DOI: 10.1111/cgf.14901

Eurographics Symposium on Geometry Processing 2023
P. Memari and J. Solomon
(Guest Editors)

COMPUTER GR PHICS *forum*
Volume 42 (2023), Number 5
e14901 pp. 1 - 15

Singularity-Free Frame Fields for Line Drawing Vectorization

Olga Guțan†[1], Shreya Hegde‡[2], Erick Jimenez Berumen[3], Mikhail Bessmeltsev[4], Edward Chien[5]

[1]Carnegie Mellon University, [2]Amherst College, [3]California Institute of Technology,
[4]Université de Montréal, [5]Boston University

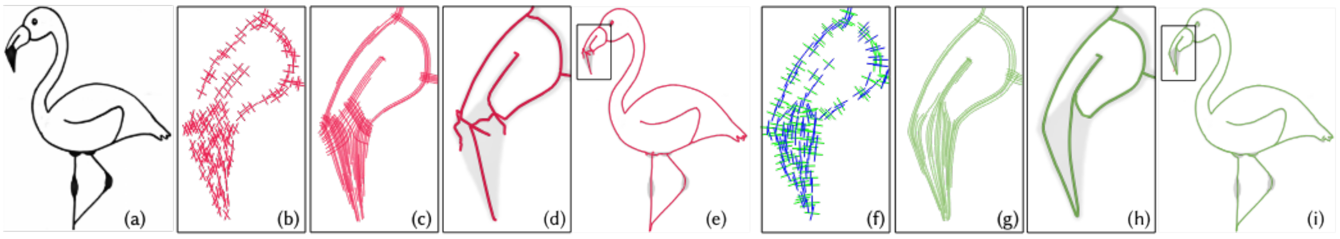


Figure 1: *For a given bitmap image (a), state-of-the-art line drawing vectorization algorithms [BS19, SBBB20, PNCB21] rely on frame fields (b) aligned with the curve directions in the image. In [BS19], for instance, those are used to trace streamlines (c), leading to the final vectorization (d,e). We develop a novel framework producing singularity-free frame fields, which can be combed globally (f), allowing for a more robust resolution of directions of intersecting curves, better streamlines (g), and ultimately, better vectorization (h,i) without changing the rest of the vectorization algorithm. Input image ©Tracie Kiernan, https://stepbysteppainting.net/.*

Abstract

State-of-the-art methods for line drawing vectorization rely on generated frame fields for robust direction disambiguation, with each of the two axes aligning to different intersecting curve tangents around junctions. However, a common source of topological error for such methods are frame field singularities. To remedy this, we introduce the first frame field optimization framework guaranteed to produce singularity-free fields aligned to a line drawing. We first perform a convex solve for a roughly-aligned orthogonal frame field (cross field), and then comb away its internal singularities with an optimal transport–based matching. The resulting topology of the field is strictly maintained with the machinery of discrete trivial connections in a final, non-convex optimization that allows non-orthogonality of the field, improving smoothness and tangent alignment. Our frame fields can serve as a drop-in replacement for frame field optimizations used in previous work, improving the quality of the final vectorizations.

CCS Concepts
• ***omputing methodologies*** → ***Parametric curve and surface models;*** *Shape analysis;*

1. Introduction

Line drawing vectorization is a foundational tool for converting hand-drawn concepts into resolution-independent, compact, and easy-to-edit curves, producing output amenable to geometry processing algorithms. Traditional line drawing vectorization algorithms fail to faithfully vectorize the geometry of junctions and sharp corners, or create spurious branches in line drawings, often making the final results unacceptable [BS19].

State-of-the-art vectorization algorithms address this challenge by using *frame fields* — a smooth assignment of two axes per pixel where at least one is always aligned to the depicted curve tangent (Fig. 1b). This helps disambiguate directions around junctions more robustly [BS19,PNCB21,SBBB20]. However, the quality of the final vectorization depends heavily on the quality of the underlying frame field. A fundamental limitation of the methods above is that their frame fields contain *singularities* — points where matching of the directions is inconsistent or one of the frame directions is zero, leading to axis switching (Fig. 2a). Singularities manifest as artifacts in the final results (Fig. 1d). Precisely, singularities can lead to inconsistent topology and geometry, such as gaps or

Corresponding author
‡ Joint first author

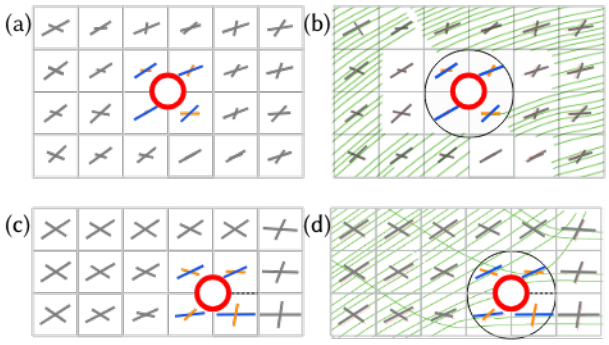


Figure 2: *Axis switch and zero singularities from polyvector fields of [BS19], and their resulting traces. (a) and (b) show an axis switch singularity where the smaller orange axis nearly vanishes and noisily switches with the blue. (c) and (d) show a zero singularity where the matching fails across the dashed line. [BS19] omits tracing through these singularities to avoid difficulties.*

spurious connections, for both tracing-based [BS19, PNCB21] and parameterization-based vectorization methods [SBBB20].

In many cases, singularities of vector fields or frame fields are topologically inevitable due to the Poincaré-Hopf theorem. We observe, however, that our domain, the set of all dark pixels in the image that we call the *narrowband*, is planar and has boundaries. Thus, any topological winding of the field may be pushed to these boundaries, resulting in singularity-free frame fields (e.g., Fig. 1f, 4d). Guided by this observation, we propose a new design and optimization for line drawing–aligned frame fields that contain no singularities, thus frequently improving final vectorization quality.

We begin with a convex optimization for a *cross field*, i.e., a frame field with orthogonal axes, that is smooth and attempts to align one of its axes with the curve tangent at each point. This field is almost certain to contain interior singularities, so we comb these singularities out of the field by matching singularities of opposite indices, or by pushing them to the narrowband boundaries. This matching is done with optimal transport in a fashion reminiscent of [SV19], albeit with a different transportation cost (Sec. 2.3), which aims to localize the field modifications and minimize the increase in field smoothness. Finally, the frame field is represented as a pair of line fields in a novel nonconvex optimization, which allows for nonorthogonality and produces robust tangent alignment at curve junctions. The absence of singularities, as well as values of boundary indices informed by the field combing, are enforced as linear constraints in the framework of discrete trivial connections [CDS10]. Furthermore, a barrier term prevents the line fields from crossing each other and developing axis switch singularities.

We validate our approach by qualitatively comparing the vectorizations and fields produced against the state-of-the-art frame field designs, using the original frame-field–based vectorization algorithm of [BS19] and the PolyVector flow of [PNCB21].

Overview. Our system takes as input a bitmap line drawing, for which we first identify interior and exterior boundaries (Sec. 3.3). We then employ a quadratic solve to compute an initial cross field, likely with singularities (Sec. 4.1). Next, we determine field indices and correct the initial cross field to have zero singularities inside the domain, thus finding a trivial connection (Sec. 4.2). The obtained indices, after a simple conversion (Sec. 4.2), are used as constraints for the nonconvex optimization (Sec. 4.3), which generates the final frame fields as pairs of nonorthogonal line fields. We validate our method on line drawing vectorization using the methods of [PNCB21] and [BS19] (Sec. 4.4, 5) with our frame fields.

2. Related Work

2.1. Line Drawing Vectorization

Vectorization of bitmap images with curves has been studied extensively in multiple disciplines. Some algorithms, e.g., for vectorizing medical imaging output or reconstructing road maps from GPS traces, rely on domain knowledge inapplicable to line drawings [BLW16, CFL13, TBA*13]. In contrast to technical drawings composed of straight lines or a small number of primitives [HT06, EVA*20], we focus on freeform line drawings containing piecewise smooth curves, which may or may not be closed.

Rough line drawings containing oversketching can be vectorized by preprocessing with blurring kernels [BCF*07, KLC07, CGBG13] or neural networks [SSISI16, SSII18, XXM*21]. These approaches only produce raster output, they do not vectorize images. In the context of vectorization, they can be used to *simplify* an oversketched bitmap drawing prior to vectorization.

Alternative approaches for line drawing vectorization include optimal transport [GCSAD11], region-based approaches [CDQM18], Delaunay-based merging of parallel strokes [PPM18], or an expensive graph-based optimization [FLB16]. In contrast to [FLB16], we focus on the task of *precise* vectorization of each drawn stroke without simplification.

Historically, methods for clean line drawing vectorization have relied on 1-skeletons or raw image gradients to extract junctions and connectivity [NS19, NHS*13, DCP17, DCP19, BLW16]. Both approaches are unreliable in the presence of noise and lead to artifacts [FLB16, PNCB21]. [DCP17] explores improving vectorization by leveraging Pearson's correlation coefficient with Gaussian kernels. This method does not reconstruct the drawing topology — instead it relies on the topology of a one-pixel-width-skeleton, an approach prone to local artifacts, as discussed by [FLB16]. To alleviate the effect of the noisy gradient, [BF12, CLMP15a, CDQM18] generate a smooth tangent direction field, which they use for tracing. While more robust to noise, a tangent field is unable to capture the full collection of directions present at a junction point, rendering methods such as [CLMP15b] reliant on user interactions to correctly solve the junctions.

More recently, researchers have leveraged deep learning machinery towards image vectorization [EVA*20, GZH*19, GTG*19, LWKF17, ZQM19, DYH*21, LHES19, CDAT20, LLMRK20, RGLM21, KWOG18, MSSG*21]. These approaches either formulate vectorization as a segmentation task [KWOG18] or predict a fixed number of curves [CDAT20, LHES19]. Other deep learning methods improve those works by designing recurrent neural networks predicting splines from images [GTG*19, RGLM21].

Unfortunately, the available training datasets lack complexity, thus rendering the algorithms capable of vectorizing only simple images. Furthermore, these works do not aim to reconstruct correct drawing topology, making the final drawings inconvenient for editing and other downstream applications [PNCB21].

We build upon the state-of-the art series of works that rely on *frame fields* [BS19, SBBB20, PNCB21]. Frame fields are a natural extension of tangent fields, specifying two directions per pixel, where at least one aligns to the curve tangent. The frame field helps disambiguate the directions of X- and T-junctions, capturing the directions of the two intersecting curves. [BS19] generates the frame field via a linear solve, then traces integral curves (streamlines) along a frame field aligned with the input drawing. These streamlines are then grouped, forming a graph that is cleaned and converted into a final vectorization. [SBBB20] starts from a similar frame field, but replaces the tracing component with a *uv*-parameterization of the narrowband, and uses the parameterization isolines. More recently, [PNCB21] uses the same frame field from [BS19], augmenting it with the extraction of curve endpoints, intersections, and sharp corners via deep learning machinery. The extracted keypoints are then connected to each other with curves aligned to the frame field via a geometric flow. These approaches fall short due to their reliance on the quality of the generated frame fields, prone to errors generated by singularities.

Our work improves the core component of the methods above: the underlying frame fields. Singularities in the frame fields lead to artifacts and incorrect vectorization topology. Our framework is guaranteed to produce frame fields with no singularities, thus improving the quality of subsequent vectorizations. We demonstrate that our method improves markedly upon the frame fields of [BS19], also used for the PolyVector flow of [PNCB21].

2.2. Cross Fields, Frame Fields and Line Fields

Cross fields, or an assignment of two orthogonal directions per point, have found many graphics uses over the past few decades, e.g., crosshatch rendering [HZ00], texture synthesis [PFH00, LH06], pattern tiling [MV21], parametrization [RLL*06], and quad meshing [BLP*13]. They may be generated in multiple ways, targeting smoothness while specifying singularities, or prescribing directional constraints [RVAL09, CDS10, PZ07]. Recent work has made use of the Ginzburg-Landau energy to automatically generate cross fields with sparse singularities [VO19]. In our application, we aim to capture junction directions more flexibly for line drawing vectorization, so we use a more recent non-orthogonal generalization: frame fields [PPTSH14, DVPSH15].

Frame fields are 4-PolyVector fields composed of two interchangeable 2-RoSy fields [DVPSH14]. Initially, frame fields have been used for anisotropic [PPTSH14] or planar quad remeshing [LXW*11], but they have since found additional uses. For example, [IBB15] aims to reconstruct 3D objects from curvature lines by generating a frame field in the space *between* the input curves.

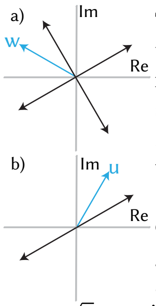
Unlike [IBB15], we follow [BS19] and compute our frame fields exclusively on the dark pixels of the narrow band, as it provides significant computational advantage. Furthermore, instead of using the PolyVector representation for frame fields [DVPSH14] featured in [BS19, PNCB21, SBBB20], we encode a frame field as a pair of line fields, allowing us to explicitly control and eliminate singularities. In the previous two paragraphs, we have cited a collection of works on vector fields and vector-field-like objects that is perhaps most related to our method, but we have undoubtedly missed many. A survey which provides additional context and applications is [VCD*16].

2.3. Singularity Matching via Optimal Transport

[SV19] is the only other work that we are aware of, which matches singularities of fields with an optimal transport framework. Their goal is the abstract one of interpolating vector fields over triangle meshes. Their method operates by interpolating parts of the Hodge decomposition of the connection 1-form, generating global interpolations of the two input fields. In contrast, our singularity-combing procedure is aimed at producing localized interpolations, and the connection 1-forms are only modified along matching paths in the pixel mesh. The ground metric directly reflects the increase in smoothness energy that would result from pushing a singularity along edges. This maintains alignment of the field to the sketch, and fixes the noisy singularities that [VO19] also aims to tame.

3. Background

3.1. Frame Fields: Representations & Singularities



To specify potential curve directions at a pixel, we use cross fields or pairs of line fields. One can parameterize the space of *cross fields* at a pixel with a single complex number w [KCPS13], termed a *representative vector*. The four *component vectors* of the cross field are the set of complex fourth roots $\sqrt[4]{w}$, see inset figure (a). Analogously, the space of *line fields* at a pixel is parameterized with a single representative vector, given by complex number u. The component vectors are the complex square roots $\sqrt{u}$, see inset figure (b). These representations are special cases of the PolyVector representation of [DVPSH14], whereby component vectors of a field are roots of a complex polynomial. The polynomials generating the fields above are $z^4 \quad w$ and $z^2 \quad u$.

For the cross and line fields above, a 2π rotation of the representative vectors w and u leads to a $\frac{\pi}{4}$ and $\frac{\pi}{2}$ rotation of the component vectors, respectively. Nonzero rotations of the representative vector fields as you circulate (counterclockwise) about a point denote *zero singularities*. These correspond to the indexed point singularities at zeros of a smooth vector field. An example of these singularities for non-orthogonal frame fields generated via the method of [BS19] may be seen in Fig. 2. Such singularities may cause errors in the downstream tracing of such fields.

In this work, the *index* of cross and line field singularities is the number of full (2π) rotations that the representative vector field performs as you circulate counterclockwise about the singularity. Note this differs from a common convention where this index is divided by 4 and 2 for cross fields and line fields, respectively, to reflect rotation of the component vectors. As we are looking at discretized vector fields, we consider points to be singularities only if they have nonzero index; see the inset figure in Sec. 3.3.

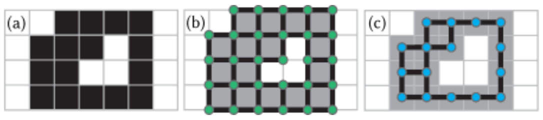


Figure 3: *Narrowband (a), primal mesh (b), and dual mesh (c).*

[BS19] uses a specific 4-polyvector space to parameterize two sets of axes, given by complex polynomials of the form $z^4 + c_2 z^2 + c_0$. These factor into products z^2 $w)$ z^2 $u)$ showing the representative vectors of the two axes. In this representation it is possible to have the w and u axes switch or pass through each other, as illustrated in Fig. 2a. This is an *axis switch singularity*, and it can also cause serious errors in downstream tracing (Fig. 2b) of these fields. In Sec. 4.3, we describe a particular frame field representation and optimization objective that avoids these singularities.

3.2. Domain & Discrete Differential Operators

The domain of our algorithm is the *narrowband* of the image, denoted Ω: the union of pixels with greyscale value above a user-specified threshold $\tau \in [0, 255)$. We form a *primal mesh* and a *dual mesh* of the narrowband, illustrated in Fig. 3. The primal mesh (Fig. 3b) has quadrilateral faces F, which are the narrowband pixels, with edges E and vertices V representing the boundaries and corners of these pixels. The dual mesh (Fig. 3c) elements are in correspondence with various primal mesh structures. Its quadrilateral faces are centered at interior primal vertices V_{int}: those that are a corner of four narrowband pixels. Each dual edge crosses a primal interior edge: those separating two narrowband pixels; the set of which are denoted E_{int}. Lastly, the dual vertices are centered at corresponding primal faces: narrowband pixels.

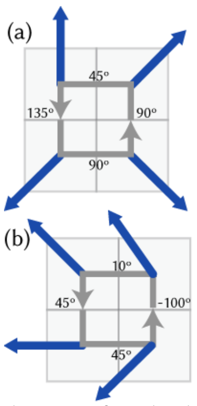
The narrowband may be partitioned into *connected components* of the dual mesh (also called *4-connected*). For simplicity, most of this paper assumes that Ω has just one connected component. This is often not the case, but the appropriate generalizations to multiple components are described in the relevant spots. Unless otherwise noted, let m denote the number of connected components, and let B denote the set of boundary components of the dual mesh, with $|B| = n$.

To constrain singularities, we use the machinery of discrete exterior calculus [CdGDS13]. The discrete gradient operator, $d_0 \in \mathbb{R}^{|E| \times |V|}$, is a sparse matrix with ± 1 entries in positions $i, j)$, where vertex j is an endpoint of edge i. The sign of the entry depends on an arbitrary reference direction chosen for each primal edge: $+1$ if j is the target, and 1 if j is the source. When d_0 is applied to a function over V, it finds the function difference over each oriented edge, giving a discretized gradient of that function.

The discrete curl operator $d_1 \in \mathbb{R}^{|F| \times |E|}$ is a sparse matrix with ± 1 entries in positions $i, j)$, where edge j is on the boundary of face i. The entry is $+1$ if edge j is directed counterclockwise about face i, and 1 if it is clockwise. When d_1 is applied to a discretized vector field (or 1-form) given as a function over oriented edges, it calculates a discretized curl about primal mesh faces.

3.3. Trivial Connections

We specialize the method of [CDS10] to our planar pixel grid setting, considering (primal) face-based fields over Ω. These fields are specified by rotations to be applied (discrete connections) as one traverses the edges of the dual mesh. Thus a connection 1-form is given by a vector $\alpha \in \mathbb{R}^{|E_{\text{int}}|}$. Topological constraints like singularities, their indices, and boundary indices become linear conditions that α must satisfy.



For locally well-defined fields, we ask that the sum of angle changes is a multiple of 2π when summed around any four dual edges that circulate about a primal interior vertex and corresponding dual face. Examples are in the inset figure: top – sum is 2π giving index 1; bottom – sum is 0 giving index 0. This is a discrete analogue to a connection having trivial holonomy about a local loop. As our domain Ω is flat, we do not need to account for any Gaussian curvature contained within our cycle about a vertex, as done in [CDS10].

To avoid internal singularities, we ask that the sum of angle changes is zero for every such loop. We express this with a $|E_{\text{int}}| \times |V_{\text{int}}|$ submatrix d_0^{int} of d_0, whose rows/columns correspond to interior primal edges/vertices (respectively). Multiplication by $d_0^{\text{int}})^\top$ sums angle changes about interior primal vertices, and $d_0^{\text{int}})^\top \alpha = 0$ enforces no internal singularities.

For globally well-defined fields, the sum of angle changes along any loop of dual edges (called *cycles* by [CDS10]) must be equal to $2\pi k$, $k \in \mathbb{Z}$. As this is already enforced about local cycles, we need only enforce it further on a set of homology generators for Ω. With a planar narrowband, the boundary cycles of the dual mesh may be chosen as a set of generators. Let $\partial\Omega)_i$ denote the i^{th} boundary component, and let k_i denote the integer number of rotations induced by the connection about $\partial\Omega)_i$. The set of k_i's is topologically constrained by a discrete Poincaré-Hopf formula (Eq. 3). If the sum of angle changes is a multiple of 2π for every loop of dual edges, the connection is called *trivial*.

Connection Integration. Given a trivial connection on a connected component of Ω, there is one more degree of freedom in specifying a field: the field angle ϕ at some root pixel (dual vertex). From there, one integrates the connection along a spanning tree in the dual mesh to get a full vector field. In particular, let θ_i denote the representative vector angle at a dual vertex v_i, and let γ_i denote the unique path to v_i from the root of the spanning tree. Then an expression for θ_i is:

$$\theta_i = \phi + \sum_{ij \in \gamma_i} \alpha_{ij} \tag{1}$$

In the expression above, $\alpha_{ij} =$ α_{ji}, so the orientation of the edge is accounted for. In general, a narrowband domain consists of many connected components, so there is an additional degree of freedom ϕ_j for each of these. They denote the field angle at each root pixel in a spanning forest of the narrowband dual mesh.

Boundary Indices & Poincar -Hopf. For a given oriented boundary $\partial\Omega)_i$, we have that $k_i = \frac{1}{2\pi} \sum_{ij \in \; \partial\Omega)_i} \alpha_{ij}$. The direction of

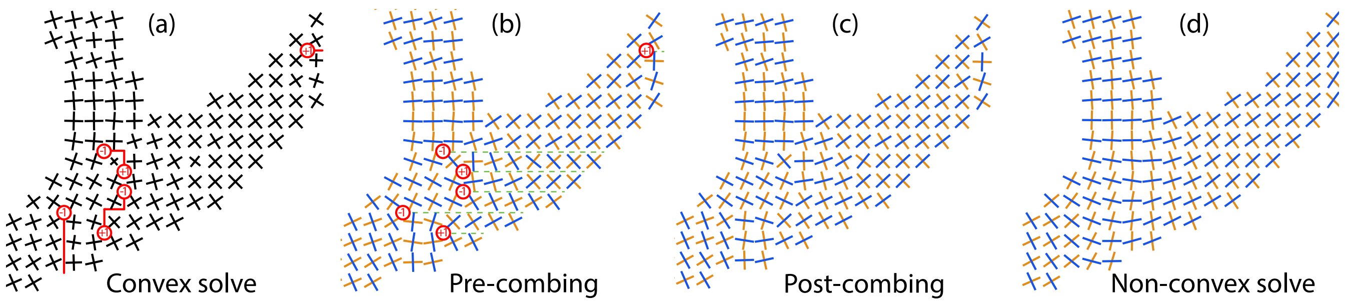


Figure 4: *Field Optimization Overview. The first step of our field optimization generates a cross field (a) which aligns well with the tangent in most places, but has singularities and fails near junctions and curve endpoints. Due to these singularities, one cannot extract a singularity-free line field directly, as is seen in (b). Integrated field discontinuities are denoted with dashed green lines. Our optimal-transport-based matching pairs singularities of opposite signs or pushes them to the boundary, e.g. (a), and the result post-combing is seen in (c). The final nonconvex solve allows our fields to be non-orthogonal achieving a smooth, singularity-free, and aligned field.*

boundary traversal follows convention, keeping the interior of Ω to the left (thus, counterclockwise on exterior boundaries and clockwise on interior boundaries). Boundary indices are defined:

$$\mathrm{ind}_\alpha \quad \partial\Omega)_i) = \begin{cases} 1 \quad k_i & \text{if } \gamma \text{ is an exterior boundary} \\ \quad 1 \quad k_i & \text{if } \gamma \text{ is an interior boundary} \end{cases} \tag{2}$$

Notably, the n boundary indices for a connected component of Ω must satisfy the Poincaré-Hopf formula:

$$\sum_{i=1}^{n} \mathrm{ind}_\alpha \quad \partial\Omega)_i) = \chi \quad \Omega) = 2 \quad n \tag{3}$$

where χ denotes the Euler characteristic. Note that this holds over all connected components of our narrowband. Intuitively, the boundary indices denote the number and kind of corners and sharp turns implied by the frame field. Some simple examples are in the supplementary materials, for readers unfamiliar with this concept.

Desired boundary indices may also be enforced with a linear constraint. We construct a matrix $H \in \mathbb{R}^{n\times|E_{\mathrm{int}}|}$ where each row corresponds to a boundary cycle $\quad \partial\Omega)_i$:

$$H \quad j,k) = \begin{cases} 1 & \text{if } jk \in \quad \partial\Omega)_i \text{ and } jk \text{ is with traversal direction} \\ \quad 1 & \text{if } jk \in \quad \partial\Omega)_i \text{ and } jk \text{ is against traversal direction} \\ 0 & \text{if } jk \notin \quad \partial\Omega)_i \end{cases} \tag{4}$$

If we gather the desired k_i into a vector $\mathbf{k}$, the boundary index constraints are succinctly expressed as $H\alpha = \mathbf{k}$.

4. Method

Our method follows a sequence of three steps: an initial quadratic solve for a roughly-aligned cross field, an identification and combing of the singularities, and a last nonconvex solve to obtain a smoother, better-aligned non-orthogonal field. An overview of how this procedure operates near a junction is shown in Fig. 4.

4.1. Convex Initialization

Our main optimization (Sec. 4.3) is nonconvex, hence its final quality depends on the initial guess. In this section, we use a linear solve to optimize a quadratic energy, and produce a cross field that is aligned with the drawing, but likely has singularities.

A complex vector $f \in \mathbb{C}^{|F|}$ represents a cross field on Ω, where f_i is the representative vector on pixel i (see Sec. 3.1). We formulate the following objective, similar to [KCPS13]:

$$\min_{f\in\mathbb{C}^{|F|}} \left(\frac{1}{\quad_S} f^\top \quad \bar{f} + \frac{w}{\quad} \sum_{i:g_i\neq 0} |f_i \quad g_i^4|^2 \right), \tag{5}$$

where $\quad = d_1^{\mathrm{int}} \; d_1^{\mathrm{int}})^\top$ is a discrete Laplacian for functions over the pixels, and $g_i \in \mathbb{C}$ is the normalized image gradient at pixel i. Here, d_1^{int} is the $|F| \times |E_{\mathrm{int}}|$ submatrix of d_1, with columns corresponding to the interior edges of the primal mesh (and edges of the dual mesh). The first term is a Dirichlet energy, measuring field smoothness, and the second term expresses alignment to the image gradient. We used $w = 1/8$ in our experiments, which is a relative weighting parameter between these two terms. The two normalization factors, $\quad_S$ and $\quad$, are respectively the number of dual edges and the number of pixels with non-zero image gradient.

The resulting linear KKT system is:

$$\left(\frac{1}{\quad_S} \quad + \frac{w}{\quad} P \right) f = \frac{w}{\quad} P g^4, \tag{6}$$

where P is a diagonal matrix selecting pixels with $g_i \neq 0$. The system matrix is sparse, symmetric, and positive semidefinite, so we solve the system via preconditioned conjugate gradients. An example optimum may be seen in Fig. 4. The result tends to aligns well in most regions, but there may be many internal singularities, often arising at junctions and curve endpoints.

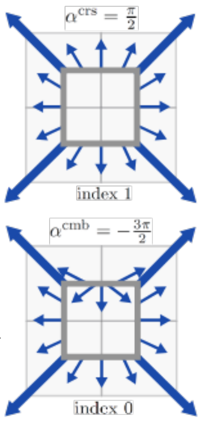
4.2. Field Combing for Index Extraction and Assignment

We ultimately represent our frame fields as pairs of line fields, and require them to satisfy particular index constraints to avoid internal singularities. In this step, we use the cross field to find one of these line fields, and we "comb" away the singularities to determine the boundary index constraints we will impose.

Having computed the representative vector field f of the optimal cross field, we compute the connection induced by this representative vector field: $\alpha_{ij}^{\mathrm{crs}} = \mathrm{Arg}\left(\frac{f_j}{f_i}\right)$. A naive attempt at finding the connection for a non-singular line field would be to consider $\alpha_{ij}^{\mathrm{crs}}/2$, as the representative vector for an axis of the cross field rotates at half the rate of the representative vector for the cross field. Unfortunately, this fails to represent a trivial connection for a line field as can be seen in Fig. 4b.

Failure occurs at internal singularities of the cross field, which may be found by computing $\ \ d_0^{\mathrm{int}})^\top \alpha^{\mathrm{crs}} \in \{\ \ \ 1, 0, +1\}^{|V_{\mathrm{int}}|}$. Nonzero elements denote interior singularities of those indices. Our singularity indices are bounded by 1 in absolute value because our dual faces have four edges and $|\alpha_{ij}^{\mathrm{crs}}| < \pi$.

To eliminate these singularities, we perform an optimal-transport-based matching of the positive and negative singularities with each other, or with the boundaries. We then push these singularities along their matching paths to eliminate each other or contribute to a particular boundary index. Before getting into problem formulation details, we note that we provide a few local examples of this matching in the supplementary for intuition.



$\alpha^{\mathrm{crs}} = \frac{\pi}{2}$
index 1
$\alpha^{\mathrm{cmb}} = -\frac{3\pi}{2}$
index 0

Pushing singularities. The initial cross field is often well-aligned along the boundaries, so we aim to keep the individual crosses the same at each pixel. We achieve this while still shifting singularities by modifying α^{crs} by increments of 2π as we cross each dual edge. This may be seen in the inset figure, where one can see that this is equivalent to reversing the direction of interpolation of the representative vectors. In particular, an index 1 singularity is pushed upwards, subtracting 2π from the original connection, and reversing the direction of interpolation. We use α^{cmb} to denote the new connection and when we push a singularity over an edge ij the appropriate change is:

$$\alpha_{ij}^{\mathrm{cmb}} = \begin{cases} \alpha^{\mathrm{crs}} \mp 2\pi & \text{if } ij \text{ is with edge orientation} \\ \alpha^{\mathrm{crs}} \pm 2\pi & \text{if } ij \text{ is against edge orientation} \end{cases} \tag{7}$$

In the above expression, the top/bottom operands correspond to pushing of a positive/negative singularity, respectively.

These changes could lead to large increases in $\|\alpha^{\mathrm{crs}}\|^2$, so we define two weighted directed graphs that specify the metric for our singularity matching problem. They motivate smaller increases in smoothness as we push singularities about Ω, and also incentivize movement of singularities through regions where $|f|$ is small. We need a directed graph each for pushing positive and negative singularities, with weights denoted w_{ij}^{+} and w_{ij}, respectively. The vertices and edges are those of the primal mesh.

In the push of a positive singularity depicted in the inset figure above, the increase in smoothness is $\ \ \alpha^{\mathrm{crs}} \ \ \ 2\pi)^2 \ \ \ \ \alpha^{\mathrm{crs}})^2 = 4\pi\ \pi\ \ \ \alpha^{\mathrm{crs}})$. Similar calculations motivate the following weights for our graphs:

$$w_{ij}^{\pm} = \begin{cases} \ \ |f_i| + |f_j|)\ \ \pi \mp \alpha^{\mathrm{crs}}) & \text{if } ij \text{ interior, with orientation} \\ \ \ |f_i| + |f_j|)\ \ \pi \pm \alpha^{\mathrm{crs}}) & \text{if } ij \text{ interior, against orientation} \\ \infty & \text{if } ij \text{ boundary} \end{cases}$$

The norm-based coefficients $\ \ |f_i| + |f_j|)$ motivate singularity movement along paths where $|f|$ is small. These weights are all positive as $\alpha^{\mathrm{crs}} \in \ \ \ \ \pi, \pi)$, so we use Dijkstra's algorithm to calculate distances on these weighted graphs.

Optimal transport matching. Let us define our optimal transport problem, with $\mathcal{S}\ \ \ ,\mathcal{S}^+ \subset V_{\mathrm{int}}$ denoting the sets of positive and negative singularities, respectively. Let $p_i\ \ $ and p_j^+ denote the i^{th} negative singularity and the j^{th} positive singularity, respectively. Then we consider a cost matrix $C \in \mathbb{R}^{\ \ |\mathcal{S}\ \ \ |+1)\times\ \ |\mathcal{S}^+|+1)}$:

$$C\ \ i,j) = \begin{cases} d\ \ \ \ p_i\ \ ,p_j^+) & \text{if } 1 \le i \le |\mathcal{S}\ \ \ | \text{ and } 1 \le j \le |\mathcal{S}^+| \\ d\ \ \ \ p_i\ \ ,\partial\Omega) & \text{if } 1 \le i \le |\mathcal{S}\ \ \ | \text{ and } j = |\mathcal{S}^+| + 1 \\ d^+\ \ p_j^+, \partial\Omega) & \text{if } i = |\mathcal{S}\ \ \ | + 1 \text{ and } 1 \le j \le |\mathcal{S}^+| \\ \infty & \text{if } i = |\mathcal{S}\ \ \ | + 1 \text{ and } j = |\mathcal{S}^+| + 1 \end{cases}$$

In the above, $d\ \ \ \ \cdot,\cdot)$ and $d^+\ \ \cdot,\cdot)$ denote distances on the +1 and -1 directed graphs. The resulting linear program solves for a matrix $T \in \mathbb{R}^{\ \ |\mathcal{S}\ \ \ |+1)\times\ \ |\mathcal{S}^+|+1)}$ as follows:

$$\begin{aligned} &\min_T \langle C, T\rangle \\ &s\ t\ \sum_j T_{ij} = 1, \text{ for } 1 \le i \le |\mathcal{S}\ \ \ | \\ &\qquad \sum_i T_{ij} = 1, \text{ for } 1 \le j \le |\mathcal{S}^+| \\ &\qquad T_{ij} \ge 0, \forall i,j \end{aligned} \tag{8}$$

The narrowband boundary serves as an infinite source and sink which turns this into an unbalanced optimal transport problem, with a binary solution. It matches each singularity with one of the opposite sign or to a nearest boundary vertex. A simple reformulation and argument shows that the resulting transport is going to be a binary matching of negative singularities and positive singularities, with some being pushed to boundaries. See the appendix for a proof of the proposition below.

Proposition 1 The optimization problem (8) has a binary solution $T_{ij} \in \{0,1\}^{\ \ |\mathcal{S}\ \ \ |+1)\times\ \ |\mathcal{S}^+|+1)}$.

After the matches are found, we push the singularities along their shortest paths, modifying α^{crs} into α^{cmb} as specified in Eq. 7. This eliminates interior singularities and adds contributions to the boundary indices. In particular, when a singularity is pushed to the boundary, its index is added to the boundary index of α^{crs}.

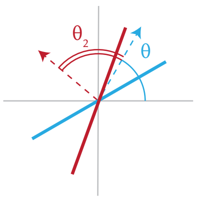
Next, we divide α^{cmb} by 2 in order to obtain a singularity-free line field to start the nonconvex optimization (Sec. 4.3). Let us denote this connection $\alpha_0 = \frac{\alpha^{\mathrm{cmb}}}{2}$. Lastly, for each boundary component $\ \ \partial\Omega)_i$, we convert the cross field index $\mathrm{ind}_{\alpha^{\mathrm{cmb}}}\ \ \ \partial\Omega)_i)$ into the line field index $\mathrm{ind}_{\alpha_0}\ \ \ \partial\Omega)_i)$ with the following formula:

$$\mathrm{ind}_{\alpha_0}\ \ \ \partial\Omega)_i) = \begin{cases} \frac{1}{2}\ \mathrm{ind}_{\alpha^{\mathrm{cmb}}}\ \ \ \partial\Omega)_i) + 1), & \text{if } \ \ \partial\Omega)_i \text{ exterior} \\ \frac{1}{2}\ \mathrm{ind}_{\alpha^{\mathrm{cmb}}}\ \ \ \partial\Omega)_i)\ \ \ \ 1), & \text{if } \ \ \partial\Omega)_i \text{ interior} \end{cases} \tag{9}$$

4.3. Nonconvex frame field optimization

Our final step is the nonconvex optimization that uses the constraints computed in the previous stages to avoid singularities and starts from a feasible initial point computed in Sec. 4.2.

Frame field representation. To allow for non-orthogonal frame fields, we represent them with a pair of unit-norm line fields. As discussed in Sec. 3.1, these may be represented by unit-norm representative vectors, illustrated with dashed lines in the inset. We choose an angle-based representation, with θ_1 denoting the (complex) argument of the first representative vector (blue), and $\theta_1 + \theta_2$ denoting the argument of the second representative vector (red).



For our optimization, we encode the frame field as three unknowns: (1) $\phi \in \mathbb{R}^m$, i.e. one value per connected component of the image, (2) $\alpha \in \mathbb{R}^{|E_{\text{int}}|}$, i.e. one value per dual edge, and (3) $\theta_2 \in \mathbb{R}^{|F|}$, i.e. one value per pixel. As described in Sec. 3.3, ϕ encodes absolute rotation of the root (for one of the line fields) for each component and α stores the connection angles per dual edge. θ_2 are the relative angles between the representative vectors as in the inset, and θ_1 can be computed via integration of α over the tree, as in Eq. 1. Note that for each root, $\theta_1 = \phi$.

Linear index constraints. We now define the linear constraints using the formalism of Sec. 3.3 that make our frame fields singularity-free. Precisely, we require that (1) sum of angle changes along a four-pixel loop around each interior primal vertex (i.e. pixel corner) should be zero, and (2) along each boundary, the line field should have the indices computed in Eq. 9. We only enforce those constraints on the line field represented by θ_1; the objective function below controls the behavior of the other line field.

We gather these constraints in the standard form as $\;x = b$, where $\in \mathbb{R}^{|V_{\text{int}}|+n\times|E_{\text{int}}|}$ and $b \in \mathbb{R}^{|V_{\text{int}}|+n}$ defined as

$$= \begin{bmatrix} d_0^{\text{int}} \\ H \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ \mathbf{k} \end{bmatrix} \tag{10}$$

where H is defined in Eq. 4, and $\mathbf{k}$ is defined in Sec. 3.3 with the indices from Eq. 9.

Objective function. Within this constrained space, we are looking for a smooth frame field aligned to the input drawing. Our objective function is a sum of three terms, controlling alignment to the line drawing, smoothness, and a barrier term preventing the two line fields from passing through each other and switching:

$$f\;\phi,\alpha,\theta_2) = \frac{1}{\;}\mathrm{E}_{\text{Alignment}} + \frac{w_B}{B}\mathrm{E}_{\text{Barrier}} + \frac{w_S}{S}\mathrm{E}_{\text{Smoothness}}, \tag{11}$$

where $w_B = 0\;5, w_S = 5$ for all our experiments.

ALIGNMENT. To express alignment to the line drawing, we aim to have at every pixel at least one of the line fields parallel to the local tangent, or, equivalently, perpendicular to the normalized image gradient g represented as a complex number. The second summand term below expresses this condition as a product of non-negative values. The first/second terms in the product vanish if the first/second line fields are perpendicular to the image gradient, respectively. A graph of this term may be seen in blue in Fig. 5.

$$\mathrm{E}_{\text{Alignment}} = \sum_{i:g_i\neq 0}\Big[\nu\mathrm{E}_\nu + \big[\cos\;\;\theta_1)_i\;\;\arg\;g_i^2)) + 1\big] \times \big[\cos\;\;\theta_1)_i + \;\theta_2)_i\;\;\arg\;g_i^2)) + 1\big]\Big] \tag{12}$$

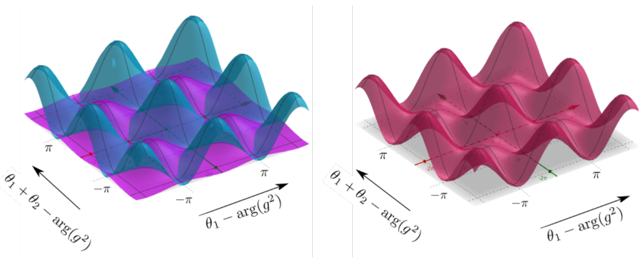


Figure 5: *Regularizing effect of E_ν. The left graph shows the summand terms for a pixel in Eq. (12), with the product term in blue (taller peaks) and E_ν in magenta (lower peaks, with an exaggerated $\nu = 0\;4$ weight). The right graph shows the resulting sum $E\;_{\text{lignment}}$ in red, with clearly non-degenerate minima. Graphs created with the GeoGebra graphing application [HB *23].*

The first summand term E_ν helps to regularize the alignment term by ensuring that local minima are nondegenerate, and pushes the line fields away from being aligned with each other.

$$\mathrm{E}_\nu = \sum_{i:g_i\neq 0}[\;\;\cos\;\;\theta_1)_i\;\;g_i^2) + 1][\;\;\cos\;\;\theta_1)_i + \;\theta_2)_i\;\;g_{\theta_i}^2) + 1] \tag{13}$$

The term introduces periodic peaks around function minima where the line fields are aligned, and ν is a constant to scale the strength of this term. A graph of E_ν may be seen in Fig. 5 in magenta. We used $\nu = 0\;05$ for all our experiments.

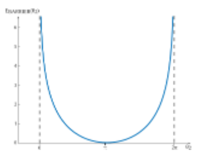
SMOOTHNESS. We express smoothness of the frame field as two Dirichlet energy terms, one for each line field:

$$\mathrm{E}_{\text{Smoothness}} = \alpha^\top\alpha + \beta^\top\beta, \tag{14}$$

where β is the change in the second line field over each dual edge. As the absolute angle of the second line field is $\theta_1 + \theta_2$, and α measures the change in θ_1, for each dual edge $\;i,j)$ we define

$$\beta_{ij} = \;\theta_2)_j\;\;\;\theta_2)_i + \alpha_{ij} \tag{15}$$

BARRIER. Finally, we use the standard logarithmic barrier (inset figure) to prevent the two axes from crossing each other and generating axis switch singularities (see Sec. 3.1):



$$\mathrm{E}_{\text{Barrier}} = \sum_{i:g_i\neq 0}\;\;\log\;\theta_2)_i\;\;\log\;2\pi\;\;\;\theta_2)_i) \tag{16}$$

With the axes unable to switch, the index constraints on the first line field are automatically imposed on the second line field.

Optimization. We use Newton's method with the linear constraints, continuing until the gradient norm of the objective function falls under $10\;^6$. Our starting point is $\;\phi_0,\alpha_0,\pi\;_{|F|})$ where α_0 was computed in Sec. 4.2 and $\theta_2 = \pi$ for each pixel. ϕ_0 at a root pixel p of the spanning forest is Arg $\;f_p)/2$, so that we start with the same underlying crosses as those obtained in the convex solve, as seen in Fig. 4c.

4.4. Tangent selection/Interface with rest of pipeline

The vectorization methods we used [BS19, PNCB21] also require per-pixel selection of one of the axes to be the curve tangent, as the pipeline begins by tracing the frame field along the tangent directions. To this end, we first measure the per-pixel alignment of each axis, comparing it with the axis perpendicular to the image gradient, represented by g_i^2:

$$\rho_1 = |e^{i\theta_1} + g_i^2|^2 \qquad \rho_2 = |e^{i\ \theta_1+\theta_2)} + g_i^2|^2 \tag{17}$$

However, certain pixels have $g_i = 0$, e.g., in the middle of a stroke when pixels are uniformly black, and others might contain local noise affecting the image gradient and the local tangent direction. To alleviate these effects, we perform Laplacian smoothing of ρ_1, ρ_2 by minimizing the following quadratic energy:

$$\min_{\rho_i'} \frac{1}{2}\lambda\rho_i'^T \ \ \rho_i' + \frac{1}{2}\ \rho_i' \ \ \rho_i)^T P\ \rho_i' \ \ \rho_i), \quad i = 1,2$$

In the above, is the Laplacian operator and P is the diagonal matrix identifying pixels with $g_i \neq 0$, both used in Sec. 4.1. In our experiments, $\lambda = 5$. Finally, we define the tangent axis to be the one with the smaller ρ_i' per pixel.

5. Results and Validation

We use our singularity-free frame fields as a drop-in replacement for the frame fields used originally by [BS19] and [PNCB21], while maintaining the rest of their pipelines. Several comparisons of the vectorizations obtained with our frame fields and those obtained with the original frame fields are in Figures 6-8. Our results highlight clear improvements in the final vectorizations, but may still contain artifacts and errors due to the tracing pipelines of [BS19] and [PNCB21]. For this reason, we include full frame field comparisons in the supplementary materials, which demonstrate our superior fields.

Since there is no perceptually aware metric available — which would be the only true quantitative measure of visual vectorization quality [PNCB21], we do not run the benchmark of [YVG20] on our results. This benchmark uses purely geometric measurements like Chamfer distance. In the supplementary materials, we show that the Chamfer distances between our vectorizations and those of [BS19] or [PNCB21] are very small ($<0\ 3$ pixels) despite clear visual differences. This brings into question the utility of such metrics for detecting geometrically small, but perceptually meaningful differences in vectorizations.

We speculate that a useful metric of vectorization quality would take into account both perceptually important differences, e.g., in tangents, topology, and curvature, as well as usability of vectorizations. Usability is clearly tied to the number and connectivity of curves. The challenging problem of quantifying and making precise this group of criteria is one that we leave to future work.

In [SBBB20], each singularity adds a cut in the parameterization, increasing the number of integer variables and runtime. Their performance, and resultant tracing quality might also benefit from our frame fields. Since they use the same frame fields as [BS19] and [PNCB21], we compare our method against these instead, as the first frame-field–based vectorization method and the state of the art, respectively.

5.1. Comparison to [BS19]

As our frame fields contain no singularities, tracing produces continuous streamlines throughout the entire domain, instead of interrupted streamlines in [BS19] (Fig. 2b). Furthermore, their method relies on heuristics to alleviate the effect of singularities, which we no longer need.

In particular, [BS19] eliminates the singular points by relaxing their alignment term in those areas, estimating that a total of 1% of the dark pixels in the narrowband are affected. This procedure successively reduces the number of singularities from hundreds to tens, typically. Final singularity counts for [BS19] are given for all displayed results in Table 1. This methodology discards information about the image, often leading to imprecise alignment or incorrect topology (Fig. 2b, 2d, 6, 7). We solve this issue by generating a higher-quality field, directly improving the streamlines and resultant vectorization. Full streamline comparisons for our method and [BS19] are in the supplementary material.

Using our frame fields in place of the original frame fields as input to the tracing method of [BS19] results in fewer spurious vectorization branches, as shown in both Figs. 6 and 7. Our higher-quality frame fields are also reflected in improved junction tracing, e.g., in Fig. 6 (donkey, human). Here, the junctions and zigzags produced by our method yield vectorizations closer to the input image. This is in contrast to the method of [BS19], which initially produces a lower-quality field, yielding slightly inaccurate topology, propagating the field error down the pipeline, and ultimately negatively affecting the quality of these regions.

5.2. Comparison to [PNCB21]

The state-of-the-art line drawing vectorization algorithm of [PNCB21] relies on frame fields both for the topology inference, when finding paths between predicted keypoints, and for the curve geometry in their polyvector flow. When run with our frame fields, their method produces vectorizations which are topologically more precise at junctions, thick strokes, and fills, as compared to their original frame fields. In particular, our frame fields improve the resolution of Y-junctions (Fig. 8, fish) and correctly resolve ambiguous connectivity (Fig. 8, cup stem).

5.3. Parameters

Here we discuss the relative effect of the parameters in the convex and nonconvex optimizations (Sec. 4.1, 4.3). We found that the w weight measuring the relative importance of alignment versus smoothness in the convex solve (high w means more emphasis on alignment) had the most impact on the final results. In particular, w controls the transition threshold between a T-junction and a Y-junction, as can be seen in Fig. 9. A T-junction is one where the intersecting curve tangents align to different field axes, and a Y-junction is one where they align to just one field axis. In our experiments, we used $w = 0\ 125$.

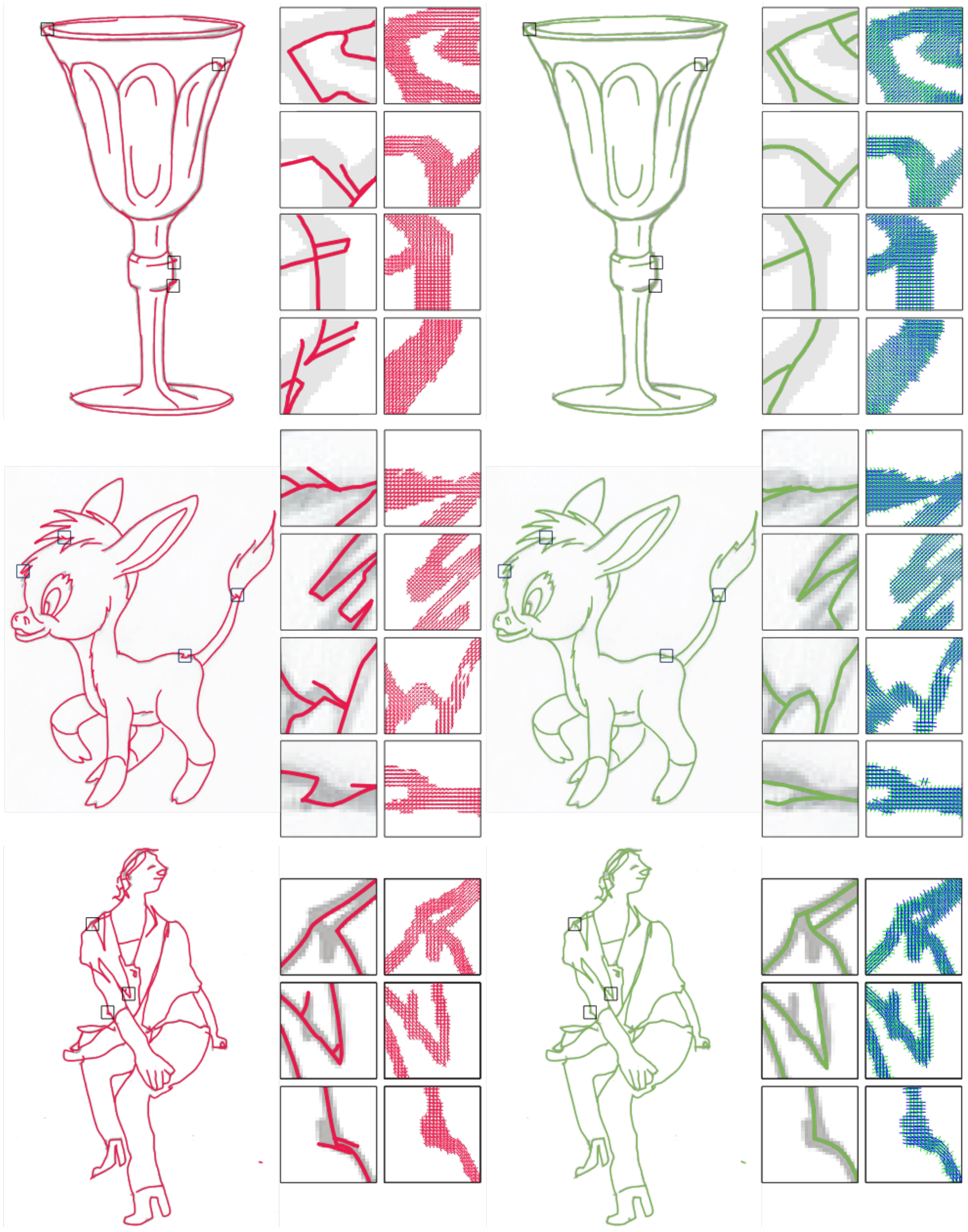


Figure 6: *Compared to the original results of [BS19] (left, red), the superior quality of the frame fields improves the results of their tracing method (green curves, right). Input images: donkey from https://www.easy-drawings-and-sketches.com/ ©Ivan Huska, sitting gesture from Sketch2Pose dataset [BB22]*

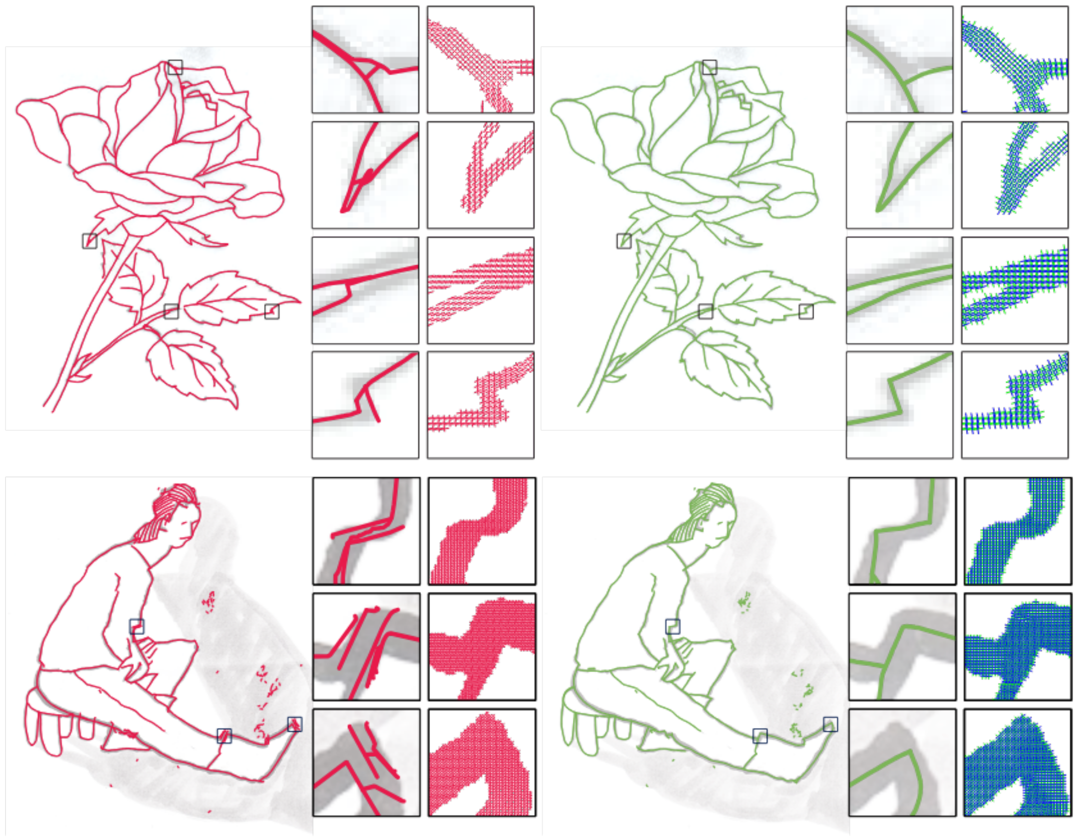


Figure 7: *The vectorization method of [BS19] (red, left) vs our method (green, right). The singularities and other imperfections cause artifacts in the final vectorizations of [BS19]. Our results do not suffer from these issues. Input images: rose from https://www.easy-drawings-and-sketches.com/ ©Ivan Huska, sitting pose from Sketch2Pose dataset [BB22], ©Olga Posukh.*

With respect to the w_B and w_S weights for the nonconvex optimization, we found them to have less pronounced influence. In particular, it seems like the starting point of the nonconvex optimization obtained via our convex solve and subsequent combing results in similar end results even when varying w_B and w_S. Lastly, we adjust the alignment term with the coefficient ν to ensure better-conditioned local minima, increasing rate of convergence and offering more stable alignment. In all of our experiments, we used $w_B = 0\ 5, w_S = 5, \nu = 0\ 05$.

Nonconvex ablation study. For completeness, we illustrate the necessity of each term in our nonconvex objective by performing an ablation study in Fig. 12. We do not omit the barrier term in our study as it clearly allows for axis switch singularities (Fig. 2a, 2b) in some instances.

5.4. Robustness

Our method is robust to various changes in image resolution, input image topology, and input stroke width, as well as density. This is partially due to the normalization terms within the optimization objectives (Eq. (5), (11)). This is demonstrated across our example inputs (Figs. 6-8), where, e.g., the stroke widths range from 1 pixel to 19 pixels (parts of the pig, see supplementary materials). We also have explicit tests of robustness to resolution (Fig. 11) and overlaid Gaussian noise (Fig. 10).

5.5. Processing Time

We ran our experiments on a Lenovo IdeaPad 3i laptop, with an Intel Core i3-1215U Processor with 8 GB of RAM. The initial convex solve, the linear solve for the optimal transport (OT) matching problem (Sec. 4.2), and the analytical computations of the Hessians and the gradients have been computed in Matlab. Then, we implemented the nonlinear optimization (Sec. 4.3) in C++ using the IPOPT library. Table 1 presents the run times in seconds, where *Opt. Time* represents the time of the convex optimization, the OT matching problem, and the nonconvex optimization time. *Vect. Time* (vectorization time) represents the time (in seconds) from when the frame field is entered into the vectorization pipeline and until the vectorization algorithm produces an SVG file with the tracing. Lastly, we also report for each input the number of singularities that result from [BS19]'s method. Our output is singularity-free.

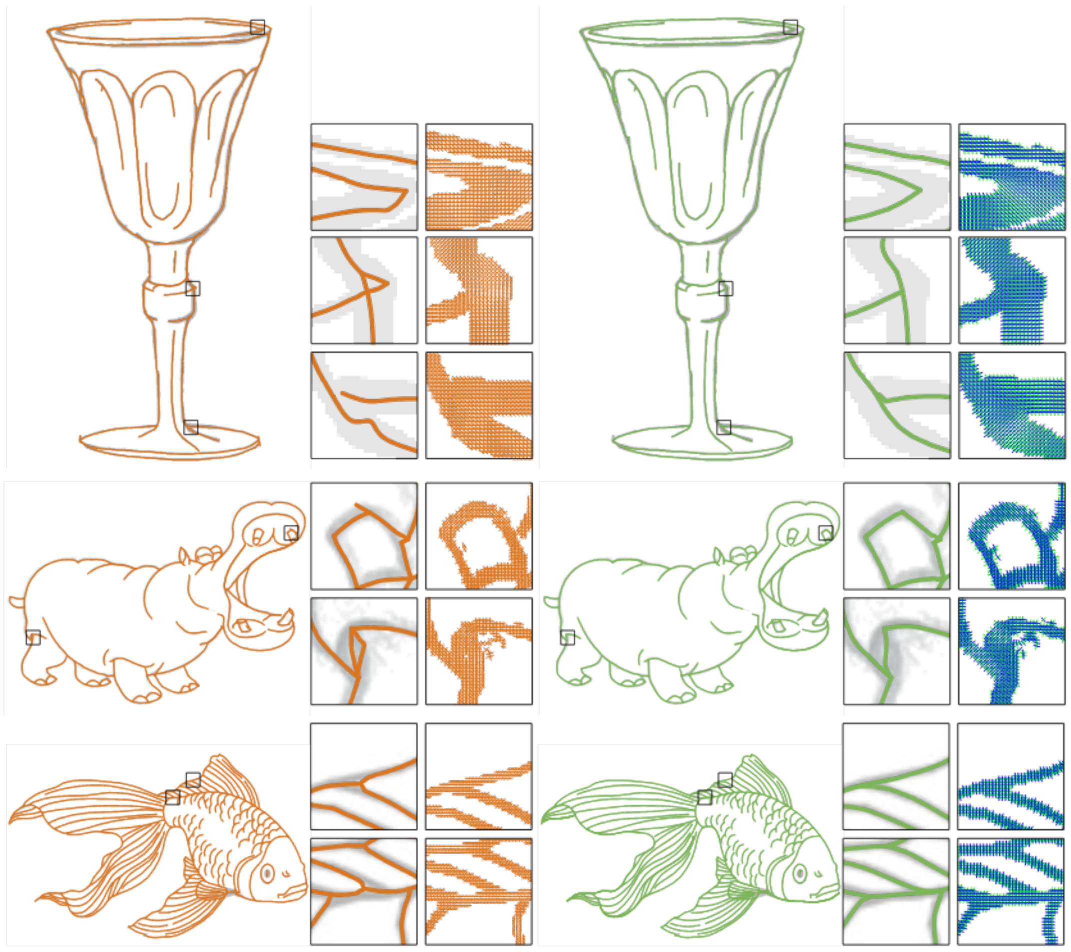


Figure 8: *Compared with the results of Polyvector Flow by [PNCB21] with their original frame fields (left, orange/brown), using our frame field design with their tracing method can substantially improve the final result (right, green). Since we kept the tracing method as is, all the improvements are solely due to the absence of singularities and higher overall quality of the frame field. Inputs from https://www.easy-drawings-and-sketches.com/ ©Ivan Huska.*

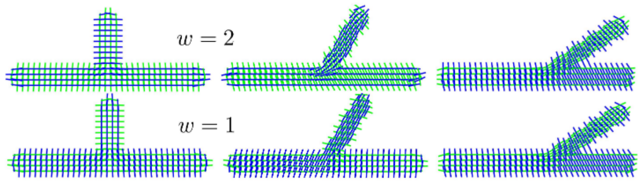


Figure 9: *The effect of the w weight can be seen on the final resulting field. ssigning greater importance to alignment means that less of an angle is needed for a T-junction to become a Y-junction.*

5.6. Limitations

Like other vectorization algorithms for line drawings from this category, such as those of [BS19], [BLW16], [FLB16], [NHS*13], and [PNCB21], our method works best on drawings without extensive shading. Furthermore, our method is dependent on determining the narrowband, done as a simple color intensity threshold τ, which may pose challenges on images with a very noisy background. Similarly, our method suffers when contrast adjustment is poor, which we demonstrate in the supplementary materials. These shortcomings are shared by narrowband-based methods [BS19, PNCB21], and we envision using more advanced learning-based approaches for background/foreground extraction as a future remedy. Lastly, we acknowledge that the current implementation of the method has relatively long runtimes, and we might look to improve these with more sophisticated optimization techniques in later investigations.

6. Conclusions and Future Work

Our method generates singularity-free frame fields, a foundational component that had been missing from state-of-the-art frame field–based vectorization algorithms. To this end, we leverage the framework of trivial connections, coupled with a simpler frame field rep-