENTATION DETAILS

Initial Sampling. We pre-process all strokes by resampling them using curvature-aware sampling: we add samples along strokes so that the distance between each pair of consecutive samples is at most one input stroke width, and the angle difference between the tangents at the sample is at most 4°. This process ensures adequate sampling density next to sharp stroke turns.

Parameter Space Sampling for Curve Fitting. Using evenly sampled isovalues u_j in our fitting formulation (Sec. 6) distributes approximation error evenly. In the interior of the cluster this error may be barely noticeable. However, drift in aggregate curve endpoint locations can change viewer perceived inter-cluster interaction, either removing perceived intersections or adding new ones, an outcome we wish to avoid. We minimize drift near aggregate curve endpoints by using higher isovalue sampling density next to them. With n samples, and parameter values in the range $[0, L]$, the j^{th} sample u_j is chosen as follows:



$$u_j = \begin{cases} \frac{L}{2}\left(2\frac{j-1}{n-1}\right)^{9/4}, & \frac{j-1}{n-1} < 0.5 \\ L - \frac{L}{2}\left(2 - 2\frac{j-1}{n-1}\right)^{9/4}, & \frac{j-1}{n-1} \geq 0.5 \end{cases} \tag{19}$$