StrokeStyles: Stroke-Based Segmentation and Stylization of Fonts

DANIEL BERIO, Goldsmiths, University of London, Computing Dept., United Kingdom
FREDERIC FOL LEYMARIE, Goldsmiths, University of London, Computing Dept., United Kingdom
PAUL ASENTE, Adobe Research, USA
JOSE ECHEVARRIA, Adobe Research, USA

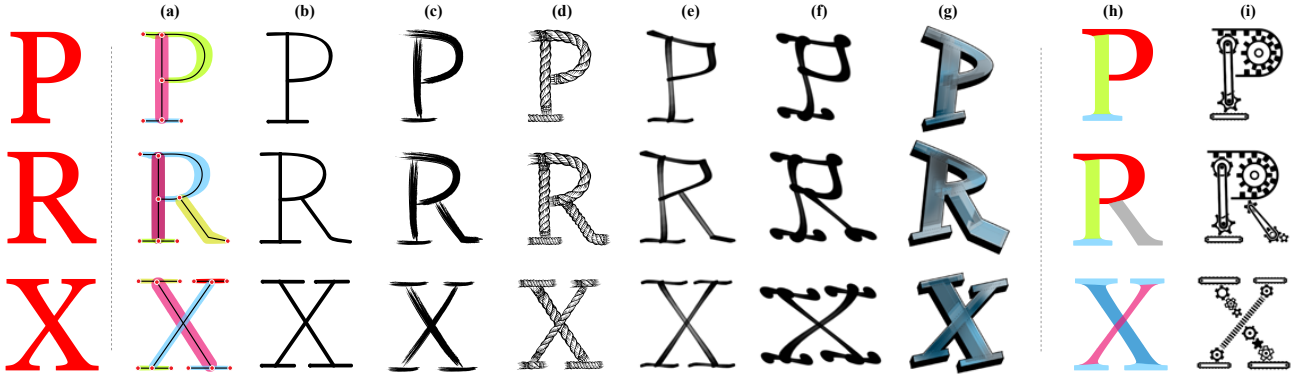


Fig. 1. We stylize glyphs by partitioning them (left, in red) into overlapping and intersecting strokes. **(a)** We represent strokes as a set of spines with variable width profiles and annotations describing structural relations among strokes. This is then used to reconstruct the glyph in a variety of path-based styles: **(b)** line-based schematizations, **(c,d)** graphic stylizations using skeletal strokes, **(e,f)** artistic stylizations that mimic handwriting and **(g)** graffiti art. **(h)** We also use strokes to segment the input into overlapping areas. **(i)** We use these areas to compute a similarity metric between strokes, allowing consistent shape-based stylizations across the glyphs of a given font.

We develop a method to automatically segment a font's glyphs into a set of overlapping and intersecting strokes with the aim of generating artistic stylizations. The segmentation method relies on a geometric analysis of the glyph's outline, its interior, and the surrounding areas, and is grounded in perceptually-informed principles and measures. Our method does not require training data or templates and applies to glyphs in a large variety of input languages, writing systems and styles. It uses the medial axis, curvilinear shape features that specify convex and concave outline parts, links that connect concavities, and seven junction types. We show that the resulting decomposition in strokes can be used to create variations, stylizations, and animations in different artistic or design-oriented styles while remaining recognizably similar to the input font.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; • **Applied computing** → **Media arts**.

Additional Key Words and Phrases: Font structure, Stroke-based representations, Glyph stylization, Junction types, Curvilinear Shape Features, Augmented Medial Axis

Authors' addresses: Daniel Berio, Goldsmiths, University of London, Computing Dept., London, United Kingdom, d.berio@gold.ac.uk; Frederic Fol Leymarie, Goldsmiths, University of London, Computing Dept., London, United Kingdom, ffl@gold.ac.uk; Paul Asente, Adobe Research, San Jose, USA, asente@adobe.com; Jose Echevarria, Adobe Research, San Jose, USA, echevarr@adobe.com.

0730-0301/2022/1-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Daniel Berio, Frederic Fol Leymarie, Paul Asente, and Jose Echevarria. 2022. StrokeStyles: Stroke-Based Segmentation and Stylization of Fonts. *ACM Trans. Graph.* 1, 1 (January 2022), 21 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Modern fonts are commonly represented as vector outlines. While this format is convenient for exchange, rendering, and printing, it makes it difficult to apply modifications or stylizations that are based on the structure of the glyphs [Campbell and Kautz 2014]. The visual conventions used in creating a font or glyph can be traced back to their origins in stroke-based handwriting and calligraphy [Noordzij 2005], in which a stroke typically embodies a trace of ink on paper left by the gesture of a calligrapher manipulating a brush or pen. The outline of a glyph often conceals a latent structure of generalized strokes that, when combined, closely reproduce the glyph's shape. Recovering this underlying structure makes it easier to stylize and modify glyphs consistently across an entire font.[1]

1.1 Motivation

Generating fonts in a variety of styles, while leaving sufficient parametric control to a user, is a well-known ill-posed problem [Hofstadter 1982]. Our goal is to capitalize on the wealth of publicly available fonts as a source for possible letter structures and styles. By segmenting the glyphs of a font into strokes and characterizing

[1] Many related terms are used to refer to an individual character shape and to a collection of them: character, glyph, letterform, letter, font, type, typeface. We use "glyph" for an individual character and "font" for a consistently-designed collection of glyphs.

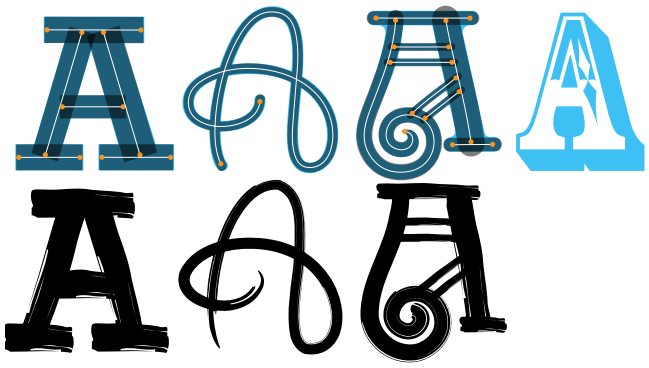


Fig. 2. The targets for our method are glyphs that have a recoverable stroke structure, such as the first three glyphs from the top left (Rockwell, Giddyup and Apollo ASM fonts), but not the glyph on the right (Rosewood). The inferred stroke reconstruction can be exact (Rockwell, Giddyup) or deviate slightly from the glyph's outline (Apollo). Our method works with glyphs having nonstandard structures, like Giddyup and Apollo, which would present challenges for template-based approaches. In the second row are stylizations of the first three glyphs, produced by our system with constant-width skeletal strokes [Hsu et al. 1993].

their topological relationships, we produce a scaffold for generating structurally-aware stylizations of the glyphs, and the wide variety of available digital fonts becomes the source for these scaffolds.

Our system relies on well-studied principles from visual perception [Wagemans et al. 2011]. It must deal with the same issues raised by the related problem of decomposing 2D object outlines into parts: multiple ambiguous hypotheses are acceptable, and their selection depends on subtle perceptual cues [De Winter and Wagemans 2006], on domain knowledge, and on functional or causal attributes [Spröte et al. 2016]. In particular, psychophysical results suggest that perceptual grouping [Brooks 2015] and formulating early part-segmentation hypotheses [Xu and Singh 2002] are low-level processes that occur pre-attentively, or at least very early in the vision process.

To approximate and model these perceptual processes, we adopt a recently introduced representation of *curvilinear shape features* based on local symmetry axes, and we identify pairwise relations between these features, called *links*, that guide the segmentation. We constrain the space of possible solutions by defining seven types of *junctions*, an intermediate representation of how symmetry axes intersect, that help characterize where and how strokes can overlap or end. Junctions are found iteratively and their identification fully characterize the recovered stroke structure of the glyph.

Our method produces plausible stroke-based segmentations of glyphs, using shape analysis alone (Fig. 2). While this can produce segmentations that are somewhat different from the traditional structure of the glyph, or from ground truth if it exists, it has the considerable advantage of being agnostic to the symbols used and works with glyphs that do not match any standard structure for a letter. The result is a system that can be applied to most glyphs and languages, and even to other 2D shapes that can be closely approximated by a series of strokes.

This paper contains many symbols; we have included a list of them in Appendix D.

2 RELATED WORK

Font stylization and synthesis. Some font stylization methods operate on glyph outlines [Campbell and Kautz 2014] or on raster glyph images [Azadi et al. 2018; Haines et al. 2016]. Other approaches operate on glyph structures like we do, but rely on a user-guided segmentation or skeleton assignment [Gingold et al. 2008]. Suveeranont and Igarashi [2010] use a skinning approach to assign a user-defined structure to font glyphs and then propagate changes made to the skin or skeleton of one glyph to newly-generated ones. With a similar objective, Phan et al. [2015] rely on a user-guided segmentation of the input glyphs into parametric strokes, defined according to the model of Jakubiak et al. [2006], and then use a probabilistic approach to propagate changes made to parts of one glyph to the rest of its font. Unlike these works, which address style transfer from one glyph to an entire font, we use stroke decomposition as input to structural stylization methods that can apply to an entire font. Our aim is closer to the work of Zhang et al. [2017], who stylize text by subdividing its letters into parts and reconstructing each part with a deformed vector image. The segmentation produced by our method could be used for similar applications.

Other related works use segmentation to create new glyphs or fonts. Xu et al. [2012] use a semi-automatic segmentation procedure to extract strokes from Chinese calligraphy instances and apply a brush model to synthesize new characters based on weighted interpolation. They then train a neural network to guide the synthesis process, based on the supervision of expert calligraphers. Lake et al. [2015] use a probabilistic programming approach to infer plausible motor programs from bitmap images of handwriting and then generate new exemplars by synthesizing novel motor plans. Their approach relies on an automatic thinning-based stroke segmentation method that unfortunately does not perform well with fonts. The output of our method can similarly be interpreted as a motor plan. Such a motor plan can then be modified, for example using the approach of Berio et al. [2017], to produce calligraphic-like glyph stylizations.

Parametric and stroke-based font models. Latin fonts are usually defined with outlines [Karow 1994], but a stroke-based representation of electronic fonts can be traced back to the MetaFont system [Knuth 1979]. It defines glyphs using raster shapes swept along splines. Jakubiak et al. [2006] similarly describe glyph parts using Bézier stroke paths paired with variable-width offsets. Hsu and Lee [1994] show examples of skeletal strokes used in instances of Chinese calligraphy; we also demonstrate how the same method can be used to stylize arbitrary fonts. Hu and Hersch [2001] present a parametric component-based representation of glyphs and emphasize that treating separately each side of a stroke, with respect to a central spine or axis, produces more aesthetically pleasing results. We follow a similar asymmetric approach using the stroking method by Berio et al. [2019], which mimics the appearance of graffiti art. Cox et al. [1982] present a graph-based description of fonts consisting of stroke-like parts and their topological relations. Our method captures similar topological relations among strokes.

Decomposition into strokes. Decomposing East Asian characters, which are often based on a hierarchical structure of radicals and

strokes, has been well studied [Chen et al. 2017; Sun et al. 2014; Wang et al. 2002] and extensive datasets are available for data-driven methods. However, such methods usually fail with Western fonts and glyphs, which have a wider range of stylistic variations and decorations, and which often blend components into each other in ways that make segmentation difficult. For example, Hofstadter [1982] reflects on the countless forms and structures that a single letter can assume. This ill-posed problem [Lamiroy et al. 2015] has been partially addressed with user-defined templates [Balashova et al. 2019; Herz et al. 1997; Phan et al. 2015; Suveeranont and Igarashi 2010; Zhang et al. 2017] or a detailed analysis of glyph outlines [Shamir and Rappoport 1996].

A related problem is decomposing objects other than glyphs into parts. Existing methods address this for applications in graphics [Jiang et al. 2013; Luo et al. 2015; Mi and DeCarlo 2007] and recognition [Macrini et al. 2008]. Recently, with an approach to outline segmentation sharing some similarities with ours, Papanelopoulos et al. [2019] use the exterior medial axis branches to identify some concavities in an outline, and then decompose this outline into parts. However, these branches can miss features [Belyaev and Yoshizawa 2001] that are important to stroke decomposition (see Fig. 4), a problem that we address in our work.

Very few methods consider the problem of potentially *overlapping parts*. With the aim of vectorization, Luo et al. [2015] and more recently Kim et al. [2018] propose a data-driven method that can vectorize overlapping parts of Chinese characters, but their methods are not easily applicable to our context, since we are interested in segmenting fonts for which no ground truth is available. Froyen et al. [2015] propose using a Bayesian Hierarchical Clustering method [Heller and Ghahramani 2005] for the segmentation of simple tubular objects with a mixture of splines paired with Gaussian thickness profiles. Favreau et al. [2016] propose another approach that uses a Monte Carlo exploration method to create vectorizations of line drawings that maximise a tradeoff between simplicity and reconstruction accuracy. However, neither method is applicable to the thicker, complex shapes found in fonts. The problem of disentangling potentially overlapping parts also relates to *multi-manifold learning* [Arias-Castro et al. 2017; Deutsch and Medioni 2017; Goldberg et al. 2009], which is the segmentation of data generated by multiple, potentially-intersecting manifolds. While these methods operate on data samples rather than outlines, we use some ideas similar to those developed by Deutsch and Medioni [2017].

3 OVERVIEW

Given a 2D glyph generated with a union of strokes, we target the inverse problem of recovering the strokes from the glyph outline. In this context, we consider a stroke to be an elongated 2D region as created by a drawing or painting gesture between two positions on a drawing surface [Noordzij 2005]. The recovered strokes when combined must reproduce the original glyph, providing good coverage of its interior 2D area; this implies that our method can also apply to other 2D shapes that can be well-approximated by the union of strokes (Fig. 20). Fig. 3 shows an overview of our system.

Our method relies on a joint analysis of the glyph outline and its interior and exterior medial axes, a hybrid approach that brings together geometry and topology. We use the interior medial axis to describe the glyph topology, while both the interior and exterior medial axes allow us to compute a series of *curvilinear shape features (CSFs)*, descriptors of concave and convex geometric features with associated support segments along the outline (Section 4).

We connect pairs of concavities through line segments that we call *links*, where each such link represent a potential location where a stroke can start crossing a region covered by multiple strokes (Section 5). Links are then paired to connect a stroke across such a region. The relevance of a given pair of links is measured via a perceptually-inspired metric of *good continuation* along disjoint outline segments.

We use CSFs and links to transform the interior medial axis into a set of strokes, a transformation that is driven by the identification of features that we call *junctions* (Section 6). Junctions capture semantic stroke attributes by relating branching regions of the interior medial axis to features along the outline like corners and tips. They also characterize regions where strokes may cross and overlap each other. We identify junctions iteratively with a procedure (Section 7) that is driven by the good continuation metric and a set of measures aimed at reproducing the glyph outline while producing strokes that are smooth and perceptually consistent with configuration of concavities along the outline and their relations to the branching features of the medial axis. Junctions produce two stroke representations (Section 8), one consisting of spines paired with variable width profiles (Fig. 1a), and one consisting of potentially overlapping areas of the glyph that correspond to different strokes (Fig. 1h).

We use strokes and junctions to generate varied stylizations of a glyph (Section 9), ranging from painterly and decorative effects with skeletal strokes, to effects that mimic the appearance of calligraphy or graffiti art, to animations. These stylizations can be generated in real time, giving the user a powerful way to explore and interactively adjust different visual results.

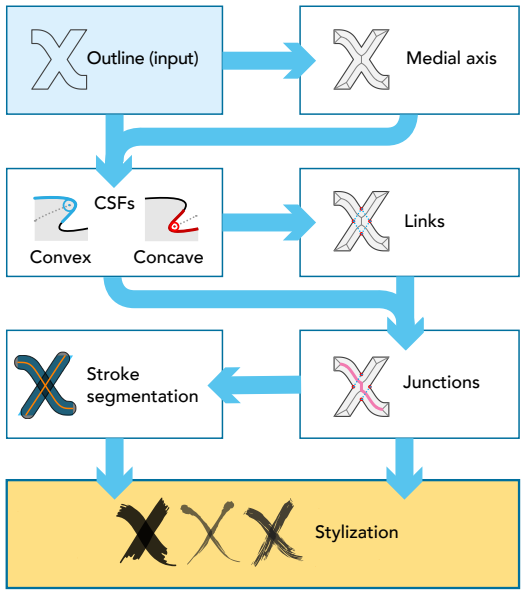


Fig. 3. High level overview of the segmentation and stylization of a glyph.

4 SHAPE ANALYSIS: OUTLINE & AXIAL FEATURES

Perceptual studies show that 2D shape understanding is driven by a combination of low-level cues from the boundary and interior local symmetries, and high-level global properties like the relationships among parts [De Winter and Wagemans 2006]. Motivated by these studies, we begin by constructing interior and exterior medial axes [Blum 1973], denoted $\mathbb{M}^I$ and $\mathbb{M}^E$, that provide geometrical and topological information about the shape of a glyph (Section 4.1). The medial axes let us identify a set of Curvilinear Shape Features (CSFs) that characterize concave and convex segments along the outline (Section 4.2) and facilitate the analysis of outline features such as tangents at concavities (Section 4.2.2). CSFs also map concave outline segments to axial features called ligatures (Section 4.3), portions of $\mathbb{M}^I$ that might need adjustment to recover smooth strokes. Furthermore, CSFs faciliate identifying outline features in relation to the branching structure of $\mathbb{M}^I$ (Section 4.4), which will help identify partition line segments called *links* and topological features called *junctions*, both needed to correctly find where and how strokes overlap.

4.1 Medial Axes

Many algorithms for computing the medial axis exist; we use the discrete Voronoi-based approximation method of Ogniewicz and Ilg [1992] because it is well-established and robust. We first densely sample each glyph outline to generate ordered sequences of points $\boldsymbol{x}_i$ and use the polylines that connect the points as an approximation of the outline. We use the authors' *chord residual* regularization to discard small spurious medial axis branches that come from discretization. Because the input to the Voronoi-based method is point samples, the medial axes always consist of planar graphs made of polylines connected by vertices $\boldsymbol{y}_j$. The interior medial axis $\mathbb{M}^I$ acts as a descriptor of the topological structure of the 2D shape and is used to identify salient convexities, while the exterior medial axis $\mathbb{M}^E$ is used to identify salient concavities (Section 4.2).

4.1.1 Terminals, forks, branches, and contact regions. *Terminals* in the medial axis are vertices of degree one. *Forks* are vertices of degree three or more. Terminals correspond to curvature extrema or sharp corners of the outline, while forks correspond to potential branching structures of the original shape and to polygonal stroke ends. Forks with degree more than three occur only in highly-symmetric configurations, like the center of a square; we remove them by making small perturbations to the outline points. A *branch* is a series of end-to-end connected edges that begins and ends at a terminal or a fork, with interior vertices of degree two.

Each vertex of a medial axis has an associated contact disk with a radius that is the minimum distance from the vertex to the outline. Each disk associated with a fork is connected to the outline at three distinct points. A terminal vertex has an associated *terminal disk* and *terminal branch*. Each terminal disk has a *contact region*, which is the arc of the disk that approximates the polygonal shape outline to within a small tolerance. We use its midpoint as the representative curvature extremum. Such an arc reduces to a point at a sharp corner where the terminal branch touches the outline. All vertices of degree 2 have disks touching the outline at two distinct points. We often visualize *ribs* that connect such vertices to the outline.

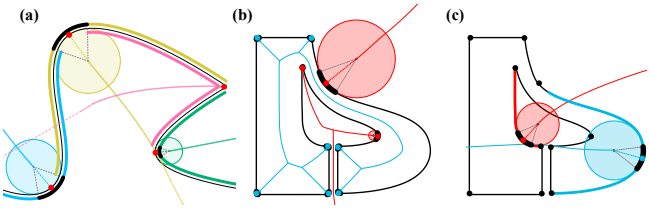
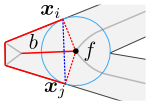


Fig. 4. **(a)** Four successive Curvilinear Shape Features (CSFs) with contact regions in black and with support segments, terminal disks and local symmetry axes in color . The pink CSF is a corner and thus has a contact region reduced to a point. Note that the local symmetry axes can intersect and overlap, unlike medial axes. **(b)** Axes $\mathbb{M}^I$ (blue) and $\mathbb{M}^E$(red), and the resulting convexities and concavities found at branch terminals. **(c)** Local medial axes are computed over the support segments highlighted in red (concave) and blue (convex), giving two new CSFs. These features are missed in **(b)** because they do not occur at terminal branches of the medial axes.

4.1.2 Branch salience, $\beta(b, f)$. To distinguish between $\mathbb{M}^I$ branches that characterize the body of a stroke from those that identify morphological features like the cap of a stroke or a corner, we define the salience of a branch b protruding from a fork f.



We consider the length s of the outline segment that connects two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ that have ribs connecting to the fork, and that contains at least two other points with ribs connecting to the branch's other endpoint. The branch salience is a measure of "stick-out" [Hoffman and Singh 1997]:

$$\beta(b, f) = \frac{s}{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|}, \tag{1}$$

which quantifies the length of the outline segment relative to its width at its point of attachment to the fork. A branch b is said to be salient with respect to a fork f if $\beta(b, f) \geq \tau_\beta$. Based on experiments, we found $\tau_\beta = 2.3$ works well. If the outline points are on different paths, e.g. when b is part of a loop surrounding a hole, we set $\beta(b, f)$ to an arbitrarily large value.

4.2 Curvilinear Shape Features (CSFs)

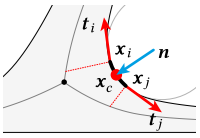
Our segmentation procedure requires identifying convex and concave outline regions, including sharp angles at corners. We call these *curvilinear shape features*, after Berio et al. [2020] (Fig. 4).

Definition 4.1 (Curvilinear Shape Feature (CSF)). A CSF has five elements: (i) a local symmetry axis, (ii) a terminal disk, (iii) a contact region (arc or point), (iv) the associated extremum of curvature, defined as the midpoint of the contact region, (v) a pair of outline segments on each side of the contact region called *support segments*, representing the CSF's region of influence.

Each support segment of a CSF is the portion of the shape outline extending from one end of the CSF's contact region to the beginning of the contact region of the adjacent CSF (Fig. 4a, colored outline segments). Adjacent CSFs always share one support segment and the CSFs for a given outline fully cover that outline. The local symmetry axis of a CSF is the medial axis of the part of the outline spanned by the CSF's contact region and its two support segments (Fig. 4a,

thin colored lines). Because this outline portion is left open, the symmetry axis starts at the CSF terminal disk center and extends, in theory, to infinity; in practice we truncate the axis at an enclosing bounding box. For visualization purposes we extend the axis from the terminal disk center to the associated curvature extremum $\boldsymbol{x}_c$ on the outline, represented by the outline sample nearest to the midpoint of the contact region. In the following, we often simply use "concavity" or "convexity" to refer to the CSF associated with the feature and we use the symbol c.

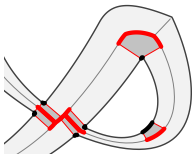
4.2.1 CSF computation. We can compute an initial set of convex and concave CSFs using the terminal disks of $\mathbb{M}^I$ and $\mathbb{M}^E$. However, this initial set is incomplete, since the medial axis can miss useful convexities and concavities, depending on the local configuration of the outline [Belyaev and Yoshizawa 2001] (Fig. 4b). We could identify these missing features with the full Symmetry Set [Giblin and Kimia 2003]; but this is difficult to compute and to manage. Instead we propose a simpler analysis in which we search for additional intermediate local medial axes by visiting each previously identified support segment. For each such open outline segment, we consider the potential additional CSFs produced by new candidate terminal branches. We select as a new CSF the one with smallest terminal disk radius, and only select a concave CSF if its radius is below an experimentally-determined threshold ($r_h = 0.15$) scaled by the glyph height. The search is iteratively repeated for the pair of newly-introduced shorter outline segments. This procedure ends when no new features are identified, which in practice takes one or two steps for most glyphs. More details are in Appendix A.



4.2.2 Concavity features. CSFs facilitate the computation of outline features useful for segmentation, such as tangents and normals near concavities. We assign each concave CSF a pair of unit tangent vectors $\boldsymbol{t}_i$ and $\boldsymbol{t}_j$ (inset, red arrows) at the endpoints $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ of its contact region and a unit *inward normal* (blue arrow), $\boldsymbol{n}$, with orientation $-(\boldsymbol{t}_i + \boldsymbol{t}_j)$ and terminating at the CSF extremum, $\boldsymbol{x}_c$. More details are in Appendix A.

4.3 CSF-based Ligatures

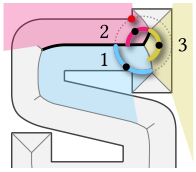
A *ligature* is a medial axis segment with ribs ending at a concavity. This notion was introduced by Blum [1973] and more recently used to identify shape parts [August et al. 1999; Macrini et al. 2011].[2]



We use ligatures more specifically to refer to medial axis vertices and segments with ribs that terminate in the contact region of a concave CSF. The inset figure shows several ligatures in red with their associated ribs. A ligature is always contiguous but can contain vertices from more than one branch. The union of the ligatures for a glyph make a set of *ligature regions*, connected subgraphs of $\mathbb{M}^I$ consisting of one or more overlapping ligatures. Each ligature region is a mapping from a connected portion of $\mathbb{M}^I$ to one or more concavities, and can be considered to be "glue" that connects perceptually distinct outline parts [Macrini et al. 2011].

[2]Blum first introduced this notion as a many-to-one mapping between medial axis points and a concave corner (semi-ligature) or a pair of concave corners (full ligature); this was later generalised to many-to-many mappings in concave outline regions for greater usefulness and robustness. We use the term ligature to refer to either semi- or full-ligatures. Our definition of ligatures in relation to CSFs is more general than those based on concave corners only.

4.4 Mapping Concavities to Forks via Sectors



The segmentation procedure requires assigning concave CSFs to each fork $f \in \mathbb{M}^I$ and identifying spatial relations such as adjacency and opposition between concavitities and branches incident to f.

We formalize these relations by computing three *viewpoints* for each $\mathbb{M}^I$ fork f, shown as black dots in the inset figure, one for each pair of incident branches. Each viewpoint is the midpoint of a circular arc connecting its two branches and centered at f. If the fork's disk (dashed circle) intersects both branches (sector 1), the arc's radius is equal to the fork's disk radius; otherwise the radius is the distance between the fork and the endpoint of the shorter branch (sectors 2 and 3).

A *sector* is a region of the plane that is visible from a given viewpoint without being occluded by any $\mathbb{M}^I$ branch. We use visibility polygons [Fabri and Pion 2009; Preparata and Shamos 1985] to compute these. Each sector is delimited by two of the branches incident to f. For example, sector 2 in the inset is delimited by the black branches. When a sector contains the extremum of a concavity, we say that the concavity and the two delimiting branches are *adjacent* (e.g. sector 2's concavity and the black branches in the inset) while the concavity and third incident branch are *opposite*.

4.4.1 Concavity assignment to forks. We use ligatures to assign zero or one concavity per sector to the sector's fork f, resulting in f being assigned up to 3 concavities. A given concavity may be assigned to more than one fork. To perform the assignments, we examine each sector in turn and all concavities adjacent to the sector's delimiting branches. Our goal is to determine whether one of these concavities can be interpreted as the result of the intersection of two strokes near f. If a concavity produces a ligature that overlaps with both of the sector's delimiting branches, we assign it to that sector (Fig. 5a). If there is no such concavity, we search for a concavity that can be assigned to the sector by computing a series of *radius-standardized distances* for all of the concavity's ligature points.

Definition 4.2. The *radius-standardized distance* between ligature point $\boldsymbol{y}_i$ and a disk in $\mathbb{M}^I$ is s^2/r^2, where r is the radius of the disk and s is the length of the shortest geodesic path through $\mathbb{M}^I$ connecting $\boldsymbol{y}_i$ to the disk center.

For each ligature point, we compute two radius-standardized distances: d_f, which is computed with respect to the fork's disk, and d_e, which is computed with respect to another disk, centered at a point $\boldsymbol{y}_e$ along the shortest path in $\mathbb{M}^I$ that connects the $\mathbb{M}^I$ branch containing the ligature point to the fork. To identify $\boldsymbol{y}_e$, we traverse the path starting from the fork towards the ligature, and conclude the search if either (i) we encounter a point that has a rib terminating at the extremum of a convex CSF that is not contained in the sector (Fig. 5b) or if (ii) we encounter the path endpoint or a fork whose disk does not overlap with the disk of the originating fork or

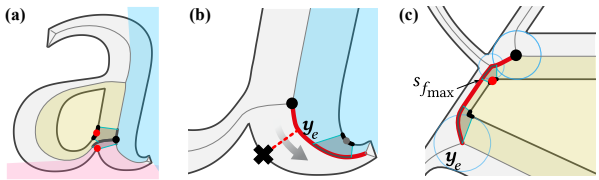


Fig. 5. Concavity assignment **(a)** Three sectors for a fork (black dot), with two concavities shown as red dots assigned to two of its sectors because they produce ligatures that overlap both of their adjacent branches. **(b)** The third sector (in blue) contains a concavity (grey dot) that is not assigned because it is closer, in a radius-standardized sense, to a point $\boldsymbol{y}_e$ with a rib terminating at the extremum of a convex CSF (black cross). **(c)** In another case, a sector (yellow) contains multiple concavities and it is assigned the red concavity. The point $\boldsymbol{y}_e$ is located at the path endpoint because the disks of the first two forks overlap. Placing $\boldsymbol{y}_e$ at the second fork along the red path would result in the sector not being assigned any concavity.

of any previously visited fork (Fig. 5c). Case (i) helps to distinguish concavities that are opposite convexities and that characterize a smooth bend rather than a potential intersection between strokes. Case (ii) helps to disambiguate nearby forks that potentially share the same concavity assignment with f. A concavity is assigned to the sector if $d_f \leq d_e$ for any of its ligature points and if the geodesic path length $s_{f_{\max}}$ going from the fork f to the concavity's last ligature point is shorter than similar geodesic paths for any other concavity (Fig. 5c).

5 PAIRING CONCAVITIES WITH LINKS

Past work has shown that pairs of concavities provide important cues for segmenting object silhouettes into parts [De Winter and Wagemans 2006]. Different approaches use such pairs to define "partition lines" [Luo et al. 2015; Singh and Hoffman 2001] or "cuts" [Papanelopoulos et al. 2019] that delimit perceptually-distinct object parts. Our stroke segmentation problem is related, but it differs in that it requires identifying incidence and crossing relations between potentially overlapping strokes. We start by joining pairs of concavities that we previously mapped to forks with line segments that we call *links*.

Definition 5.1. A *link*, denoted η, is a line segment that connects the extrema of two concave CSFs (c_i, c_j) that have been assigned to forks, and that is entirely within the glyph.

A single link identifies a potential location where one stroke can intersect or cross another stroke. Furthermore, by pairing links we seek to identify where a stroke enters and exits an ambiguous glyph region, i.e. an area that could be shared by multiple crossing and overlapping strokes.

Valid links. A glyph typically contains numerous links, some of which are not helpful, such as the diagonal links in the stem of the "B" in Fig. 6.a. We call the ones that do help *valid links* (Fig. 6.b). For a link to be valid, it must be possible to assign it a branch and an orginating fork (Section 5.1) that indicate an intersection between two strokes. Furthermore, we consider two valid links to be *incompatible* if they intersect or have the same branch/fork assignment; otherwise they are compatible. We call a set of valid

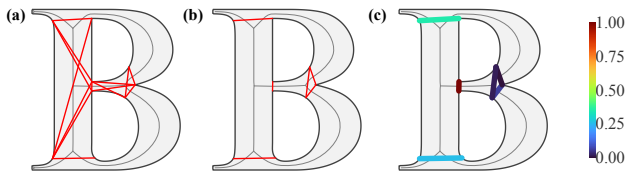


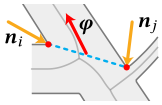
Fig. 6. Valid link selection. **(a)** Links that are internal to the shape. **(b)** Valid links that result in a valid branch assignment and are mutually compatible. **(c)** Valid links with corresponding salience.

links that are mutually compatible a *segmentation hypothesis*. We choose among segmentation hypotheses using a measure of *link salience* (Section 5.2), itself in part dependent of a measure of good continuation between concavities (Section 5.3).

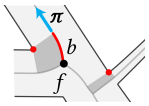
5.1 Assigning Branches and Forks to Links

A link encodes a direction that can indicate how a stroke protrudes from an intersection with another stroke; we call this direction the link's *flow*. We use the flow to search for a branch that emanates from an originating fork in a similar *protruding direction*. A link is valid only if such a branch can be identified.

5.1.1 *Flow and protruding directions.*



Definition 5.2. The *flow* $\boldsymbol{\varphi}_{ij}$ (inset: red arrow) for a link (dashed blue) connecting a pair of concavities $C = \{c_i, c_j\}$ is a unit vector with orientation $-(\boldsymbol{n}_i + \boldsymbol{n}_j)$, where $\boldsymbol{n}_i$ and $\boldsymbol{n}_j$ are the concavities' inward normals (yellow arrows).



Definition 5.3. The *protruding direction* $\boldsymbol{\pi}(b, f, C)$ (inset: blue arrow) of a branch b connected to a fork f with respect to a set of concavities C is given by the first unit tangent vector along the branch that is not part of a ligature (red branch segment) produced by a concavity in C. If the whole branch is part of a ligature, $\boldsymbol{\pi}$ is the tangent at f. A branch connecting two forks has two protruding directions, one for each fork.

Intuitively, the protruding direction uses ligatures to identify a point along a branch that is not shared by multiple strokes, and thus can be used to approximate the tangent along a single stroke spine.
A link will be considered for further evaluation only if we can identify a branch b and an originating fork f, for which the projection, p, of flow $\boldsymbol{\varphi}_{ij}$ onto $\boldsymbol{\pi}$ is strictly positive, where p is computed as:

$$p(b, f, C) = \boldsymbol{\varphi}_{ij} \cdot \boldsymbol{\pi} \quad . \tag{2}$$

5.1.2 *Fork and branch assignment.*

We use Eq. 2 to search for a branch b and fork f by considering the forks f_i and f_j that were assigned to the concavities $C = \{c_i, c_j\}$ (Section 4.4.1). We consider two mutually-exclusive cases to identify valid links.

Case 1: Normal link. There is at least one fork f_i that has both of the link's concavities assigned to it. For each such f_i we identify the branch b_i that delimits both sectors containing the concavities and compute $p(b_i, f_i, C)$. If there is more than one such f_i, we take the one with the largest positive p and assign (b, f_i) to the link.

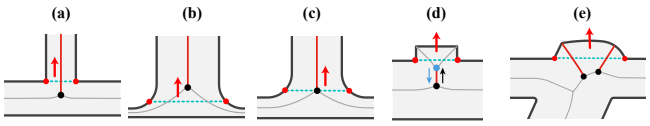


Fig. 7. Branch and fork assignment of a link (dashed blue line). A red arrow shows the flow direction, $\boldsymbol{\varphi}_{ij}$, (Def. 5.2) of the link, while black dots and red segments are forks and branches that could be assigned to that link. **(a, b, c, d)** show prototypical branch configurations that result in a normal link. In **(d)**, the concavities are assigned to both the black and the blue fork, but p (Eq. 2) is positive only for the black fork. **(e)** The link intersects two branches incident to different forks, creating a compound link.

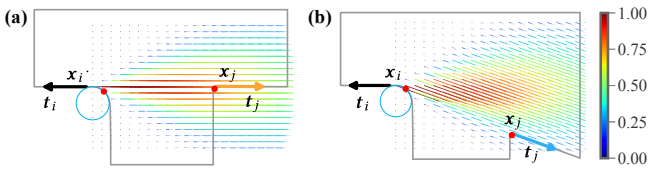


Fig. 8. Association fields for two concavities in a letter T with corresponding colored values ψ. The tangents pointing to the right are colored according to their association with the black tangents. **(a)** Case where the corners are well-aligned and the association field gives a sufficiently high good-continuation value $\psi \approx 0.75$. **(b)** Case where the corners are not well-aligned: the association field from one corner reaches the other but only with a low good-continuation value $\psi \approx 0.25$.

Fig. 7a-d show four prototypical configurations resulting from such an assignment.

Case 2: Compound link. There is no fork that has both concavities assigned to it and the link intersects two or more branches (Fig. 7e). We define F_B to be the set of forks at the endpoints of the intersecting branches. We then consider each $f_i \in F_B$ and find the branch b_i with maximum projection $p(b_i, f_i, C)$ among branches incident to each fork. If (i) all the projections $p(b_i, f_i, C)$ are positive, (ii) all the branches b_i are non-salient, (iii) none of the branches share a fork, (iv) each branch b_i has ribs terminating in only one concavity in C, and (v) each fork f_i has a disk that overlaps with the disk of another fork in F_B, this results in a special configuration we call a *compound link*. Then, the link is assigned any one of these branches together with its fork. This configuration is similar to a normal link, except that the protrusion is not sufficient for the branches to merge at a single fork (compare Fig. 7 d & e).

5.2 Link salience

The disambiguation of incompatible links can be achieved with a measure that prioritizes perceptually salient links. We use three concepts from perceptually-driven studies of part decomposition to favor links that are: (i) short (aka the "short-cut rule" [Singh and Hoffman 2001; Singh et al. 1999]), (ii) located between outline regions with good continuation (aka "limbs" [Siddiqi and Kimia 1995]) and (iii) connecting pairs of concavities with a relatively small radius of curvature (aka the "minima rule" [Hoffman and Richards 1984]). *Link salience* is then computed as:

$$\omega(\eta) = e^{-(r_1+r_2)/(2r_{\max})} + \psi(c_1, c_2) \tag{3}$$

combining an exponential function that decays with increasing concavity radii r_1, r_2, scaled by the maximum concavity radius $r_{\max}$, together with a measure of good-continuation between the linked concavities, denoted $\psi(c_1, c_2)$, that decays with the distance between concavities and thus penalizes longer links.

5.3 Good continuation (ψ) for links

Selecting valid links requires pairing concavities using a measure of good continuation along the outline. We use *association fields* [Wagemans 2018], which have been proposed to model the neural processes responsible for contour integration and perceptual grouping in early vision. A few computational implementations have been defined, the original one being based on *cocircularity* [Parent and Zucker 1989; Yen and Finkel 1998], i.e., how one local orientation, typically specified by an edge, can be connected to another nearby edge if it is reachable by circular paths within a region specified by the field. We adapt a more recent experimentally verified approach by Ernst et al. [2012] that is based on a stochastic model of contour integration [Williams and Thornber 2001]. Given two oriented edge elements, the model defines a field that decays as a Gaussian function of deviation from perfect cocircularity, collinearity, and distance between the two edges (Fig. 8).

We compute the *good-continuation* value $\psi(c_i, c_j)$ for a link's two concavities by first selecting the outline tangent at an endpoint of each concavity (§4.2.2) that is most orthogonal to the link's flow $\boldsymbol{\varphi}_{ij}$ (Def. 5.2). The association field is then computed using the concavity endpoints and tangents, and set to decay exponentially with distance relative to a spread parameter, σ_d, which we set to $2r_{\max}$, i.e. twice the maximum radius of any $\mathbb{M}^I$ disk that is not part of a ligature region. More details are in Appendix B.

We will also evaluate a good continuation measure ψ in two other cases: (i) when selecting the best crossing paths for strokes, such that a stroke can enter and exit an ambiguous glyph region, by comparing pairs of links (§7.2), and (ii) when associating stroke spines (§7.3.2). Before we reach those cases, using links and their assigned concavities, we need to categorize all $\mathbb{M}^I$ forks and their incident branches into higher level features we call *junctions*.

6 JUNCTIONS

A junction γ comprises a set of $\mathbb{M}^I$ forks F_γ together with their assigned links and concavities. It defines a configuration of $\mathbb{M}^I$ branches that correspond to a particular area in the glyph. For brevity, we will say that the junction *covers* these branches and forks. Once all junctions have been identified, they uniquely determine the inferred stroke decomposition of a glyph. Note that we use "junction" to refer to $\mathbb{M}^I$ branches joining, and not to glyph strokes joining. Junctions often coincide with areas where strokes join, but not always.

We define a taxonomy of seven junction types (half, Y, T, L, stroke end, protuberance and null), shown in Fig. 9, sufficient for our goal of stroke stylization.

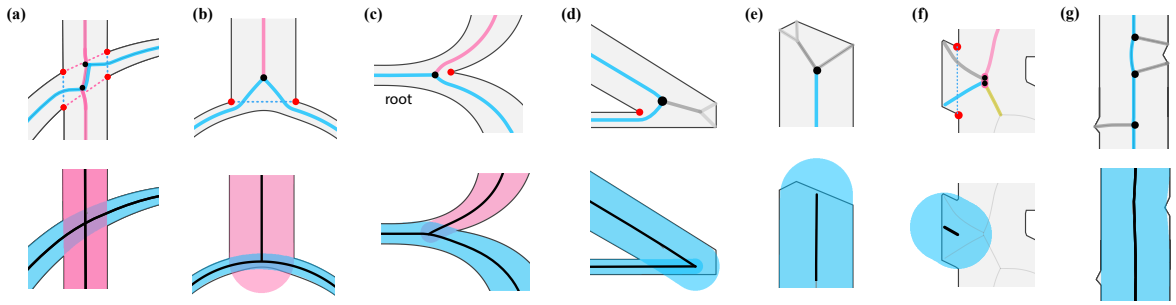


Fig. 9. Junctions. The first row shows each fork (black dot), branch (thick colored path), concavity (red dot) and link (dashed segment) configuration that characterizes each junction type. The second row shows the resulting strokes. **(a)** *Two* half-junctions, defined by two link pairs (dashed blue and pink) that share the same multitraced crossing path that connects the same forks (black dots). The two junctions produce two crossing strokes. **(b)** One T-junction, defined by one link (dashed blue). **(c)** One Y-junction, defined by a concavity opposite a root branch. **(d)** One L-junction, with the root in grey and its hierarchy (lighter grey), which will be discarded. **(e)** Stroke end. **(f)** Protuberance, characterized by a compound link (dashed blue). **(g)** Three null-junctions leading to a single stroke.

6.1 Junction types

Half-junctions. One stroke goes across one or more other strokes. A half-junction is characterized by a pair of links that are assigned to different forks and have a high good-continuation value between the concavities at the link ends (Fig. 9a). Unlike the other junction types, which identify one or two strokes with all branches incident to one fork, a half-junction identifies one stroke that connects two branches and enters and exits a region delimited by the links. This simplifies the analysis of complicated crossings with more than two strokes. A simple crossing like that in Fig. 9a consists of two half-junctions that cover the same forks. Any additional stroke crossing the same area would imply an additional half-junction. The *crossing path* of each half-junction is the shortest sequence of branches in $\mathbb{M}^I$ connecting the forks assigned to the links.

T-junctions. One stroke is incident to another in a near-perpendicular fashion. A T-junction is characterized by a link that identifies a branch protruding from a fork (Fig. 9b).

Y-junctions. Two strokes branch out of an overlapping region. A Y-junction is characterized by a representative concavity c that is assigned to a fork f and that is opposite to a salient *root* branch incident to f (Fig. 9c).

L-junctions. One stroke contains a corner or an elbow-like bend (Fig. 9d). L-junctions have a configuration similar to Y-junctions, with a representative concavity opposite a root branch, but this root is short and often not salient.

Stroke-ends. The $\mathbb{M}^I$ branching structure at the junction is a tree composed of non-salient branches and is associated with the end of a stroke (Fig. 9e).

Protuberances. The $\mathbb{M}^I$ branching structure is *nested* with one of the non-salient branches being assigned to a compound link (Fig. 9f).

Null-junctions. $\mathbb{M}^I$contains a fork arising from a small or noisy outline feature. (Fig. 9g).

6.2 From junctions to strokes

Junctions identify semantic stroke features while also helping determine how $\mathbb{M}^I$ branches are transformed into strokes. To drive this transformation, we create a *stroke graph* $\mathbb{S}$ in which each vertex is a $\mathbb{M}^I$ branch and each edge connects branches that are part of the same stroke. Each vertex in $\mathbb{S}$ is also associated with a *stroke segment*, a spine and a width profile that initially coincide with the the path of the branch in $\mathbb{M}^I$ and the union of the disks for that branch. The vertices in $\mathbb{S}$ are initially set to be disconnected. We identify junctions one by one, and with each identification we perform *structural operations* (Section 6.2.1) on the connectivity and structure of $\mathbb{S}$ and *adjustment operations* (Section 6.2.2) on the associated stroke segments. Which operations are performed is a function of the junction type (Section 6.2.3). We will describe the identification procedure in detail later, in Section 7, after discussing the different junction types in more depth and how these transform $\mathbb{M}^I$ into strokes.

Once all junctions have been identified, the stroke segments for each connected component of $\mathbb{S}$ map to one stroke in the glyph. Fig. 10 shows the steps in the segmentation of a single glyph. Each

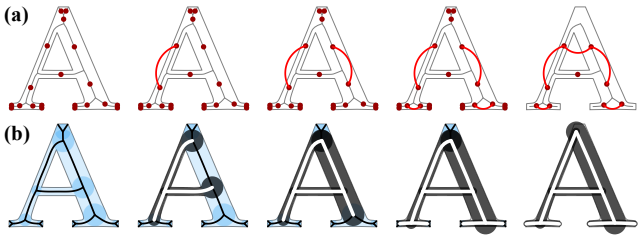


Fig. 10. Recovering strokes for a letter "A". **(a)** Structural operations on the stroke graph $\mathbb{S}$ with vertices ($\mathbb{M}^I$ branches) shown as red dots and edges as red arcs. The arcs connect branches that are combined into a single stroke. **(b)** Adjustment operations on the stroke segments: blue strokes with black spines are before adjustments, while black strokes with white spines are after adjustments. Each step straightens curved strokes near a junction; in the first step the effect is very subtle.

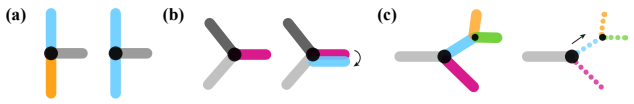


Fig. 11. Operations on the branches and connectivity of $\mathbb{S}$. **(a)** Connecting two branches incident to the same fork (black dot). **(b)** Multitracing (duplicating) a branch. **(c)** Discarding two branches and the tree-like hierarchy attached to one of them.

step shows the changes to the stroke graph and the adjustments to the stroke segments.

6.2.1 Structural operations on $\mathbb{S}$. Identifying a junction leads to applying one or more of the following structural operations to $\mathbb{S}$:

- **Connect** two or more $\mathbb{M}^I$ branches: add edges between pairs of vertices in $\mathbb{S}$. The corresponding $\mathbb{M}^I$ branches are incident, and will be part of the same stroke (Fig. 11a).
- **Multitrace** one or more $\mathbb{M}^I$ branches: duplicate the corresponding vertices in $\mathbb{S}$. This will let an $\mathbb{M}^I$ branch be shared by more than one stroke (Fig. 11b).
- **Discard** one or more branches incident to the same fork: remove them from $\mathbb{S}$. This removes branches that are not relevant to constructing a stroke. The operation is similar to conventional medial axis pruning methods [Shaked and Bruckstein 1998] and is applied recursively to any attached tree-like hierarchy in $\mathbb{M}^I$. It ends when it encounters a branch already connected in $\mathbb{S}$ (Fig. 11c).

6.2.2 Adjustment operations. Transforming the connected components of $\mathbb{S}$ into strokes involves assembling a simple path for each component, starting from a degree 1 vertex in $\mathbb{S}$ if it exists, or from an arbitrary vertex if the path forms a loop. Recall that the paths in $\mathbb{S}$ begin as the paths of the $\mathbb{M}^I$ branches. To recover smooth strokes from these paths, we must remove distortions that often occur near ligature regions. We use *transitions* that replace portions of one or more stroke segments with ones that smoothly interpolate an initial and a final pair of centers and radii, $(\boldsymbol{y}_i, r_i)$ and $(\boldsymbol{y}_j, r_j)$. We define two transition types:

- A *smooth transition* has a spine with points sampled from a clothoid connecting $\boldsymbol{y}_i$ to $\boldsymbol{y}_j$ and disk radii given by linearly interpolating between r_i and r_j. The parameters of the clothoid are determined by estimating a pair of tangents $(\boldsymbol{t}_i, \boldsymbol{t}_j)$ coinciding with $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ (Fig. 12, top row) and then using a secant-based optimization method by Levien [2009]. If not stated otherwise, r_i and r_j are also assumed to be the disk radii at $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$. Note that in many cases this clothoid reduces to a straight line segment.
- A *straight transition* has a straight spine that connects $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ and a constant width profile $r_i = r_j = r$ (Fig. 12, bottom row).

6.2.3 Junction operations. Each junction type determines a series of structural and adjustment operations that transform $\mathbb{S}$ into strokes:

Half-junctions. A half-junction **multitraces** the branches that fall along the crossing path and then **connects** the multitraced branches

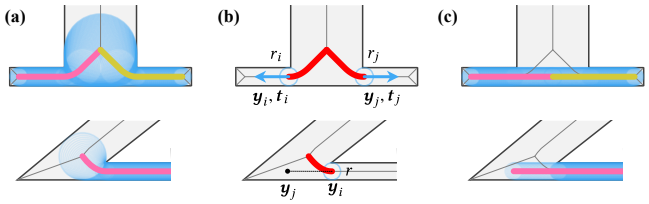
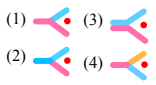


Fig. 12. Disk adjustment operations with a smooth transition (top row) and a straight transition (bottom row). **(a)** Branches (color coded) and disks (blue) before adjustment. **(b)** Ligature (in red) being replaced by the transition. The black dot in the second row is the endpoint of the straight transition. **(c)** Adjusted branch and disks.

with the branches protruding from the two links. The junction adjusts the stroke segments associated with these branches with a smooth transition. The transition starts and ends at the limits of the ligature region produced by the junction's linked concavities.

T-junctions. A T-junction **connects** the two non-protruding branches incident to the fork and adjusts the associated stroke segments with a smooth transition. The junction also adjusts the stroke segment associated with the protruding branch with a straight transition. The transition extends the segment so it terminates at the intersection with the path produced by the smooth transition.

Y-junctions. A Y-junction **connects** the root to one of the other branches and leaves the other branch smoothly protruding from the connected path (inset, 1, 2). The choice of which branch is connected depends on the junction identification procedure detailed in Section 7.3. Other interpretations are possible: 3, with a multitraced root, and 4, with three separate strokes. We find that the first two are sufficient for our stylization purposes, but others might be useful in other scenarios. The junction adjusts the connected stroke segments with a smooth transition and the protruding one with a straight transition. Both transitions replace the ligature produced by the concavity and the disks centered within the fork.



L-junctions. An L-junction **discards** the root branch, connects the other two branches and adjusts the corresponding stroke segments with straight transitions that meet at a common point of intersection. The transitions have a constant radius $r_i = r_j$ determined by the radii at the edges of the ligature produced by the concavity.

Stroke-ends. A stroke-end **discards** the two least salient branches incident to the fork along with any attached branch hierarchies, and adjusts the third branch with a straight transition that extends the fork to a location near the outline.

Protuberances. A protuberance adjusts the branch assigned to the compound link with a straight transition and **discards** all the other non-salient branches associated with the link for which Eq. 2 is positive. The junction also **connects** the branches incident to the discarded ones.

Null-junctions. A null-junction **discards** the least salient branch incident to the fork and **connects** the other two branches incident to the fork.

7 JUNCTION IDENTIFICATION

Junctions often occur in complex and nested configurations and their identification becomes non-trivial. Similar to existing approaches for part decomposition [Papanelopoulos et al. 2019; Siddiqi and Kimia 1995], our identification method uses an iterative approach (Fig. 10) consisting of four main steps. First, we identify protuberances associated with compound links. Second, we use good continuation to identify half-junctions that cover forks in a single ligature region. We process these junctions early since they identify crossing paths that should not be disconnected by other subsequently identified junctions. Third, we identify all the remaining junction types by examining one $\mathbb{M}^I$ fork at a time. Fourth, we examine pairs of previously identified T-junctions, transforming some into half-junctions if the corresponding links have high good continuation value.

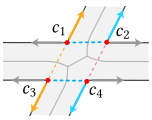
Auxiliary graph $\mathbb{H}$. Identifying a junction can be interpreted as recovering a part of a stroke, and this can change which concavities and links are meaningful for identifying the next junction. To manage these changes we use a graph $\mathbb{H} = (C, H)$ having one vertex per concavity and one edge per valid link. Each iteration of the identification procedure removes vertices (concavities) and edges (links) from $\mathbb{H}$ depending on the identified junction. Removing a vertex also removes all incident edges, affecting the subsequent identification of remaining junctions.

7.1 Step 1: Identify protuberances

In the first step, we identify a protuberance from each previously identified compound link. This step does not modify $\mathbb{H}$ but it localizes small protrusions, transforming these into small strokes that can later be associated with T-junction or a half-junction, and less often to another junction

7.2 Step 2: Identify half-junctions

Identifying half-junctions requires finding candidate link pairs in $\mathbb{H}$ that can be associated based on good continuation.



For two links η_i and η_j having concavities (c_1, c_2) and (c_3, c_4), we define the *connecting good-continuation value* $\psi(\eta_i, \eta_j)$ to be the product $\psi(c_1, c_3) \times \psi(c_2, c_4)$ for the non-crossing links connecting the endpoints of η_i and η_j.

Two links η_i, η_j, can be paired into a half-junction if they do not share a concavity and the connecting good-continuation value $\psi(\eta_i, \eta_j)$ is greater than a threshold, set experimentally to 0.25.

Candidate half-junctions can occur in ambiguous nested configurations. We consider a link in a pair to be *nested* if it is assigned a branch that is part of any crossing path defined by another pair (Fig. 13a). If this is the case, the nested pair is not considered a potential half-junction.

In some glyphs, particularly in hanzi, one stroke crosses another and ends in a short, rounded protrusion (Fig. 7e). In this case the tangents at the concavities do not always capture the perceived direction of stroke continuation, but we can detect this because one of the links η_i or η_j is assigned a non-salient branch. We then re-orient the tangents corresponding to its concavities to match the link flow direction.

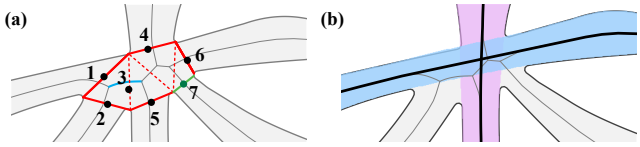


Fig. 13. Half-junction disambiguation. **(a)** The red colored links are all part of potential half-junction pairs. The dashed-red links are nested. For example link 3 is nested because its protruding branch (blue) is part of the crossing path between link 1 and link 6, which can be paired. Link 7 (green) is not paired with any other link i because $\psi(\eta_i, \eta_7)$ never exceeds the pairing threshold. **(b)** This configuration results in two half-junctions produced by the pairs $(1, 6)$ and $(4, 5)$

We identify an initial set of half-junctions by examining groups of links that are assigned forks that are part of a single ligature region. For each group we identify candidate link pairs η_i, η_j that are consistent with the conditions above and are not nested, and then process these pairs in order of decreasing good continuation $\psi(\eta_i, \eta_j)$. We create a half-junction only if a pair does not include a previously processed link.

7.2.1 Updating $\mathbb{H}$. Every time we identify a half-junction, we remove its two links from $\mathbb{H}$. We also remove any link that is assigned a branch that shares more than one vertex with the crossing path. This guarantees that the path will not be disconnected by a subsequently-identified junction.

7.3 Step 3: Identify other junctions

The five other junction types are identified one fork at a time with a procedure that depends on the links H_f and concavities C_f assigned to a given fork f, both of which are in $\mathbb{H}$. Our goal is to select a junction γ for f that produces a good approximation of the glyph near f, while also producing strokes that are smooth and consistent with the configurations of concavities C_f and links H_f. We evaluate a potential junction using four measures:

(1) *Coverage,* $\Lambda_{\mathbb{I}}$*:* Rewards strokes that provide a good cover of the corresponding glyph region (Section 7.3.1).
(2) *Smoothness,* Λ_ψ*:* Rewards T-, Y- or L-junctions that produce smooth strokes (Section 7.3.2).
(3) *Concavity significance* Λ_w*:* Rewards T-, Y- or L-junctions that are consistent with the configuration of concavities in C_f (Section 7.3.3).
(4) *Link salience* Λ_η*:* Rewards T-junctions that are characterized by a salient link (Section 7.3.4).

Coverage applies to all junction types, while *smoothness* and *concavity significance* only apply to T-, Y- and L-junctions, and *link salience* only applies to T-junctions.

This class-dependent organization of measures lets us disambiguate junctions without relying on training data [Plamondon and Srihari 2000] but poses the challenge of comparing measures with different ranges [Bailey 2001]. We adopt a heuristic solution akin to the "one-versus-one" classification [Galar et al. 2011] where, given N *candidate* junctions (Section 7.3.5), we evaluate all $N(N-1)/2$ pairwise junction combinations using the terms that are valid for both junction types (Section 7.3.6). We process forks iteratively with an ordering procedure (Section 7.3.7) that depends on H_f and C_f.

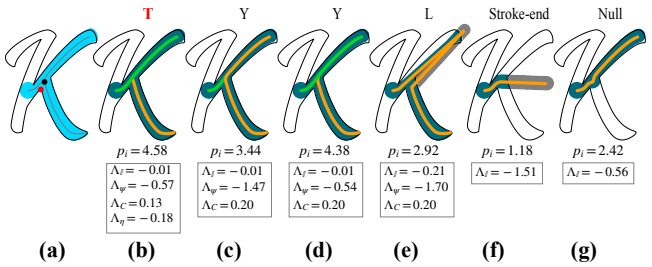


Fig. 14. Possible interpretations for the fork in this K (black dot) where the two diagonal strokes, shown in blue, join. **(a)** The blue area also shows the before-adjustment area used in the coverage calculation and the most salient concavity (red dot). The darkened areas in the remaining subfigures show the adjusted strokes. **(b)** The T-junction interpretation has the highest score and so is the one selected. **(c-d)** The most salient concavity (red dot) is below the fork (black dot), leading to the root for the two Y-junction interpretations being the branch that extends to the upper right. **(e)** It also forces the L-junction's corner, which must be opposite the most salient concavity, to be in the upper right and forcing the implausible structure shown. **(f)** In the stroke-end interpretation, the entire branch hierarchy to the right of the fork is evaluated as an elaborately flared end to the short branch connecting the fork to the vertical; because this is not plausible, the score is very low. Note that the two highest-scoring interpretations give the same stroke decomposition.

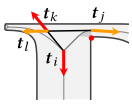
7.3.1 $\Lambda_{\mathbb{I}}$*: Coverage.* For each junction type, we seek to evaluate how well the resulting strokes cover the underlying glyph region. To do so, we consider the connected components of $\mathbb{S}$ that include any branch covered by the junction. We then rasterize the associated stroke segments before and after the operations induced by the junction and compute their respective areas A_γ and A'_γ (Fig. 14). The coverage is computed as:

$$\Lambda_{\mathbb{I}} = \ln\left[(A_\gamma \cap A'_\gamma)/A_\gamma\right] . \tag{4}$$

7.3.2 Λ_ψ*: Smoothness.* For T-, Y-. and L-junctions, we reward the junction type that maximizes stroke smoothness. We quantify smoothness as a geometric mean:

$$\Lambda_\psi = \ln\left[\left(\prod_{i=1}^{M}\psi_i\right)^{\frac{1}{M}}\right] , \tag{5}$$

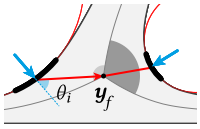
where M depends on the junction type and ψ_i are good continuation values computed along stroke spines. For T- and Y-junctions, $M = 1$ and ψ_i is the good continuation for the point-tangent pairs used to compute the smooth transition of the connected stroke.



For L-junctions, $M = 2$ and we compute two good continuation values, respectively between the tangents $\boldsymbol{t}_i, \boldsymbol{t}_j$ at the ends $\boldsymbol{y}_i, \boldsymbol{y}_j$ of the ligature and two other tangents $\boldsymbol{t}_k, \boldsymbol{t}_l$, parallel to the junction's straight transitions (inset, black lines) and anchored at their point of intersection.

For all cases we compute good continuation (Section 5.3) with spread parameter σ_d set to the distance between the tangent origins, which results in a measure invariant to distance.

7.3.3 Λ_w*: Concavity significance measure.* The inward normal $\boldsymbol{n}$ at a concavity c is similar to the "process arrow" proposed by Leyton [1988], which captures the direction of a force that produces the concavity when locally applied to an elastic version of the outline.



We use this analogy to compute the *significance* of a concavity c with respect to the fork position $\boldsymbol{y}_f$, by assuming that the glyph is made of an idealized linear-elastic, isotropic, incompressible material with a Poisson ratio of 0.5. Based on Flamant's solution for a normal force acting on an elastic half-plane [Kachanov et al. 2003], the normal *displacement* at $\boldsymbol{y}_f$ produced by a force applied at a point $\boldsymbol{x}_i$ along the contact region and oriented according to $\boldsymbol{n}$ is proportional to:

$$u = -0.5 \ln\left(\frac{R^2}{L^2}\right) + \frac{\cos^2\theta}{R^2} , \tag{6}$$

where θ_i is the angle between $\boldsymbol{n}$ and the vector from $\boldsymbol{x}_i$ to $\boldsymbol{y}_f$, R is the magnitude of this vector and L is a constant that determines a distance at which the displacement vanishes [Timoshenko and Goodier 1951]. We use $L = 2r_f$, where r_f is the disk radius at the fork. The significance of a concavity c (inset: grey sectors) is then

$$w(c, f) = u/r^{1/2} , \tag{7}$$

where r is the CSF radius. Intuitively, high significance for just *one* concavity in C_f suggests the presence of a Y- or L-junction, while high significance for *two* such concavities suggests the presence of a T-junction. To quantify this, we sort the concavity significances w_i for the three fork sectors (Section 4.4) in decreasing order, $w_1 \geq w_2 \geq w_3$, setting $w_i = 0$ if a sector has no concavity. We consider their differences normalized by the sum $\Sigma_w = w_1 + w_2 + w_3$ [Westin et al. 2002]. If we are evaluating the junction γ_i as a Y- or L-junction, then we compute:

$$\Lambda_w(\gamma_i) = \ln\left[(w_1 - w_2 + w_3)/\Sigma_w\right] , \tag{8}$$

where a high value of $\Lambda_w(\gamma_i)$ means that $w_1 > w_2 \simeq w_3$. If we are evaluating γ_i as a T-junction, then we compute:

$$\Lambda_w(\gamma_i) = \ln\left[(2w_2 - w_3)/\Sigma_w\right] , \tag{9}$$

where a high value of $\Lambda_w(\gamma_i)$ means that $w_1 \simeq w_2 > w_3$.

7.3.4 Λ_η*: Link salience .* This last measure is simply given by:

$$\Lambda_\eta = \ln \omega(\eta) , \tag{10}$$

where $\omega(\eta)$ is the salience of the junction's link (Eq.3).

7.3.5 Candidate junctions. For each fork f we build a set J_f of *candidate junctions*. Each such set always include a null-junction and a stroke-end, given by the least salient individual branch and the least salient branch pair incident to the given fork f. The other members of J_f depend on the presence and configuration of links and forks assigned to the fork, and on the saliency of the fork's incident branches.

If H_f (the set of forks assigned to f) is non-empty, J_f can contain a T-junction for each link in H_f. If C_f (the set of concavities assigned to f) is non-empty, we consider the two Y-junction configurations and single L-junction produced by the most significant concavity in C_f. The presence of T-, Y- and L-junctions in J_f depends on the following constraints:

(1) J_f contains a Y-junction only if the root is salient.
(2) J_f contains a T-, Y- or L-junction only if all the branches incident to the fork, with the exception of the root if present, have saliency greater than a lower bound $\beta_{\min}$.
(3) J_f contains an L-junction only if the representative concavity is *well aligned* with the root, i.e. the dot product between the root protruding direction and the concavity bisector is positive.
(4) J_f contains an L-junction only if the representative concavity has a curvature radius smaller than $\lambda_{\mathrm{L}} r_f \cos(\theta_c/2)$, with r_f the fork's radius and λ_{L} a user configurable multiplier experimentally set to 0.5.

Constraint 1 enforces consistency with the Y-junction definition. Constraint 2 functions similarly to a medial axis pruning strategy, but considers junction configurations to determine which $\mathbb{M}^I$ branches are not significant. We use a lower bound $\beta_{\min} = 1.5$ for the examples given. Increasing $\beta_{\min}$ will make flared stroke ends more likely to be categorized as stroke-end junctions and less likely to be separated as a serif with a T-junction. Constraint 3 avoids certain cases where a concavity being assigned to a fork can result in misidentifying a stroke-end or null-junction as an L-junction. Constraint 4 avoids certain cases in which corners are misidentified as null-junctions. While this last constraint is not critical to the recovery of plausible strokes, it improves stylization results in particular in cases that involve structural modifications of the stroke spines.

7.3.6 Identification. The preference for one junction γ_i among a pair (γ_i, γ_j) is given by:

$$\Lambda_{ij}(\gamma_i) = \Lambda_{\mathbb{I}} + \delta_{\mathrm{TYL}}\Lambda_{\psi} + \delta_{\mathrm{TYL}}\Lambda_{w} + \delta_{\mathrm{T}}\Lambda_{\eta} + \Lambda_{\gamma} \tag{11}$$

where $\delta_{\mathrm{TYL}} = 1$ if both γ_i and γ_j are one of a T-, Y- or L-junction, and $\delta_{\mathrm{T}} = 1$ if both are T-junctions. Otherwise, both terms are zero. The last term Λ_γ lets a user express a preference for the identification of certain junctions types with $\Lambda_\gamma = \ln \lambda_\gamma$, where λ_γ is a junction-dependent weight that defaults to 1. We find that it works well to use a slightly lower value of λ_γ for null-junctions, and a higher value of λ_γ for T-junctions. This generally favors L-junctions over null-junctions in certain corner configurations, and favours T-junctions over Y-junctions in the presence of a link. In the examples given, we use $\lambda_\gamma = 1.1$ for T-junctions and $\lambda_\gamma = 0.95$ for null-junctions. We finally estimate the probability of selecting a junction γ_i among a pair, with:

$$P_{ij}(\gamma_i) = \frac{\exp \Lambda_{ij}(\gamma_i)}{\exp \Lambda_{ij}(\gamma_i) + \exp \Lambda_{ij}(\gamma_j)} \tag{12}$$

and select the junction γ_i that maximises:

$$P_i = \sum_{1>=j\neq i<N} P_{ij}(\gamma_i). \tag{13}$$

7.3.7 Iterative process. We process forks in an order of decreasing *priority* given by

$$\begin{cases} \min\limits_{b\in B_f} \beta(b,f) \max\limits_{\eta\in H_f} \omega(\eta) \ + 2K, & \text{if } H_f \text{ is non-empty}, \\ \min\limits_{b\in B_f} \beta(b,f) \max\limits_{c\in C_f} w(c,f) \ + K, & \text{if } H_f \text{ is empty and } C_f \text{ is not}, \\ \min\limits_{b\in B_f} \beta(b,f), & \text{otherwise}, \end{cases} \tag{14}$$

where B_f is the set of branches incident to the fork , $w(c,f)$ is the significance of a concavity (Eq. 7), $\omega(\eta)$ is the salience of a link (Eq. 3) and K is an arbitrarily large constant that favours processing forks with assigned links before forks without a link, and forks with assigned concavities before forks without. Using the minimum branch salience terms $\beta(b,f)$ generally favours a depth-first processing of forks. Fig. 10 shows a typical sequence resulting from this ordering.

7.3.8 Updating $\mathbb{H}$. As with half-junctions, we remove vertices (concavities) and edges (links) from $\mathbb{H}$ after each junction identification. We remove a concavity from $\mathbb{H}$ if it is shared by two previously processed links that are assigned to the same fork, and if it is the representative concavity of a Y-junction. Finally, every time we process a T-junction, we test if the good continuation along the link is greater than a relatively high threshold (0.4 in the examples given). If this is the case, we remove both concavities from $\mathbb{H}$. This is based on the observation that separating the protrusion identified by the representative link can produce a locally flat region in the neighborhood of the discarded concavities.

7.4 Step 4: Convert T-junction pairs to half-junctions

In certain configurations such as the one emphasized in Fig. 15a, an area that is characterized by two ligature regions is perceived to be coverd by two crossing strokes. However, the steps described so far result in identifying two T-junctions that produce two strokes incident to a common stroke (Fig. 15b). We check if this kind of configuration can be transformed into one consisting of two crossing strokes (Fig. 15c), with a procedure similar to the one used for half-junctions in Section 7.2. If the distance between the incident endpoints of the two strokes is less than both radii of the forks

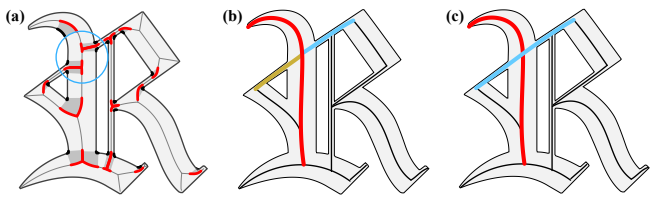


Fig. 15. Final identification of half-junctions. **(a)** The glyph region emphasied with the blue circle consists of two strokes that are perceived as crossing and have two separate ligatures. **(b)** The procedure up to Section 7.4 produces two T-junctions, which results in three strokes; the yellow and blue strokes protrude from the red stroke. **(c)** The links that characterize the two T-junctions have high good continuation, so the procedure in Section 7.4 transforms these junctions into one single half-junction replacing the previous two strokes (yellow and blue) with a single longer one (blue).

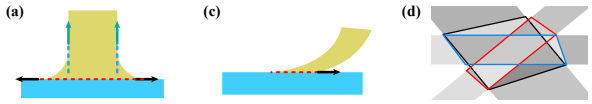


Fig. 16. Faces and edges of $\mathbb{Q}$ for different junction types. The tangents determining the edges are marked in black. **(a)** A T-junction adds one edge and produces two faces, one including the two arc segments of the concavities' contact regions. **(c)** A Y-junction adds one edge and produces two faces. **(d)** Three half-junctions in the same area, adding 12 edges (3 quadrilaterals).

covered by the T-junctions, and the good continuation $\psi(\eta_i, \eta_j)$ for the junction links is greater than the same threshold used in Section 7.2, we construct one half-junction from the two links and discard the two T-junctions. The result is similar to the one produced by two half-junctions, but in this case one half-junction connects two previously disconnected strokes, while one stroke has been already connected by the previously identified T-junction pair.

8 STROKE RECONSTRUCTION

The stroke graph $\mathbb{S}$, together with junctions, provides a flexible, high-level descriptor of the inferred stroke structure of a font. We use this descriptor in two methods to reconstruct strokes. Our first method produces strokes using the traditional definition of a stroke—a spine paired with a varying *width profile* function. The second produces *stroke areas*, a type of part-decomposition of the glyph consisting of potentially overlapping shapes that when unified closely reproduce the original glyph.

8.1 Strokes

To recover a spine and a width profile function from a connected component of $\mathbb{S}$ we consider the concatenated sequence of stroke segments associated with each branch in the component. Each sequence is akin a polyline in $\mathbb{R}^3$ with each coordinate $[\boldsymbol{y}_i, r_i]$ consisting of an adjusted position concatenated with an adjusted radius. To remove small discontinuities that can persist after the adjustment steps, we smooth the coordinates with a conventional spline method. We perform smoothing in a piecewise manner, with pieces delimited by the stroke endpoints and at L-junctions.

We also check for strokes that can be closely approximated with a straight spine and a constant width profile. To do so, we compute a linear least square fit of a 3D line to the coordinates and use this line if the mean-squared error of the fit is less than a user-configurable threshold.

8.2 Stroke areas

A stroke area is a 2D part of the glyph derived from a single stroke. Stroke areas enable stylizations that depend on the shape of the stroke parts and help quantify the accuracy of our segmentations (Section 9.1). They are created by using junctions to partition the input shape into disjoint faces and then using the connected components of the stroke graph to guide the assembly of these faces into stroke areas.

We construct a *planar map* [Fabri and Pion 2009; Preparata and Shamos 1985] $\mathbb{Q}$ built from the glyph outline with additional edges

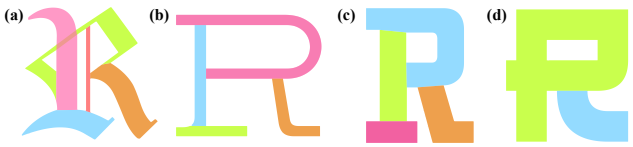


Fig. 17. Stroke areas for the letter R in different fonts. The last result is based on a stroke that crosses itself, producing a stroke area with a hole.

derived from the junctions. Each T-junction adds one edge to $\mathbb{Q}$, connecting the origins of tangents on the ends of its link (Fig. 16a). Each Y-junction also adds one edge to $\mathbb{Q}$.

We take the direction of one of the tangents of the junction's representative concavity and connect the concavity extremum to the first intersection with the outline. The tangent is the one that is least aligned with the protruding branch (Fig. 16c). A half-junction adds a quadrilateral to the graph. Two of its edges connect the tangent origins of the non-crossing link endpoint pairs, the same ones used to compute good continuation in Section 7.2. The other two edges connect the same tangent origins along the links (Fig. 16d).

Once $\mathbb{Q}$ has been constructed, we create one area for each stroke by performing a union of some of the faces in $\mathbb{Q}$. We first construct a *disk area* for each stroke. Each area is the union of the disks for the branches in the associated connected component of $\mathbb{S}$ as well as for any branches discarded by a junction that covers a branch in this connected component. We then assign any face enclosed by any of the quadrilaterals added by a half-junction to the stroke for that half-junction. Each remaining face is assigned to the stroke for which the intersection of the face and the stroke disk area is largest. Fig. 17 shows the stroke areas for the letter R in various fonts.

9 RESULTS AND DISCUSSION

Strokes and stroke areas provide the basis for evaluating our method with respect to ground truth segmentations and for producing a variety of stroke-based stylizations of the input glyphs.

9.1 Segmentation quality

Quantitative evaluation of the results of stroke segmentation is difficult because there is no ground truth for most Western fonts. However, we can compare the segmentation results with the *make-me-a-hanzi* dataset [Kishore 2018], which includes outline and stroke ground truth for a variety of simplified and traditional Chinese (Han) characters. To perform the evaluation, we first segment the glyph into stroke areas as described in the previous section.

Similarly to Kim et al. [2018] we then perform an "Intersection over Union" (IoU) test on the rasterized stroke areas. For each segmented stroke area, we identify the most similar stroke from ground truth by maximizing the intersection area. Rasterizing at a resolution of 512×512 gives an average per pixel accuracy of 0.979, which is slightly better then the result of 0.958 reported by Kim et al. [2018]. This result is influenced by some inaccuracies in the planar map edges (Fig. 18a) and by different stroke decompositions (Fig. 18b). We consider a stroke to be incorrect if its IoU is < 0.8, which excludes small errors like the one in Fig. 18a, and results in a per-stroke accuracy of 0.976.

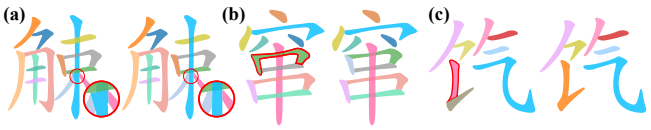


Fig. 18. Quantitative evaluation with the *make-me-a-hanzi* dataset; the ground truth is to the right of each pair. **(a)** Our method derives the same stroke structure as that of the ground truth but one T-junction (marked with a red circle) includes a stroke deformation. **(b)** All strokes are correctly identified by our method except for the middle area, emphasized in red. We derive one single stroke rather than the two in the ground truth because there is no salient concavity near the top left of that area. **(c)** Using $\lambda_\gamma = 1.$ for L-junctions results in a segmentation different from the ground truth. With an increased value of $\lambda_\gamma = 1.2$ the segmentation is identical to the ground truth, globally producing the results discussed in Section 9.1

Certain ground-truth decompositions cannot be deduced from the outline alone because they depend on domain knowledge. For example, "boxes" in Chinese characters should almost always be segmented into three strokes. Sometimes there are outline details that lead to a correct segmentation, but not always (Fig. 18b). In other cases, the segmentation result is sensitive to parameter choices. For example, with the parameters used throughout the paper for Western fonts, Fig. 18c results in a corner being interpreted as a Y-junction. For the *make-me-a-hanzi* dataset we found that using a value of $\lambda_\gamma = 1.2$ for L-junctions (Section 7.3.6) fixed cases like Fig. 18c and generally improved the segmentation results. With this parameter choice only 8% of the glyphs in the *make-me-a-hanzi* dataset had segmentation errors that were not of the types discussed above; 11% had errors that could not be avoided without domain knowledge or different parameter choices, and 81% were segmented identically to the ground truth. From a qualitative viewpoint, 100% of our segmentations produced strokes that create a readable reconstruction and robust stylization of the glyph, independent of the choice of parameters.

We further tested our method on 100 fonts and it generates plausible segmentation results in the vast majority of cases. The most encountered failure case is for very thick glyphs in which the average stroke thickness is larger than the average stroke length (Fig. 19), leading to a medial axis with branches that cannot readily be discarded as non-salient. The segmentation also gives useful results on other types of non-glyph shapes as long as there is a recoverable articulated or branching structure (Fig. 20). This suggests that our method could be useful to recover stroked paths from filled vector art or from scanned documents.

9.2 Stylization and animation

Strokes and junctions provide the basis for a variety of stylization methods. In particular, junctions provide semantic annotations that determine connectivity relations between strokes or features such as corners that can be preserved across stylizations. Grounding text stylization on fonts has a number of advantages: (i) the large variety of existing fonts provides a large variety of starting points for stylization, (ii) the method is agnostic to the language or writing system, and (iii) the embedded kerning information can be used to determine inter-glyph spacing, which is known to be difficult to achieve with methods that create stylized text from scratch [Haines et al. 2016].

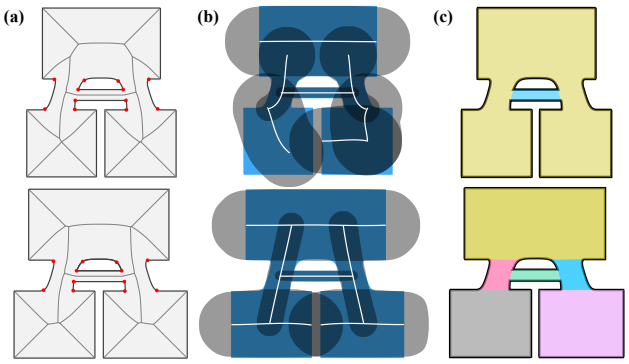


Fig. 19. Challenging case with a glyph in the Manicotti font. The first row shows the character "A", which produces non-salient features similar to stroke ends that , in **(a)**, do not produce a medial axis subtree that our method is able to identify as a stroke-end junction. These cases are difficult to detect with our current junction taxonomy and result, in **(b)**, in an inplausible stroke decomposition, and, in **(c)**, in an area decomposition that simply ignores the non-salient features. In the second row the ends of the top and bottom serifs have been slightly extended, disambiguating the medial axis and giving plausible strokes and areas.

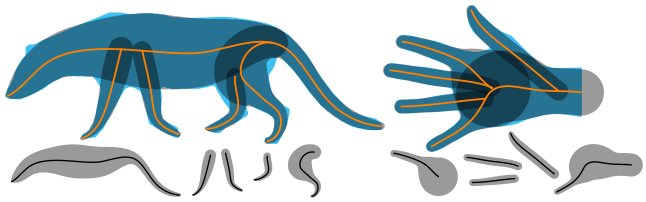


Fig. 20. Stroke decomposition of silhouettes. The left mammal silhouette (from the PhyloPic database, http://phylopic.org) results in strokes that capture its articulated structure. The right hand results in a plausible reconstruction, but the segmentation deviates from the perceived structure of a hand, with the pinkie being part of the same "stroke" as the palm.

9.2.1 Skeletal strokes. The spines recovered from the stroke graph can be directly used to produce some simple stylizations. Fitting Bézier curves to each spine can produce "Hershey fonts", which have glyphs consisting of constant-width strokes (Fig. 21). Such fonts are well-suited for fabrication and manufacturing applications. The same curves can be used as the spines for skeletal strokes [Hsu and Lee 1994], which enable a variety of glyph stylizations ranging from painterly to decorative (Fig. 22).

9.2.2 Schematization. We can also generate stylizations by simplifying the spines into sparse control polygons and schematizing the results. This can be done with a number of polygonal simplification methods; we use Discrete Contour Evolution [Latecki and Lakämper 1998] with a user-controlled threshold (Fig. 23). For a schematized stylization, we quantize the orientations of stroke segments using the C-oriented method [Nöllenburg 2014], which approximates a polyline with another one consisting of segments having a discrete

Fig. 21. Hershey font stylization (black) overlayed on the original font (gray).

Fig. 22. Font stylization with skeletal strokes. The left column shows the text in the original font. The right column shows the corresponding stroke stylizations. The first example on the right shows the result of using skeletal strokes as implemented in Adobe Illustrator to change weight, cap, and join styles; the other three show various decorative effects. The strokes in the bottom right example use variable width.

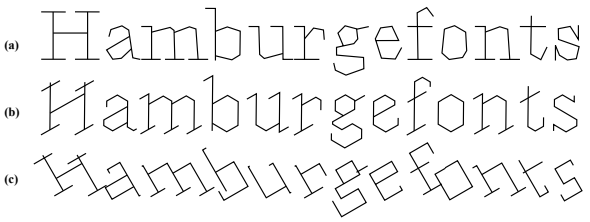


Fig. 23. Simplification and schematization of the strokes in Fig. 21: **(a)** Path simplification. Spine schematization [Dwyer et al. 2008]: **(b)** quantizing orientations to multiples of 60°; **(c)** restricting orientations to 30° and 120°.

set of orientations. We use the solution of Dwyer et al. [2008], which creates regular-looking polygonizations and stylistic abstractions of the letter structure (Fig. 23b,c). These also form the basis for other stylization techniques described below.

Structural adjustment. Schematization applies to each stroke separately. This can corrupt the glyph topology by disconnecting strokes that previously met at T- or Y-junctions. However, the connectivity information encoded by these junctions allows us to correct these errors by extending spines along their end-tangents until they intersect their opposite nearby spine. We use a similar procedure to maintain structural relations between strokes in the stylization methods that follow, with details given in Appendix C.

9.2.3 Calligraphic effects. The schematized or simplified vertices of a spine can be used as a motor plan for generating motion paths that mimic the aesthetics of certain kinds of calligraphic writing (Figs. 1e,f and 24). We generate smooth trajectories with the adaptive smoothing method of Berio et al. [2017] and increase dynamism by varying brush thickness depending on the synthesized trajectory

Fig. 24. Schematization and smoothing applied to strings in different fonts (Apollo ASM, Impact, Amador). The second example in **(c)** shows stroke ends extended for calligraphic effect. The second example in **(a)** and both examples in **(c)** use varying brush thickness derived from the stroke width profiles

speed [Berio et al. 2018], or depending on the width profile of a stroke (Fig. 24), similar to [Seah et al. 2005]). Different degrees of smoothing at the corners results in different calligraphic effects.

9.2.4 Graffiti art: The origins of graffiti styles can be traced back to more traditional forms of calligraphy and lettering as well as to certain types of fonts [Arte 2015]. We can simulate this contemporary art form, which features disctinctive forms of letter-stylization, with a variant of skeletal strokes [Berio et al. 2019] that mimics the intertwining, folding and extrusion effects that can be seen in traditional graffiti art (Fig. 25). The local width of each stroke can be determined by the previously computed thickness profile, and multiple strokes optionally can be combined with local union operations at T-, Y- junctions and half-junctions. We also use curve smoothing [Berio et al. 2017] and the rendering methods of Berio et al. [2019] to provide smooth color gradations, solid-colored blocks, and highlight effects common to contemporary graffiti (Fig. 26).

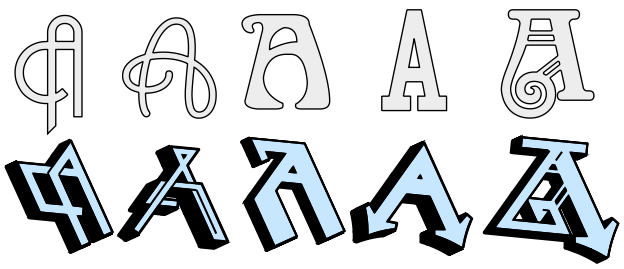


Fig. 25. Schematization and "graffiti strokes" [Berio et al. 2018] applied with the same parameters to the letter "A" in different fonts (Andes, Giddyup, Doctor Fibes, College, Apollo ASM). In the last two examples (bottom left), the serifs are replaced with arrow heads.

Fig. 26. Mimicking graffiti effects with the layering and rendering techniques of Berio et al. [2018].

9.2.5 Structural modifiers. T-, Y- and L-junctions identify strokes that can be altered for additional stylization effects. Strokes that do not terminate in one of these junctions can be extended to achieve various visual effects (Fig. 24, last row). Junctions also help identify serifs. We detect a serif as a relatively short, straight stroke that contains the connected portions of one single T-junction and does not contain any other Y- or L- junction. This detection can be exploited for stylization, for example, replacing serifs with arrow heads as is common in graffiti. (Fig. 25).

9.2.6 Stroke animation. Strokes can be used as motion paths to generate a variety of animation effects. The smoothing method of Berio et al. [2017] produces dense polylines, with distances between vertices reflecting the kinematics that are similar to human hand motion. This can be exploited to generate natural-looking stroke animations (Fig. 27). Stylized brush animations can also be generated by incrementally visualizing a stroke, or by animating a particle system that follows the stroke's path. We order strokes with a simple topological sorting heuristic rewarding top-to-bottom and left-to-right movements, but the strokes derived by our method

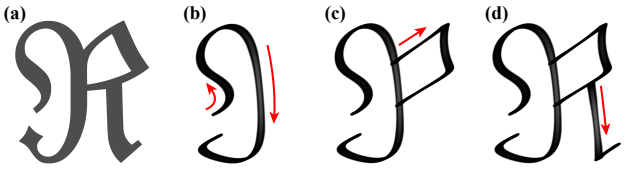


Fig. 27. Animating the drawing of a stylized R.

Fig. 28. Stylization based on similarity between stroke areas. In the first row, strokes are color-coded based on common clusters. In the second row, each stroke in a cluster is replaced with the same custom artwork. Note that including junction structure in the stroke similarity metric allows distinct stylizations to apply to otherwise-similar strokes, like the horizontal strokes in R, P, L, and A. Artwork ©Daichi Ito, Adobe Research. The same stroke areas can be used to drive other replacement-based stylization methods such as the one by Zhang et al. [2017].

are suitable for more sophisticated approaches [Fu et al. 2011; Tang et al. 2017].

9.2.7 Area-based stylization: Stroke similarity. The same stroke areas used to evaluate segmentation are also the basis of a similarity measure among strokes in a complete font. We compute the difference between two stroke areas by aligning their centroids, rasterizing them, and then measuring the *Jaccard distance* [Deza and Deza 2013, p. 299] between the resulting bitmaps: 1 minus the intersection divided by the union. If one stroke terminates in a T- or Y-junction and the other does not, the distance takes the maximum value of 1. We then group strokes using single-linkage agglomerative clustering and determine clusters based on a user-configurable threshold. While the distance is computed offline, the clustering procedure is interactive, and users can adjust the threshold to their preference. We then replace each stroke area in a cluster with an artistic rendering based on the shape, generating stylizations that apply uniformly across an entire font (Figs. 1i and 28).

9.3 Implementation details

The core segmentation procedure is written in the Python programming language, and includes the QHull library [Barber and Huhdanpaa 1995] to efficiently compute 2D Voronoi diagrams used for medial axes recovery. We executed our methods on a 2.7 GHz Intel Core i7 processor with four cores. Outline analysis and segmentation together take an average of 2 seconds per glyph; normally we precompute these for an entire font, but they could also be computed on demand and cached. The stylization procedures described in this section are written in C++ using OpenGL for hardware-accelerated rendering. They run in real-time and allow exploring different stylizations through an interactive user interface.

10 CONCLUSION

In this paper we first presented concepts and algorithms to automatically segment font glyphs into strokes, and then demonstrated how such strokes can be used to generate in real time a variety of stylizations of the input. We introduced along the way a number of innovations, namely:

(1) Using Curvilinear Shape Features (CSFs) that describe convex and concave outline regions.
(2) Links that connect concave CSFs across the body of a glyph and describe potential stroke crossings.
(3) A set of seven junction types—half-, T-, Y-, L-, stroke end, protuberance, and null—that characterize $\mathbb{M}^I$ sub-structures and determine semantic stroke attributes useful to drive structurally-aware stylizations of a glyph.
(4) A novel application of association fields [Ernst et al. 2012] to the problem of stroke segmentation of glyphs.

Our system, StrokeStyles, solves a long-standing inverse problem of segmenting 2D font glyphs [Wang 2013, §4]. The strokes we can recover are based on spines and profile functions and enable a variety of stylization methods including skeletal strokes [Hsu et al. 1993], animation, calligraphy, and graffiti.

This paper did not compare our stylizations with the ones produced by existing methods because our objective of producing stylization based upon the glyph structure is novel. Our real-time stylization framework provides a "sandbox" in which a user/designer can explore many different stylization options, ranging from readable stylisations to highly abstract renditions that still evoke the original font structure. This applies especially to the calligraphic and graffiti stylisations, which can operate in a domain where aesthetics take priority over readability [Craveiro 2017].

Stroke segmentation can also be useful in related applications like automatic font hinting [Shamir 2003], segmenting characters in historical documents [Lamiroy et al. 2015], painterly applications of robotic [Deussen et al. 2012], stylization methods that require taking glyph structure into account [Zou et al. 2016], animated reconstructions of arbitrary glyphs [Gingold et al. 2008] and producing training data for sequence-based generative models [Ha and Eck 2018; Kotani et al. 2020].

Data-driven approaches based on deep-learning typically rely on a large body of human-labelled training data. We instead demonstrate a solution that relies on experimentally-validated principles of visual perception and computational geometry concepts. The advantage of our approach is that it is adaptable to fonts for which training data might be scarce or non-existent and to glyphs that do not match the training data. Our solution requires tuning a few parameters, but these have intuitive visual and perceptual interpretations and can be adjusted by the user for the required use case.In Section 9.1 for example, we have adjusted the system parameters to increase segmentation accuracy for Han characters. Such parameter changes could also be determined depending on information encoded as font metadata.

In future research, we plan to explore how data-driven solutions could be combined with our approach. For example, we could use data to incorporate language-specific domain knowledge. More specifically, we could add a data-driven term to Eq. 11 to help identify junctions while still using the geometric and perceptual factors that we have defined. In our experiments we also have considered using measures such as stroke-radius variation or quality of fit to concave glyph areas. We finally selected the measures described in Section 7.3 as a sufficient minimum to produce plausible stroke segmentation for the glyphs we tested.

REFERENCES

Ery Arias-Castro, Gilad Lerman, and Teng Zhang. 2017. Spectral Clustering Based on Local PCA. *Journal of Machine Learning Research* 18 (2017), 1–57.

Anssi Arte. 2015. *Forms of Rockin': Graffiti Letters and Popular Culture.* Dokument Press.

Jonas August, Kaleem Siddiqi, and Steven W. Zucker. 1999. Ligature Instabilities in the Perceptual Organization of Shape. *Computer Vision and Image Understanding* 76, 3 (1999), 231–243. https://doi.org/10.1006/cviu.1999.0802

Samaneh Azadi, Matthew Fisher, Vladimir G Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. 2018. Multi-Content GAN for Few-Shot Font Style Transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7564–7573. https://doi.org/10.1109/CVPR.2018.00789

Alex Bailey. 2001. *Class-dependent features and multicategory classification.* Ph. D. Dissertation. Southampton Univ. (United Kingdom).

Elena Balashova, Amit H. Bermano, Vladimir G. Kim, Stephen DiVerdi, Aaron Hertzmann, and Thomas Funkhouser. 2019. Learning a Stroke-Based Representation for Fonts. *Computer Graphics Forum* 38, 1 (2019), 429–442. https://doi.org/10.1111/cgf.13540

Brad Barber and H Huhdanpaa. 1995. QHull. *The Geometry Center, University of Minnesota, http://www. geom. umn. edu/software/qhull* (1995).

Alexander Belyaev and Shin Yoshizawa. 2001. On Evolute Cusps and Skeleton Bifurcations. In *International Conference on Shape Modeling and Applications*. IEEE, 134–140. https://doi.org/10.1109/SMA.2001.923384

Daniel Berio, Paul Asente, Jose Echevarria, and Frederic Fol Leymarie. 2019. Sketching and Layering Graffiti Primitives. In *8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. 51–59. https://doi.org/10.2312/exp.20191076

Daniel Berio, Sylvain Calinon, and Frederic Fol Leymarie. 2017. Dynamic Graffiti Stylisation with Stochastic Optimal Control. In *Proceedings of the 4th International Conference on Movement Computing*. Association for Computing Machinery. https://doi.org/10.1145/3077981.3078044 Article no. 18.

Daniel Berio, Frederic Fol Leymarie, and Réjean Plamondon. 2018. Expressive Curve Editing with the Sigma Lognormal Model. In *Proceedings of the 39th Annual European Association for Computer Graphics Conference: Short Papers*. Eurographics Association, 33–36.

Daniel Berio, Frederic Fol Leymarie, and Réjean Plamondon. 2020. Kinematics Reconstruction of Static Calligraphic Traces from Curvilinear Shape Features. In *The Lognormality Principle and its Applications in e-Security, e-Learning and e-Health*, Réjean Plamondon, Angelo Marcelli, and Miguel Ángel Ferrer (Eds.). Series in Machine Perception and Artificial Intelligence, Vol. 88. Chapter 11, 237–268. https://doi.org/10.1142/9789811226830_0011

Harry Blum. 1973. Biological Shape and Visual Science (Part I). *Journal of Theoretical Biology* 38, 2 (1973), 205–287. https://doi.org/10.1016/0022-5193(73)90175-6

Joseph L Brooks. 2015. Traditional and New Principles of Perceptual Grouping. (2015), 57–87.

Neill DF Campbell and Jan Kautz. 2014. Learning a Manifold of Fonts. *ACM Transactions on Graphics (TOG)* 33, 4 (2014). https://doi.org/10.1145/2601097.2601212 Article no. 91.

Xudong Chen, Zhouhui Lian, Yingmin Tang, and Jianguo Xiao. 2017. An Automatic Stroke Extraction Method Using Manifold Learning. In *Proceedings of the European Association for Computer Graphics: Short Papers (EG '17)*. Eurographics Association, 65–68.

Charles H Cox, Philippe Coueignoux, Barry Blesser, and Murray Eden. 1982. Skeletons: A Link Between Theoretical and Physical Letter Descriptions. *Pattern Recognition* 15, 1 (1982), 11–22. https://doi.org/10.1016/0031-3203(82)90056-5

Rodrigo Pena Carvalho Dos Anjos Craveiro. 2017. The Influence of Graffiti Writing in Contemporary Typography. *SAUC — Street Art and Urban Creativity Scientific Journal* 3, 2 (2017), 65–83. https://doi.org/10.25765/sauc.v3i2.82

Joeri De Winter and Johan Wagemans. 2006. Segmentation of Object Outlines Into Parts: A Large-Scale Integrative Study. *Cognition* 99, 3 (2006), 275–325. https://doi.org/10.1016/j.cognition.2005.03.004

Oliver Deussen, Thomas Lindemeier, Sören Pirk, and Mark Tautzenberger. 2012. Feedback-Guided Stroke Placement for a Painting Machine. In *8th Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*. 25–33.

Shay Deutsch and Gérard Medioni. 2017. Learning the Geometric Structure of Manifolds with Singularities Using the Tensor Voting Graph. *Journal of Mathematical Imaging*

and Vision 57, 3 (2017), 402–422. https://doi.org/10.1007/s10851-016-0684-2

Michel Marie Deza and Elena Deza. 2013. *Encyclopedia of Distances*. Springer. https://doi.org/10.1007/978-3-642-30958-8 Updated and revised second edition.

Tim Dwyer, Nathan Hurst, and Damian Merrick. 2008. A Fast and Simple Heuristic for Metro Map Path Simplification. In *International Symposium on Visual Computing*. Springer, 22–30. https://doi.org/10.1007/978-3-540-89646-3_3

Udo A. Ernst, Sunita Mandon, Nadja Schinkel–Bielefeld, Simon D. Neitzel, Andreas K. Kreiter, and Klaus R. Pawelzik. 2012. Optimality of Human Contour Integration. *PLOS Computational Biology* 8, 5 (2012), 1–17. https://doi.org/10.1371/journal.pcbi.1002520

Andreas Fabri and Sylvain Pion. 2009. CGAL: The Computational Geometry Algorithms Library. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09)*. 538––539. https://doi.org/10.1145/1653771.1653865

Alexandre Faure, Lilian Buzer, and Fabien Feschet. 2009. Tangential cover for thick digital curves. *Pattern Recognition* 42, 10 (2009), 2279–2287. https://doi.org/10.1016/j.patcog.2008.11.009

Jean-Dominique Favreau, Florent Lafarge, and Adrien Bousseau. 2016. Fidelity vs. Simplicity: A Global Approach to Line Drawing Vectorization. *ACM Transactions on Graphics (TOG)* 35, 4 (2016). https://doi.org/10.1145/2897824.2925946 Article no. 120.

Vicky Froyen, Jacob Feldman, and Manish Singh. 2015. Bayesian Hierarchical Grouping: Perceptual Grouping as Mixture Estimation. *Psychological Review* 122, 4 (2015), 575–597. https://doi.org/10.1037/a0039540

Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy J Mitra. 2011. Animated Construction of Line Drawings. In *ACM Transactions on Graphics (TOG)*, Vol. 30. 1–10. https://doi.org/10.1145/2070781.2024167

Mikel Galar, Alberto Fernández, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. 2011. An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition* 44, 8 (2011), 1761–1776. https://doi.org/10.1016/j.patcog.2011.01.017

Étienne Ghys, Sergei Tabachnikov, and Vladlen Timorin. 2013. Osculating Curves: Around the Tait-Kneser Theorem. *The Mathematical Intelligencer* 35, 1 (2013), 61–66. https://doi.org/10.1007/s00283-012-9336-6

Peter J. Giblin and Benjamin B. Kimia. 2003. On the Local Form and Transitions of Symmetry Sets, Medial Axes, and Shocks. *International Journal of Computer Vision* 54, 1 (Aug 2003), 143–157. https://doi.org/10.1109/ICCV.1999.791246

Yotam Gingold, David Salesin, and Denis Zorin. 2008. *Stroke-by-Stroke Glyph Animation*. Technical Report. Creativity and Graphics Lab (CraGL) at George Mason University, Fairfax, Virginia, USA. https://cragl.cs.gmu.edu/fontanim/.

Andrew Goldberg, Xiaojin Zhu, Aarti Singh, Zhiting Xu, and Robert Nowak. 2009. Multi-Manifold Semi-Supervised Learning. In *12th International Conference on Artificial Intelligence and Statistics*. 169–176. http://proceedings.mlr.press/v5/goldberg09a.html

David Ha and Douglas Eck. 2018. A Neural Representation of Sketch Drawings. In *Sixth International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1704.03477.

Tom SF Haines, Oisin Mac Aodha, and Gabriel J Brostow. 2016. My Text in Your Handwriting. *ACM Transactions on Graphics (TOG)* 35, 3 (2016). https://doi.org/10.1145/2886099 Article no. 26.

Katherine A Heller and Zoubin Ghahramani. 2005. Bayesian Hierarchical Clustering. In *22nd International Conference on Machine learning (ICML)*. ACM, 297–304. https://doi.org/10.1145/1102351.1102389

Jacky Herz, Roger D Hersch, and Jakob Gonczarowski. 1997. Coherent Processing of Character Skeletal Forms. *Computers and Graphics* 21, 6 (1997), 727–736. https://doi.org/10.1016/S0097-8493(97)00050-2

Donald D Hoffman and Whitman A Richards. 1984. Parts of Recognition. *Cognition* 18, 1-3 (1984), 65–96. https://doi.org/10.1016/0010-0277(84)90022-2

Donald D Hoffman and Manish Singh. 1997. Salience of Visual Parts. *Cognition* 63, 1 (1997), 29–78. https://doi.org/10.1016/S0010-0277(96)00791-3

Douglas R Hofstadter. 1982. Variations on a Theme as the Essence of Imagination. *Scientific American* 247, 4 (1982), 14–21.

Siu Chi Hsu and Irene H. H. Lee. 1994. Drawing and Animation Using Skeletal Strokes. *21st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (1994), 109–118. https://doi.org/10.1145/192161.192186

S. C. Hsu, I. H. H. Lee, and N. E. Wiseman. 1993. Skeletal Strokes. In *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology* (Atlanta, Georgia, USA) *(UIST '93)*. 197–206. https://doi.org/10.1145/168642.168662

Changyuan Hu and Roger D Hersch. 2001. Parameterizable Fonts Based on Shape Components. *IEEE Computer Graphics and Applications* 21, 3 (2001), 70–85. https://doi.org/10.1109/38.920629

Elena J. Jakubiak, Ronald N. Perry, and Sarah F. Frisken. 2006. An Improved Representation for Stroke-Based Fonts. In *ACM SIGGRAPH 2006 Sketches*. https://doi.org/10.1145/1179849.1180020

Tingting Jiang, Zhongqian Dong, Chang Ma, and Yizhou Wang. 2013. Toward Perception-Based Shape Decomposition. In *Computer Vision – ACCV 2012*. Lecture Notes in Computer Science, Vol. LNCS 7725. Springer, 188–201. https://doi.org/10.1007/978-3-642-37444-9_15

Mark Kachanov, Boris Shafiro, and Igor Tsukrov. 2003. *Handbook of Elasticity Solutions*. Springer Netherlands. https://doi.org/10.1007/978-94-017-0169-3

Peter Karow. 1994. *Digital Typefaces: Description and Formats*. Springer. https://doi.org/10.1007/978-3-642-78105-6

Byungsoo Kim, Oliver Wang, A Cengiz Öztireli, and Markus Gross. 2018. Semantic Segmentation for Line Drawing Vectorization Using Neural Networks. *Computer Graphics Forum* 37, 2 (2018), 329–338. https://doi.org/10.1111/cgf.13365

Shaunak Kishore. 2018. Make Me a Hanzi Dataset. https://github.com/skishore/makemeahanzi. www.skishore.me/makemeahanzi/

Donald E. Knuth. 1979. Mathematical Typography. *Bull. Amer. Math. Soc.* 1, 2 (1979), 337–373. https://doi.org/10.1090/S0273-0979-1979-14598-1

Atsunobu Kotani, Stefanie Tellex, and James Tompkin. 2020. Generating Handwriting via Decoupled Style Descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 764–780.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. 2015. Human-Level Concept Learning Through Probabilistic Program Induction. *Science* 350, 6266 (2015), 1332–1338. https://doi.org/10.1126/science.aab3050

Bart Lamiroy, Thomas Bouville, Julien Blégean, Hongliu Cao, Salah Ghamizi, Romain Houpin, and Matthias Lloyd. 2015. Re-typograph Phase I: A Proof-of-Concept for Typeface Parameter Extraction from Historical Documents. In *Document Recognition and Retrieval XXII*, Eric K. Ringger and Bart Lamiroy (Eds.), Vol. 9402. International Society for Optics and Photonics, SPIE, 80–91.

Longin Jan Latecki and Rolf Lakämper. 1998. Discrete Approach to Curve Evolution. In *Mustererkennung 1998*. Springer, 85–92. https://doi.org/10.1007/978-3-642-72282-0_7

R.L. Levien. 2009. *From Spiral to Spline: Optimal Techniques in Interactive Curve Design*. Ph. D. Dissertation. EECS Department, University of California, Berkeley. PhD thesis, EECS Department, University of California, Berkeley.

Michael Leyton. 1987. Symmetry-Curvature Duality. *Computer Vision, Graphics, and Image Processing* 38, 3 (1987), 327–341. https://doi.org/10.1016/0734-189X(86)90087-3

Michael Leyton. 1988. A process-grammar for shape. *Artificial Intelligence* 34, 2 (March 1988), 213–247. https://doi.org/10.1016/0004-3702(88)90039-2

Lei Luo, Chunhua Shen, Xinwang Liu, and Chunyuan Zhang. 2015. A Computational Model of the Short-Cut Rule for 2D Shape Decomposition. *IEEE Transactions on Image Processing* 24, 1 (2015), 273–283. https://doi.org/10.1109/TIP.2014.2376188

Diego Macrini, Sven Dickinson, David Fleet, and Kaleem Siddiqi. 2011. Bone Graphs: Medial Shape Parsing and Abstraction. *Computer Vision and Image Understanding* 115, 7 (July 2011), 1044–1061. https://doi.org/10.1016/j.cviu.2010.12.011

Diego Macrini, Kaleem Siddiqi, and Sven Dickinson. 2008. From Skeletons To Bone Graphs: Medial Abstraction for Object Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. https://doi.org/10.1109/CVPR.2008.4587790

Xiaofeng Mi and Doug DeCarlo. 2007. Separating Parts From 2D Shapes Using Relatability. In *IEEE 11th International Conference on Computer Vision (ICCV)*. https://doi.org/10.1109/ICCV.2007.4409014

Martin Nöllenburg. 2014. A Survey on Automated Metro Map Layout Methods. https://i11www.iti.kit.edu/extra/publications/n-asamm-14.pdf. In *1st Schematic Mapping Workshop*. University of Essex, UK. https://sites.google.com/site/schematicmapping/home.

Gerrit Noordzij. 2005. *The Stroke — theory of writing*. Hyphen Press. Translated from the Dutch original of 1985 by Peter Enneson.

Robert L Ogniewicz and Markus Ilg. 1992. Voronoi Skeletons: Theory and Applications. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 63–69. https://doi.org/10.1109/CVPR.1992.223226

Nikos Papanelopoulos, Yannis Avrithis, and Stefanos Kollias. 2019. Revisiting the Medial Axis for Planar Shape Decomposition. *Computer Vision and Image Understanding* 179 (2019), 66–78. https://doi.org/10.1016/j.cviu.2018.10.007

Pierre Parent and Steven W. Zucker. 1989. Trace Inference, Curvature Consistency, and Curve Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11, 8 (1989), 823–839. https://doi.org/10.1109/34.31445

Huy Quoc Phan, Hongbo Fu, and Antoni B Chan. 2015. Flexyfont: Learning Transferring Rules for Flexible Typeface Synthesis. In *Computer Graphics Forum*, Vol. 34. 245–256. https://doi.org/10.1111/cgf.12763

R. Plamondon and S. N. Srihari. 2000. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 1 (2000), 63–84. https://doi.org/10.1109/34.824821

Franco P. Preparata and Michael Ian Shamos. 1985. *Intersections*. 266–322. https://doi.org/10.1007/978-1-4612-1098-6_7

Hock Soon Seah, Zhongke Wu, Feng Tian, Xian Xiao, and Boya Xie. 2005. Artistic Brushstroke Representation and Animation with Disk B-Spline Curve. In *ACM SIGCHI International Conference on Advances in Computer Entertainment Technology*. 88––93. https://doi.org/10.1145/1178477.1178489

Doron Shaked and Alfred M Bruckstein. 1998. Pruning Medial Axes. *Computer Vision and Image Understanding* 69, 2 (1998), 156–169. https://doi.org/10.1006/cviu.1997.0598

Ariel Shamir. 2003. Constraint-Based Approach for Automatic Hinting of Digital Typefaces. *ACM Transactions on Graphics (TOG)* 22, 2 (2003), 131–151. https://doi.org/10.1145/636886.636887

Ariel Shamir and Ari Rappoport. 1996. Extraction of Typographic Elements From Outline Representations of Fonts. *Computer Graphics Forum* 15, 3 (1996), 259–268. https://doi.org/10.1111/1467-8659.1530259

Kaleem Siddiqi and Benjamin B Kimia. 1995. Parts of Visual Form: Computational Aspects. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17, 3 (1995), 239–251. https://doi.org/10.1109/34.368189

Manish Singh and Donald D Hoffman. 2001. Part-Based Representations of Visual Shape and Implications for Visual Cognition. In *Advances in Psychology*. Vol. 130. 401–459. https://doi.org/10.1016/S0166-4115(01)80033-9

Manish Singh, Gregory D Seyranian, and Donald D Hoffman. 1999. Parsing Silhouettes: The Short-Cut Rule. *Perception and Psychophysics* 61, 4 (1999), 636–660. https://doi.org/10.3758/BF03205536

Patrick Spröte, Filipp Schmidt, and Roland W Fleming. 2016. Visual Perception of Shape Altered by Inferred Causal History. *Scientific Reports* 6, 36245 (2016). https://doi.org/10.1038/srep36245

Yuandong Sun, Huihuan Qian, and Yangsheng Xu. 2014. A Geometric Approach to Stroke Extraction for the Chinese Calligraphy Robot. In *IEEE International Conference on Robotics and Automation (ICRA)*. 3207–3212. https://doi.org/10.1109/ICRA.2014.6907320

Rapee Suveeranont and Takeo Igarashi. 2010. Example-Based Automatic Font Generation. In *Smart Graphics*. Number LNCS 6133 in Lecture Notes in Computer Science. 127–138. https://doi.org/10.1007/978-3-642-13544-6_12

Fan Tang, Weiming Dong, Yiping Meng, Xing Mei, Feiyue Huang, Xiaopeng Zhang, and Oliver Deussen. 2017. Animated Construction of Chinese Brush Paintings. *IEEE Transactions on Visualization and Computer Graphics* 24, 12 (2017), 3019–3031. https://doi.org/10.1109/TVCG.2017.2774292

S P Timoshenko and J N Goodier. 1951. *Theory of Elasticity*. McGraw-Hill. https://books.google.co.uk/books?id=11ISAAAAIAAJ

Johan Wagemans. 2018. Perceptual Organization. In *Stevens' Handbook of Experimental Psychology and Cognitive Neuroscience, Sensation, Perception, and Attention*. Vol. 2. Chapter 18, 803–872. https://doi.org/10.1002/9781119170174.epcn218 4th Edition.

Johan Wagemans, Andrea J van Doorn, and Jan J Koenderink. 2011. Measuring 3D Point Configurations in Pictorial Space. *i-Perception* 2, 1 (2011), 77–111. https://doi.org/10.1068/i0420

Jue Wang, Chenyu Wu, Ying-Qing Xu, Heung-Yeung Shum, and Liang Ji. 2002. Learning-Based Cursive Handwriting Synthesis. In *Eighth IEEE International Workshop on Frontiers in Handwriting Recognition*. 157–162. https://doi.org/10.1109/IWFHR.2002.1030902

Yue Wang. 2013. Interview with Charles Bigelow. *TUGboat* 34, 2 (2013), 136–167.

Carl-Fredrik Westin, Stephan E Maier, Hatsuho Mamata, Arya Nabavi, Ferenc A Jolesz, and Ron Kikinis. 2002. Processing and Visualization for Diffusion Tensor MRI. *Medical Image Analysis* 6, 2 (2002), 93–108. https://doi.org/10.1016/S1361-8415(02)00053-1

Lance Williams and Karvel K Thornber. 2001. Orientation, Scale, and Discontinuity as Emergent Properties of Illusory Contour Shape. *Neural Computation* 13, 8 (Aug 2001), 1683–1711. https://doi.org/10.1162/08997660152469305

Songhua Xu, Hao Jiang, Francis CM Lau, and Yunhe Pan. 2012. Computationally Evaluating and Reproducing the Beauty of Chinese Calligraphy. *IEEE Intelligent Systems* 3 (2012), 63–72. https://doi.org/10.1109/MIS.2012.46

Yaoda Xu and Manish Singh. 2002. Early Computation of Part Structure: Evidence From Visual Search. *Perception and Psychophysics* 64, 7 (2002), 1039–1054. https://doi.org/10.3758/BF03194755

Shih Cheng Yen and Leif H. Finkel. 1998. Extraction of Perceptually Salient Contours by Striate Cortical Networks. *Vision Research* 38, 5 (1998), 719–741. https://doi.org/10.1016/S0042-6989(97)00197-1

Junsong Zhang, Yu Wang, Weiyi Xiao, and Zhenshan Luo. 2017. Synthesizing Ornamental Typefaces. *Computer Graphics Forum* 36, 1 (2017), 64–75. https://doi.org/10.1111/cgf.12785

Zhiyuan Zhao and Alan Saalfeld. 1997. Linear-time sleeve-fitting polyline simplification algorithms. In *Proceedings of the 13th AutoCarto symposium*, Vol. 13. https://cartogis.org, 214–223.

Changqing Zou, Junjie Cao, Warunika Ranaweera, Ibraheem Alhashim, Ping Tan, Alla Sheffer, and Hao Zhang. 2016. Legible Compact Calligrams. *ACM Transactions on Graphics (TOG)* 35, 4, Article 122 (2016), 12 pages. https://doi.org/10.1145/2897824.2925887 Article no. 122.

A CSF COMPUTATIONS

When searching for additional CSFs (§4.2.1), we need to avoid false positives, particularly those associated with outline segments that approximate spirals. A spiral is a curve segment with monotonically-varying curvature. Such a curve does not have any curvature extrema between its ends [Leyton 1987] and thus should not produce an additional CSF. This can be further characterized by the *Tait-Kneser* theorem [Ghys et al. 2013], which states that all osculating circles of a spiral segment with strictly positive or negative curvature are *disjoint* and *nested*. However, because CSF analysis operates on a sampled curve, looking for additional CSFs for an outline segment that closely resembles a spiral is likely to produce many additional terminal branches and spurious CSFs (Fig. 29a,b). To avoid such false positives, we compute the *degree of overlap* $\delta_C \in [0, 1]$ between any two discs as the area of their intersection divided by the area of the smaller disk. We then discard any new terminal disk if there is a pre-existing CSF with a smaller disk radius and for which the degree of overlap is greater than a user-defined threshold, which we empirically set to 0.98 (Fig. 29c).

Once we have identified the CSFs for a given outline, we compute a pair of tangents for each concave CSF (§4.2.2). To evaluate the tangents, we compute a tangent cover along each support segment, from which we keep the first tangents next to the ends of the contact region. We use "sleeve fitting" [Zhao and Saalfeld 1997] for this purpose. The more recent "alpha thick segments" technique [Faure et al. 2009] could also be used.

B ASSOCIATION FIELDS

Our association fields are adapted from Ernst et al. [2012]. The model predicts the conditional link probability of one oriented element relative to another. The link probability α is given by the product $A^{\phi}A^{d}$ of an angular and a radial component. The angular component parameterizes deviations from perfect cocircularity and deviations from zero curvature with the product of two von Mises distributions, analogs of Gaussian distributions with a circular support. Given two orientations ϕ_i, ϕ_j and planar positions $(x_i, y_i), (x_j, y_j)$ the angular component simplifies to:

$$A^{\phi} = \frac{C}{4} \cosh\left(\frac{1}{\sigma_{\beta}^2} \cos\left(\beta/2\right) + \frac{1}{\sigma_{\theta}^2} \cos\left(\theta - \beta/2\right) \right) \quad , \qquad (15)$$

with $\beta = \phi_j - \phi_j, \theta = \tan^{-1}\left(\left(y_j - y_i\right)/\left(x_j - x_i\right)\right) - \phi_i$, and $\sigma_{\theta} = 0.27$ and $\sigma_{\beta} = 0.47$ the spread parameters for cocircularity and curvature.[3] We use the spread parameter values that were experimentally

[3]This equation corrects a typographic error in the original paper

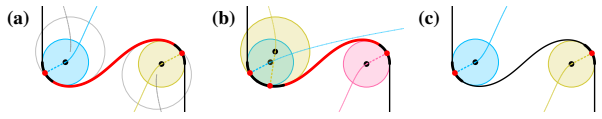


Fig. 29. Overlapping disks along a spiral segment. **(a)** the segment in red between the contact regions of the two CSFs is a spiral. However its local medial axis has two branches producing two terminal disks, shown in gray. **(b)** Without filtering, the left disk produces an additional CSF, since it is slightly more salient than the other disk. **(c)** However, the disk fully encloses the previously identified one so it is discarded. This results in the spiral segment not producing any new CSF.

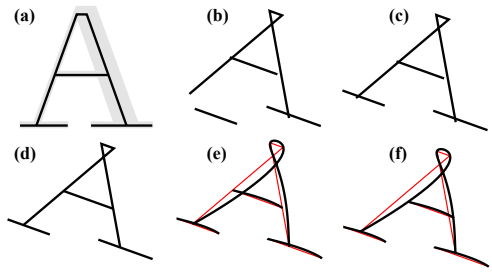


Fig. 30. Structural adjustment steps for a stylized letter "A". **(a)** Unstylized stroke spines after segmentation. **(b)** Schematization can corrupt topological relations among strokes. **(c)** We reestablish these by shifting strokes that are covered by a single T-junction or branching Y-junction. Note that the triangular part of the "A" is covered by two T-junctions, so it is not adjusted. **(d)** A second adjustment step reconnects all stroke endpoints. **(e)** Using the schematized stroke as a control polygon for a smoothing method can also corrupt the incidence relations among strokes. **(f)** A last adjustment step moves the non-smoothed polygon endpoints (the middle section of the "A") so they terminate at the intersection with the smoothed strokes.

found to be optimal by Ernst et al. [2012]. The constant C is a normalization factor derived from the von Mises distribution, with:

$$C = \pi^2 I_0\left(1/\sigma_a^2\right) I_0\left(1/\sigma_b^2\right) \quad , \tag{16}$$

where I_0 is the modified-Bessel function of the first kind with order 0. We also divide A^{ϕ} by 0.602, so it falls in the $[0, 1]$ range, which facilitates parameter setting in our application-driven use case.
For the task of grouping closely-spaced oriented elements, Ernst et al. [2012] express the radial component as an exponential function that decays with distance. Again, we opt for a formulation that facilitates parameter tuning and express the component with a Gaussian decay:

$$A^d = \exp\left(d^2/\left(2\sigma_d^2\right)\right) \quad , \tag{17}$$

with d the distance between the two positions and σ_d a distance-spread. We set σ_d to twice the maximum $\mathbb{M}^I_+$ radius when computing good continuation for links (Section 7.2), and to the distance between the tangent origins when computing tangent origins during Y-junction interpretation (Section 7.3.2).

C STRUCTURAL ADJUSTMENTS

The schematization and smoothing procedures discussed in Section 9.2 modify stroke geometry, which can corrupt the structural relations between strokes. However, junctions provide the necessary information for rectifying these relations.

Schematization adjustment. Schematization [Dwyer et al. 2008] is applied to each stroke separately. This can corrupt the topology of stylized glyph, making it difficult to apply intersection-based adjustments to the stroke endpoints. While a correct topology could be imposed with constraint solving algorithms [Nöllenburg 2014], we observe that this issue mostly affects strokes such as the lower-left serif in Fig. 30b, which has another stroke ending within it. This configuration can be detected by counting the number of T-junctions and branching Y-junctions along a stroke. If, for a given stroke, only one such junction exists, we translate the stroke by $\boldsymbol{p}' - \boldsymbol{p}$, where $\boldsymbol{p}$ is the original endpoint of the incident stroke and $\boldsymbol{p}'$ is the endpoint after schematization (Fig. 30c). After executing this procedure, we can shift the stroke endpoints to the closest intersection of the end-tangents with the opposite stroke (Fig. 30d).

Smoothed stroke adjustment. Smoothing also can corrupt the adjacency relations between strokes. This is especially likely to occur when using a simplified or schematized spine as an input for a stylized smoothing method such as the one by Berio et al. [2017] (Fig. 30e). To adjust these configurations, we perform a first smoothing pass on each stroke. We then adjust the end-vertices of the spines used as an input to the smoothing methods, so that the non-smoothed spines are incident to the smoothed ones (Fig. 30f).

D LIST OF SYMBOLS

Main symbols

$\boldsymbol{x}$	Outline point – §4.1
f	Fork (degree-3 medial axis vertex) – §4.1
$\boldsymbol{t}$	Tangent – §4.2.2
$\boldsymbol{n}$	Inward normal at CSF – §4.2.2
$\boldsymbol{x_c}$	CSF extremum – §4.2
$\boldsymbol{y}$	Medial axis point – §4.1
$\boldsymbol{\varphi}_{ij}$	Flow of a link for the concavity pair (c_i, c_j) – §5.1
$\boldsymbol{\pi}$	Protruding direction of a branch – §5.1
p	Scalar product of $\boldsymbol{\varphi}_{ij}$ and $\boldsymbol{\pi}$ – §5.1
F	A set of forks – §6
r	Disk radius of medial axis vertex – §4.4.1
$\boldsymbol{y}_f$	Fork's position – §7.3.3
r_f	Fork disk radius – §7.3.3
d_f, d_e	Radius-weighted distances – §4.4.1
$s_{f_{\max}}$	Max geodesic length from a fork along $\mathbb{M}^I$ – §4.4.1
σ_d	Spread parameter for ψ – §5.3 & §B
b	Medial axis branch – §4.1
c	CSF – §4.2
η	Link – §5
C	A set of concave CSFs – §5.1
γ	Junction – §6
H	A set of valid links (also H_f) – §7
J_f	Set of candidate junctions of a fork – §7.3.5
B	A set of branches – §7.3.7

Objects/Structures

$\mathbb{M}^I$	Interior medial axis – §4.1
$\mathbb{M}^E$	Exterior medial axis – §4.1
$\mathbb{S}$	Stroke graph – §6.2
$\mathbb{H}$	Graph of valid links and concavities - §7
$\mathbb{Q}$	Planar map used to construct stroke areas – §8.2

Saliency/significance measures

$\beta(b, f)$	Salience of branch b protruding from fork f – §4.1.2
$\omega(\eta)$	Link salience – §5.2
ψ	Good continuation – §5.2, §5.3 & §B
Λ	Junction evaluation measure – §7.3
$w(c, f)$	Significance of a concavity with respect to a fork f – §7.3.3
Λ_{I}	Measure of coverage – §7.3 & §7.3.1
Λ_{ψ}	Measure of smoothness – §7.3 & §7.3.2
Λ_{w}	Measure of concavity significance – §7.3 & §7.3.3
Λ_{η}	Measure of link salience – §7.3

Thresholds and Tolerances

τ_{β}	Branch saliency threshold – §4.1.2
$\beta_{\min}$	Branch saliency lower bound – §7.3.5
λ_{L}	Maximum concavity radius multiplier for L-junctions – §7.3.5
r_h	Maximum CSF radius, divided by glyph height – §4.2.1