BendFields: Regularized Curvature Fields from Rough Concept Sketches

Emmanuel Iarussi[1], David Bommes[1,2], Adrien Bousseau[1]
[1]Inria, [2]RWTH Aachen University

Designers frequently draw curvature lines to convey bending of smooth surfaces in concept sketches. We present a method to extrapolate curvature lines in a rough concept sketch, recovering the intended 3D curvature field and surface normal at each pixel of the sketch. This 3D information allows us to enrich the sketch with 3D-looking shading and texturing.

We first introduce the concept of *regularized curvature lines* that model the lines designers draw over curved surfaces, encompassing curvature lines and their extension as geodesics over flat or umbilical regions. We build on this concept to define the orthogonal cross field that assigns two regularized curvature lines to each point of a 3D surface. Our algorithm first estimates the projection of this cross field in the drawing, which is non-orthogonal due to foreshortening. We formulate this estimation as a scattered interpolation of the strokes drawn in the sketch, which makes our method robust to sketchy lines that are typical for design sketches. Our interpolation relies on a novel smoothness energy that we derive from our definition of regularized curvature lines. Optimizing this energy subject to the stroke constraints produces a dense non-orthogonal 2D cross field, which we then lift to 3D by imposing orthogonality. Thus, one central concept of our approach is the generalization of existing cross field algorithms to the non-orthogonal case.

We demonstrate our algorithm on a variety of concept sketches with various levels of sketchiness. We also compare our approach with existing work that takes clean vector drawings as input.

Categories and Subject Descriptors: I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*Curve, surface, solid, and object representations*

Additional Key Words and Phrases: cross field, non-orthogonal cross field, regularized curvature line, sketch-based modeling, sketch editing, line drawing interpretation

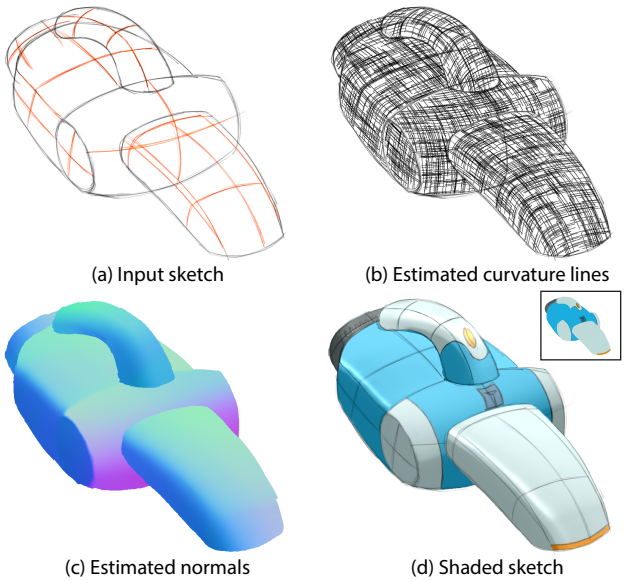


Fig. 1: Designers commonly draw curvature lines to emphasize surface bending in concept sketches (a, red lines). We extrapolate these lines in a bitmap to form a dense cross field, from which we estimate the 3D curvature directions and the surface normal at each pixel (b,c). We use this information to compute shading, greatly enhancing the 3D look of the sketch (d). Note that this sketch is composed of several layers to represent the main body, handle and nozzle of the vacuum cleaner.

1. INTRODUCTION

Designers frequently draw curvature lines to convey bending over smooth surfaces in concept sketches [Eissen and Steur 2011] (Figure 1(a)). We introduce a method to extrapolate strokes in a sketch to form a dense *cross field* that assigns two curvature lines to each pixel of the drawing while extending smoothly over flat and umbilical regions, where the lines of curvature are ill-defined (Figure 1(b)). By estimating this curvature information, our method enables the application of several 3D curvature-based algorithms to 2D drawings. For example, curvature lines have been used to guide parameterization [Ray et al. 2006], texture synthesis [Lefebvre and Hoppe 2006], cross-hatching [Hertzmann and Zorin 2000], and can also provide surface normals for local shading. By applying these algorithms directly in the sketch, our approach allows designers to enhance the 3D look of their drawing during the early stages of design (Figure 1(d)), when 3D modeling is often distracting and time-consuming [Pipes 2007; Bae et al. 2008; Shao et al. 2012].

Our method takes as input rough bitmap sketches drawn on paper or with painting software, with lines often made of sketchy overlapping strokes. Our algorithm copes with such unorganized rough inputs in two steps. We first express the 2D projection of the two lines of curvature at each pixel as a scattered interpolation of the nearby lines. This interpolation results in a 2D *non-orthogonal* cross field, as the two projected curvature lines are not orthogonal due to foreshortening. We then lift this cross field to 3D by leveraging the fact that the lines of curvature should be orthogonal in 3D. We call the resulting 3D cross field over the image a *BendField*. We finally compute surface normals as the cross-product of the two 3D directions at each pixel.

While expressing our problem as a scattered interpolation makes our method robust to sketchy inputs, it requires us to address two challenges. First, each stroke in the sketch can only constrain one of the two lines of the cross field. Thus, for each pair of strokes there is a discrete choice of making them either parallel or transversal in

DOI 10.1145/2710026
http://doi.acm.org/10.1145/2710026

the interpolation. Existing work in the context of *orthogonal* cross fields over 3D surfaces tackles this ambiguity either by nonlinear formulations that exploit orthogonality to map the lines into a space where they become equal [Ray et al. 2009; Knöppel et al. 2013] or by discrete variables that are part of the optimization [Bommes et al. 2009]. We extend the second formulation to our context by introducing a new representation for non-orthogonal cross fields, using one angle to encode the orientation of an orthogonal cross and a second angle to encode the deviation from orthogonality.

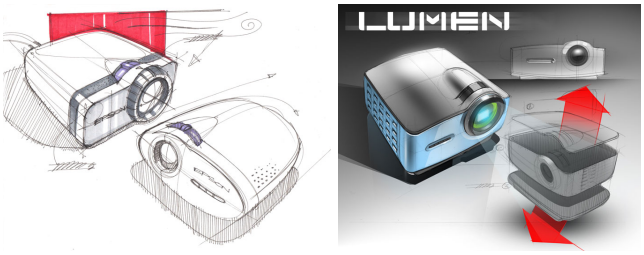
The second challenge we address is the design of an interpolation energy that produces plausible lines of curvature at each pixel, subject to the constraints provided by the sketch. To do so, we conduct a mathematical analysis of curvature lines over smooth surfaces. We deduce the concept of *regularized curvature lines* that model the way lines in a sketch align with curvature directions when these directions are well defined, and become geodesic, i.e. shortest path, in flat or umbilical regions. We then derive that, under parallel projection, the two families of regularized curvature lines that compose the 2D cross field are smooth along each-other. We use this specifically-designed measure of smoothness to extrapolate the lines over the sketch. The resulting cross field provides a vivid sense of the 3D shape, smoothly interpolating the prescribed curvature lines without introducing extraneous surface variations. We use our cross fields to enrich a variety of concept sketches with 3D-looking shading and texturing.

In summary, we introduce three contributions:

- A method to estimate 3D consistent curvature and normal fields from rough 2D concept sketches. In contrast to existing methods that require clean vectorial curves, our approach is able to handle sketchy drawings provided in bitmap form.
- A representation and optimization algorithm for non-orthogonal cross fields.
- The mathematical formulation of regularized curvature lines, from which we derive a novel smoothness energy to extrapolate curvature lines.

2. RELATED WORK

Sketch editing. Designers commonly use line drawings to quickly explore 3D concepts without the burden of CAD modeling [Pipes 2007; Bae et al. 2008]. While rough *ideation sketches* are often only made of lines (Figure 2(a)), shading and textures are subsequently painted to produce *presentation sketches* (Figure 2(b)) that better communicate 3D appearance to the clients and decision makers [Eissen and Steur 2011].



(a) Rough ideation sketch (b) Shaded presentation sketch

Fig. 2: Typical design sketches by Spencer Nugent on sketch-a-day.com©. (a) Designers draw rough ideation sketches to explore early concepts. (b) Shading is subsequently painted for presentation to decision makers.

Several sketch-editing tools have been proposed to facilitate colorization, shading and texturing of line drawings. Scribble-based interfaces propagate colors in empty or uniformly-textured regions [Qu et al. 2006; Sýkora et al. 2009]. Inspired by modeling systems based on shape inflation [Igarashi et al. 1999; Nealen et al. 2007], *Lumo* and subsequent algorithms consider that the lines in a drawing delineate an inflatable proxy on which they compute shading and texturing [Johnston 2002; Joshi and Carr 2008; Winnemöller et al. 2009; Sýkora et al. 2011]. Inflated normal maps have also been used to generate stylized shading that mimics artistic guidelines [Lopez-Moreno et al. 2013] or that follows plausible shading flows [Vergne et al. 2012]. Sýkora et al. [2014] further improve realism by generating a bas-relief proxy with consistent depth ordering that they use to compute global illumination effects. Finally, Cole et al. [2012] adopt an example-based approach to estimate normal fields from contour drawings of smooth abstract shapes. The above methods produce convincing results on cartoon blobby shapes solely defined by contours. In contrast, we target man-made shapes from concept sketches and leverage interior curvature lines to control the shape away from contours. To do so, we propose a novel smoothness energy that better preserves curvature lines than the harmonic energies used in shape inflation.

Closer to our work are the *CrossShade* and *True2Form* algorithms [Shao et al. 2012; Xu et al. 2014] which generate normal maps and 3D curve networks from concept sketches. Both methods work with vector drawings and estimate 3D information from intersecting curves locally aligned with curvature directions. Our method targets a similar application domain as CrossShade but handles rough bitmap drawings rather than clean vector art. Designers often produce rough preliminary sketches in a bitmap form, either from scanned pen-on-paper drawings or from painting software, which requires less precision than vector tracing. Working with bitmaps requires us to formulate the extrapolation of curvature lines as a scattered data interpolation rather than the parametric Coons interpolation used by CrossShade.

While vectorization algorithms could be used to convert bitmaps into vectorial curves, state-of-the-art algorithms remain challenged by sketchy drawings [Noris et al. 2013], or require the temporal information provided by digital sketching [Orbay and Kara 2011]. When applied on a rough sketch, the recent method by Noris et al. [2013] produces multiple curves in the presence of overlapping strokes (Figure 3(a,b)). While filtering the sketch can group overlapping strokes [Bartolo et al. 2007], curve segments cannot be automatically merged at junctions because of ambiguous configurations (Figure 3(c,d)). As a result, vectorized segments should be manually edited and merged to form suitable input for CrossShade, which assumes that each curvature line is formed by a single curve. We designed our approach to bypass vectorization and avoid all these shortcomings (Figure 3(e,f)).

Line fields in images. Kass and Witkin first proposed to analyze oriented patterns by computing a smooth *line field* (i.e. 2-direction vector field) perpendicular to the strong gradients in an image [Kass and Witkin 1987]. Similar image-guided line and vector fields have been used for image filtering [Weickert 1999; Kang et al. 2009; Kyprianidis and Kang 2011], edge detection [Kang et al. 2007], painterly rendering [Haeberli 1990]. While we take inspiration from this body of work, our goal is to estimate a *cross field* rather than a line field, which requires us to assign the strong gradients in the image to one of two lines. The structure tensor [Harris and Stephens 1988; Aach et al. 2006] and streerable filters [Freeman and Adelson 1991] can be used to estimate multiple orientations at corners and junctions but their response vanishes away

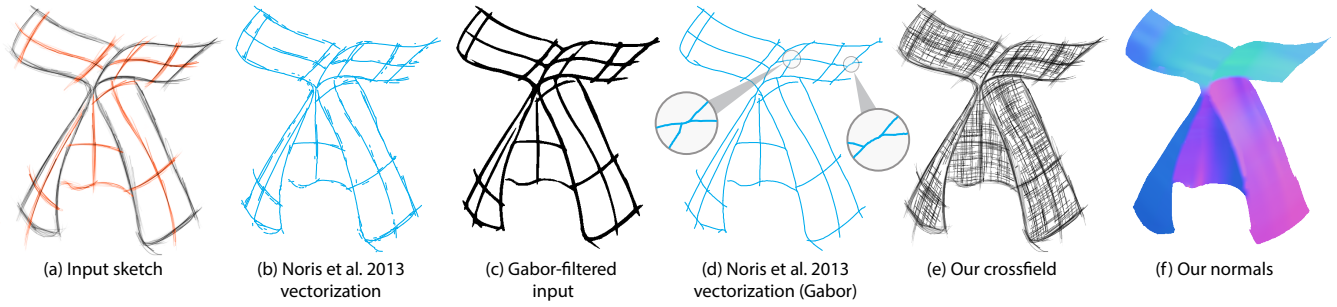


Fig. 3: Limitations of vectorization. Rough sketches are made of many overlapping strokes (a) that vectorization algorithms [Noris et al. 2013] interpret as multiple short curves (b). The Gabor filter of [Bartolo et al. 2007] groups many strokes together but also tends to smooth junctions between intersecting strokes (c). As a result, junctions form ambiguous configurations (d, inset) that prevent the merging of segments into continuous curves suitable for CrossShade [Shao et al. 2012]. We propose an alternative approach to directly estimate curvature information and normals from the rough sketch (e,f), alleviating the need for vectorization and manual cleanup. The results of [Noris et al. 2013] were produced with default parameters.

from the image contours. In addition, notice that line fields are different mathematical objects than cross fields. Since line fields cannot represent the quarter-index singularities of cross fields, they are inappropriate in our context (cf. [Ray et al. 2008]).

Line fields and cross fields on surfaces. Many surface-processing algorithms rely on the definition of smooth line fields or cross fields over 3D objects [Knöppel et al. 2013]. These fields have been used to guide parameterizations [Ray et al. 2006], texture synthesis [Praun et al. 2000; Lefebvre and Hoppe 2006; Fisher et al. 2007], cross-hatching strokes [Hertzmann and Zorin 2000], quad remeshing [Alliez et al. 2003; Bommes et al. 2013]. Most of these methods work with *orthogonal* cross fields aligned with the principal directions of curvature of a *known* 3D surface. Our goal is to allow the use of this family of algorithms directly in the 2D drawing of an *unknown* surface by estimating an orthogonal cross field from a sparse set of projected curvature lines. To achieve this goal, our algorithm first generates the *non-orthogonal* cross field which results from projection into the 2D drawing plane.

Liu et al. [2011] describe an algorithm to compute *conjugate* cross fields over 3D surfaces, which are non-orthogonal. Their approach generalizes the orthogonal cross field approach of Hertzmann and Zorin [2000], which due to its nonlinear and non convex energy functional has the tendency to produce non-optimal additional singularities. We instead adopt a mixed-integer formulation which overcomes such problems using an iterative optimizer [Bommes et al. 2009], where in each step a convex linear problem is solved.

Concurrently to our work, two alternatives to handle non-orthogonal cross fields were developed. Panozzo et al. [2014] introduce the concept of *frame fields*, which are non-orthogonal and non-unit-length generalization of cross fields. They model a frame field as a combination of an orthogonal cross field – generated with a mixed-integer algorithm [Bommes et al. 2009] – and a harmonic deformation field that captures scaling and skewness. Instead of splitting the optimization into two subsequent parts, we directly generalize [Bommes et al. 2009] to determine skewness and unit-length cross field in one combined step.

Diamanti et al. [2014] propose the powerful idea of *polyvector fields*, which encode arbitrary sets of vectors as roots of complex polynomials, and thus can handle non-orthogonal cross fields as a special case. This method also performs a single combined optimization but the harmonic interpolation of boundary constraints is done in the space of polynomial coefficients while we interpolate rotations in the more natural space of angles.

One major difference between the two concurrent approaches and ours is that both frame fields and polyvector fields are interpolated from sparse *frame constraints*, i.e. they require local constraints on both lines of the cross field. In contrast, our method needs to handle *partial* constraints since sketched strokes only constrain one of the directions of the field while leaving the transverse direction free for optimization. Another difference is that our angle-space parametrization enables the optimization of unit-length fields, which prevents undesired field shrinkage experienced by Diamanti et al. (see Figure 14 in their paper).

All the aforementioned methods use a harmonic energy to generate smooth fields. We show that applying a harmonic energy in our context does not produce plausible curvature lines as it does not account for the way such lines behave on a surface in 3D.

3. OVERVIEW

Designers extensively use free-hand sketches to quickly visualize the shape they envision [Eissen and Steur 2011]. Figure 2 shows a real-world example of a projector. In such sketches, skilled artists capture all surface information by drawing two types of lines:

- *Discontinuity lines* mark the sharp creases and silhouettes that delineate smooth surface patches (Figure 1a, black lines).
- *Curvature lines* convey bending within the surface patches and extend smoothly in flat or umbilical regions (Figure 1a, red lines).

While concept sketches are typically made of a sparse set of lines, they prove to be sufficient to describe 3D shapes since viewers mentally extrapolate the curvature lines to form a dense network on the imagined surface, assuming that the geometry of a curve is representative of the geometry of the surface in its vicinity [Stevens 1981; Bessmeltsev et al. 2012]. Mathematically speaking, this network corresponds to a smooth *cross field*, which associates two orthogonal lines to each point of the surface. Our goal is to mimic viewers' inference to recover the cross field over the visible 3D surface conveyed by a concept sketch.

Figure 4 illustrates the main steps of our algorithm. We take as input a bitmap line drawing, as commonly drawn on paper or with

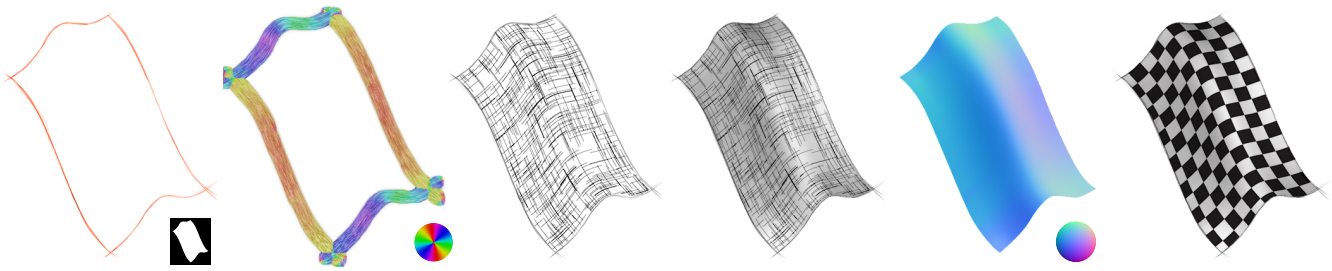
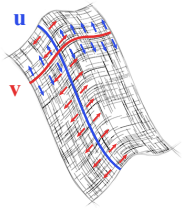


Fig. 4: Our algorithm takes as input a bitmap sketch (a) from which we estimate the local orientation of the strokes (b). Assuming that the strokes represent curvature lines on an imaginary surface, our formulation extrapolates them as a non-orthogonal cross field that mimics the behavior of a projected curvature field (c). We then lift the cross field to 3D by imposing orthogonality constraints (d). We finally compute normals and texture parameterization from the 3D cross field (e,f).

painting tools like Adobe Photoshop and Autodesk SketchBook, and a binary mask to identify pixels that belong to the object (Figure 4a). We additionally assume that users draw discontinuity and curvature lines in a different color and decompose complex models by drawing independent parts in different layers. Since designers draw in 2D, the curvature lines in the sketch only provide us with constraints on the *projected* cross field, which is non-orthogonal due to foreshortening. The first part of our algorithm consists in estimating this smooth non-orthogonal cross field from the local orientation of the lines in the sketch (Figure 4b,c). In a second step, we lift the cross field to 3D by assuming parallel projection and constraining the 3D lines to be orthogonal (Figure 4d). This 3D information allows us to apply several geometry-processing algorithms over the drawing, including the computation of surface normals for shading (Figure 4e) and parameterization for texture mapping (Figure 4f).

Our two main technical contributions are described in Sections 5 and 6. Section 5 introduces the concept of *regularized curvature lines* that encompass the curvature lines in a sketch and their extension as geodesics over flat or umbilical regions. From this concept we deduce a variational formulation for the non-orthogonal cross field. In a nutshell, our energy encourages the two families of lines that compose the cross field to be smooth along each other, as illustrated in the inset where the lines of the family $\mathbf{u}$ are smooth along the family $\mathbf{v}$ and vice-versa. We derive this property from the fact that curvature lines on a surface are free of geodesic torsion.



Section 6 then describes how to solve for the cross field that minimizes our energy subject to the stroke constraints. The main challenge we face is that, while each stroke constrains one of the two lines ($\mathbf{u}$,$\mathbf{v}$) in the cross field, we don't know which of the four directions $\{\mathbf{u}, -\mathbf{u}, \mathbf{v}, -\mathbf{v}\}$ is constrained a priori. We handle these discrete degrees of freedom using a mixed-integer formulation that jointly solves for the smooth cross field and the assignment of each constraint to one direction. Our mixed-integer formulation is also necessary to handle singularities in the cross field, in which case the surface needs to be split into charts related by discrete permutations of the directions that form the cross field.

4. STROKE CONSTRAINTS

Given a rough bitmap sketch, we first need to estimate the tangent of the strokes which will then act as constraints to align the cross field (Figure 4b). We obtain this information from the structure tensor, a popular tool to estimate local orientation in images [Kyprianidis and Kang 2011]. The structure tensor of an image $I(x, y)$ is expressed by means of the partial derivatives $I_x = \partial I/\partial x$ and $I_y = \partial I/\partial y$ as

$$S(I) = \begin{pmatrix} I_x \cdot I_x & I_x \cdot I_y \\ I_x \cdot I_y & I_y \cdot I_y \end{pmatrix}.$$

Its major and minor eigenvectors provide the directions of maximum and minimum change in the image. The tangent along a stroke is thus given by the minor eigenvector, except at corners and junctions where the presence of two orientations make the two eigenvalues almost equal. We use the magnitude of the minor eigenvalue to attenuate the influence of these unstable corners and junctions in the cross field estimation. In practice, we normalize the minor eigenvalue λ_2 at each pixel i by the maximum minor eigenvalue of the image to obtain a weight $w_i = 1 - \frac{\lambda_2(i)}{\max(\lambda_2)} \in [0, 1]$ that we apply on the orientation constraints.

We found that applying a bilateral filter on the structure tensor produces smoother estimates while preserving abrupt changes of direction. We used the same range parameter $\sigma_r = 2$ for all inputs, and different presets for the spatial extent σ_s to account for various levels of sketchiness, as discussed in Section 9. Figure 5 illustrates the local orientations and the attenuation weight at cor-

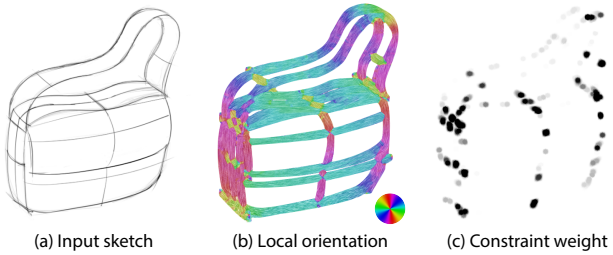


Fig. 5: We use the structure tensor to estimate the local orientation of the strokes in a sketch (b). We attenuate the strength of the orientation constraints near corners and junctions, where the estimation is unstable (c).

ners and junctions. Note that while we visualize the orientations as a wide strip around all strokes, we only apply constraints on the pixels covered by curvature strokes.

5. ESTIMATING CURVATURE FIELDS

The 2D strokes in the sketch correspond to the projection of a subset of the curvature lines. The goal of this section is to derive a proper way of smoothly extrapolating the sparse strokes to the dense curvature network of the intended surface. This step, which results in a dense (non-orthogonal) cross field, is illustrated in Figure 4c. We first provide an intuitive motivation for our 2D smoothness energy and its relation to interpolants used in prior work (Section 5.1). We then provide a formal derivation of this energy from properties of curvature lines and fields (Section 5.2) and extend the energy to lift the cross field to 3D (Section 5.3).

5.1 Motivation for the BendField energy

For clarity, we assume for now that each stroke constraint has been assigned to one of the two lines ($\mathbf{u}$,$\mathbf{v}$) of the cross field. Our goal is to generate a smooth field aligned with these constraints. Prior work on the design of line fields [Fisher et al. 2007], cross fields [Knöppel et al. 2013] and normal fields [Johnston 2002] use an harmonic energy to measure the smoothness of a field. In our context, the harmonic energy independently penalizes strong gradients in the $\mathbf{u}$ and $\mathbf{v}$ fields

$$E_h = \int ||\nabla \mathbf{u}||^2 + ||\nabla \mathbf{v}||^2.$$

Figure 6(a) illustrates the behavior of this energy, that tends to flatten the surface away from the constraints. To prevent such flattening, Shao et al. [2012], Biard et al. [2010] and Bessmeltsev et al. [2012] interpolate normals and surfaces over quads bounded by curvature lines using parametric Coons patches [Farin and Hansford 1999]. Since Coons patches interpolate the boundary segments linearly, they naturally align the ($\mathbf{u}$,$\mathbf{v}$) iso-lines to these boundaries, as shown in Figure 6(b). This alignment corresponds well to viewer expectation that a given curve is representative of other curves in its vicinity [Stevens 1981]. We designed our smoothness energy to produce a similar alignment in a scattered interpolation fashion. More precisely, our *BendField* energy relies on the *covariant derivatives* $\nabla_{\mathbf{u}}\mathbf{v}$ and $\nabla_{\mathbf{v}}\mathbf{u}$ to measures the smoothness of the vector field $\mathbf{u}$ along the streamlines of $\mathbf{v}$, and vice versa

$$E_{\text{bend2D}} = \int ||\nabla_{\mathbf{u}}\mathbf{v}||^2 + ||\nabla_{\mathbf{v}}\mathbf{u}||^2$$

where $\mathbf{v} = (v_x, v_y)^T$ and

$$\nabla_{\mathbf{u}}\mathbf{v} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{pmatrix} \mathbf{u}.$$

Note that the covariant derivatives couple the two vector fields, which harmonic and biharmonic energies cannot do. While additional work is needed to formalize the connection between our energy and Coons patches, Figure 6(c) shows that they behave similarly, even though our algorithm and CrossShade [Shao et al. 2012] do not produce strictly identical normals on such a complex freeform surface patch (see Section 9 for additional comparisons).

5.2 Formal derivation of the BendField energy from properties of curvature lines and fields

Curvature lines. Given a parameterized surface $\mathbf{S}(u, v)$ embedded in $\mathbb{R}^3$, all curves on this surface can be described by

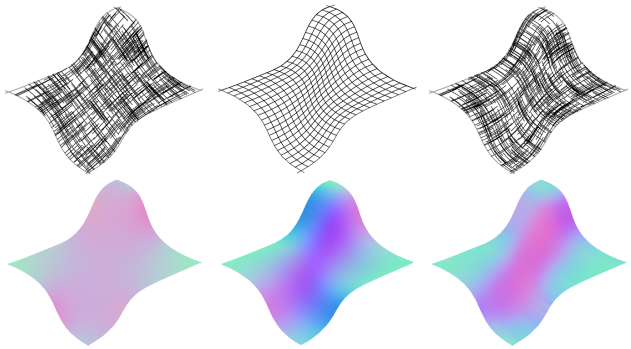


Fig. 6: The harmonic energy produces a flat surface patch that does not preserve curvature away from the strokes (a). Prior work uses Coons patches to better capture the directionality of the boundary curves (b) [Shao et al. 2012]. Our energy produces a similar interpolation by making the $\mathbf{u}$ and $\mathbf{v}$ vector fields smooth along each-other (c).

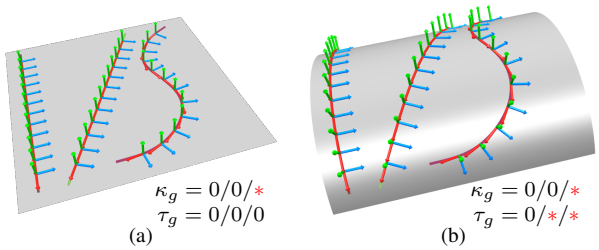
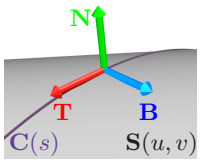


Fig. 7: The geodesic curvature κ_g and geodesic torsion τ_g of three curves are listed from left to right with $*$ indicating nonzero. In the plane, only straight curves are geodesics ($\kappa_g = 0$) while every curve has $\tau_g = 0$ since the Darboux frame cannot rotate around T without changing the tangent plane. In (b) the plane is deformed to a cylinder. Now the second curve, although being geodesic, has a nonzero τ_g. Notice the implied rotation of B and N around T which results from misalignment to the bending direction and accordingly vanishes for the curvature-aligned leftmost curve.

univariate functions $\mathbf{C}(s) := \mathbf{S}(u(s), v(s))$. For simplicity in the following we assume an arc length parametrization such that $s \in [0, L]$ with L being the length of $\mathbf{C}$. The curvature properties of such a curve w.r.t. to its surface are characterized by the behavior of the so called Darboux frame. This orthonormal frame $(\mathbf{T}, \mathbf{B}, \mathbf{N}) \in \mathbb{R}^{3\times 3}$ consists of the unit tangent $\mathbf{T} = \frac{d\mathbf{C}}{ds}$, the surface normal $\mathbf{N}$ and the tangent normal $\mathbf{B} = \mathbf{N} \times \mathbf{T}$. While traversing the curve, this orthonormal frame undergoes rotations. The rotational speed around the axes of the frame are known as *geodesic torsion* $\tau_g = \mathbf{N} \cdot \frac{d\mathbf{B}}{ds}$, *normal curvature* $\kappa_n = \mathbf{N} \cdot \frac{d\mathbf{T}}{ds}$ and *geodesic curvature* $\kappa_g = \mathbf{T} \cdot \frac{d\mathbf{B}}{ds}$ for rotations around $\mathbf{T}$, $\mathbf{B}$ and $\mathbf{N}$ respectively. Figure 7 illustrates the geometric intuition behind these notions.



Important in our context is the observation that curvature lines are characterized by vanishing geodesic torsion. More precisely, a curve is a curvature line if and only if $\tau_g = 0$ [do Carmo 1976; Biard et al. 2010]. The name curvature line reflects the fact that such a curve is always tangent to one of the principal curvature

directions of the surface. Intuitively, a surface curve has non-zero geodesic torsion if the surface bends most in a direction that is not $\mathbf{T}$ nor $\mathbf{B}$ (Figure 7b), which contradicts the definition of curvature lines.

In each point of the surface where the two principal curvatures κ_1 and κ_2 are different ($\kappa_1 \neq \kappa_2$), exactly two unique curvature lines intersect. However, in flat and umbilical regions with $\kappa_1 = \kappa_2$ we have the ambiguity that every curve through these regions is a curvature line. This is also reflected by the fact that the geodesic torsion can be computed as $\tau_g = 0.5(\kappa_2 - \kappa_1)sin(2\theta)$ where θ is the angle between $\mathbf{T}$ and the direction of minimal curvature $\mathbf{K}_1$. Note that τ_g is proportional to the *curvature anisotropy* $\kappa_2 - \kappa_1$ and thus linearly vanishes in isotropically curved regions, independently of the tangent direction. We conjecture that designers avoid such ambiguity of curvature lines by sketching what we call *regularized curvature lines*, which we define more precisely next.

Regularized curvature lines. In areas of high curvature anisotropy, i.e. where $|\kappa_2 - \kappa_1|$ is large, designers sketch smooth curves that strictly follow principal curvature directions. However, the more isotropic the curvature gets, the more geodesic the sketched curves tend to be. Geodesic curves are characterized by vanishing geodesic curvature $\kappa_g = 0$, i.e. the curve does not bend within the tangent plane. Therefore sketch curves behave like minimizers of the functional

$$E_\alpha = \int_C \tau_g^2 + \alpha\kappa_g^2 ds \tag{1}$$

The geodesic curvature κ_g behaves like a regularizer for τ_g since it becomes dominant in regions of isotropic curvature $\kappa_1 \approx \kappa_2$, where τ_g vanishes. We refer to this family of curves as *regularized curvature lines*, where α controls the strength of the regularization.

We hypothesize that designers apply a similar regularization when sketching because the trajectory of curvature lines is hard to predict in near-isotropic regions. This regularization toward geodesics is also supported by prior observations that designers draw curves aligned with curvature and geodesic lines [Shao et al. 2012]. Similarly, perceptual studies suggest that people interpret intersecting curves in a drawing as aligned with principal directions of curvature [Stevens 1981; Mamassian and Landy 1998] and geodesics [Knill 1992]. This concept also provides a mathematical definition to the notion of *flowlines* mentioned in prior work [Bessmeltsev et al. 2012; Zhuang et al. 2013]. Finally, notice the closely related approach of modern quad meshing algorithms [Bommes et al. 2013] that align the quad mesh solely in anisotropic curvature regions while preferring smoothness everywhere else.

Regularized curvature cross field. Ultimately we are searching for the full (regularized) curvature network that extends the sketch curves to a dense orthogonal cross field. Therefore we have to extend the previous concept from curves on a surface $\mathbf{S}$ to cross fields that are tangent to $\mathbf{S}$. As an intermediate step first observe that the generalization to a unit-length tangent vector field $\mathbf{T}$, which is now defined for each point on S, yields the functional

$$E_\alpha(\mathbf{T}) = \int_{\mathbf{S}} \tau_g^2 + \alpha\kappa_g^2 dA = \int_{\mathbf{S}} (\mathbf{N} \cdot \nabla_{\mathbf{T}}\mathbf{B})^2 + \alpha(\mathbf{T} \cdot \nabla_{\mathbf{T}}\mathbf{B})^2 dA$$

where $\nabla_{\mathbf{T}}$ is the derivative along the streamline tangent to $\mathbf{T}$. This (extrinsic) directional derivative is necessary since the curves are now only implicitly defined as the streamlines of $\mathbf{T}$. Since a cross field is nothing more than a vector field on a 4-sheeted covering [Kälberer et al. 2007] that can locally be parameterized by two vector fields $\mathbf{U}$ and $\mathbf{V}$, we end up with

$$\begin{aligned} E_\alpha(\mathbf{U}, \mathbf{V}) = \int_{\mathbf{S}} &(\mathbf{N} \cdot \nabla_{\mathbf{U}}\mathbf{V})^2 + (\mathbf{N} \cdot \nabla_{\mathbf{V}}\mathbf{U})^2 \\ &+ \alpha((\mathbf{U} \cdot \nabla_{\mathbf{U}}\mathbf{V})^2 + (\mathbf{V} \cdot \nabla_{\mathbf{V}}\mathbf{U})^2) dA. \end{aligned} \tag{2}$$

Notice that in case of cross field singularities it is not possible to globally represent a smooth cross field by two smooth representative vector fields $\mathbf{U}$ and $\mathbf{V}$. This technical problem, however, can be easily handled by splitting the surface into coordinate charts that are related by discrete *transition functions* that permute the vectors of the cross field to align a cross in one chart to a cross in another chart [Ray et al. 2006; Bommes et al. 2009]. We explain the concept of transition functions in detail in Section 6.

Difficulty of sketch reconstruction. Equipped with the mathematical description of the 3D curvature network that is intended by the given 2D strokes we are now ready to state the optimization problem of sketch reconstruction. We are searching for a minimizer of $E_\alpha(\mathbf{U}, \mathbf{V})$ subject to local unit length and orthogonality constraints $||\mathbf{U}|| = ||\mathbf{V}|| = 1$, $\mathbf{U} \cdot \mathbf{V} = 0$ and $\mathbf{N} = \mathbf{U} \times \mathbf{V}$, where we additionally require that the 2D strokes locally align to the 2D projection of $\mathbf{U}$ or $\mathbf{V}$. This is a very hard nonlinear mixed-integer problem since both the surface S and its tangent cross field $(\mathbf{U}, \mathbf{V}) \in \mathbb{R}^{3\times 2}$, including discrete transition functions between charts, are unknown. Instead of optimizing it directly, we aim at first optimizing for its 2D projection which can then be used to estimate an appropriate initial solution for the 3D problem.

2D projection of curvature cross field. By parallel projection $P((x, y, z)^T) = (x, y)^T$, the unknown surface $\mathbf{S}$ becomes a known part of the Euclidean plane. However, this simplification comes at the cost of a distorted metric. Due to foreshortening the projection heavily affects dot and cross products such that $E_\alpha(\mathbf{U}, \mathbf{V})$ is useless for our 2D setting. However, by restricting to the case $\alpha = 1$ it is possible to obtain a suitable formulation with a stable behavior under projection.

First notice that for curves we have $\int_C ||\frac{d\mathbf{B}}{ds}||^2 ds = E_{\alpha=1}$. This can be verified by projecting $\frac{d\mathbf{B}}{ds}$ on the orthonormal basis $\mathbf{T}, \mathbf{B}, \mathbf{N}$ and exploiting $\frac{d\mathbf{B}}{ds} \cdot \mathbf{B} = 0$ since $\mathbf{B}$ is unit length. This means that for $\alpha = 1$, Equation (1) becomes

$$E_1 = \int_C ||\frac{d\mathbf{B}}{ds}||^2 ds = \int_C ||\nabla_{\mathbf{T}}\mathbf{B}||^2 ds$$

or equivalently for the cross field case

$$E_1(\mathbf{U}, \mathbf{V}) = \int_{\mathbf{S}} ||\nabla_{\mathbf{U}}\mathbf{V}||^2 + ||\nabla_{\mathbf{V}}\mathbf{U}||^2 dA.$$

Intuitively, our BendField energy favors parallelism of one vector field along the streamlines of the other one, which is consistent with the fact that non-parallelism can only be introduced by a non-zero geodesic torsion or geodesic curvature of the streamlines. Since parallelism is preserved by parallel projection, we can measure exactly the same quantity in a 2D projection, leading to

$$\boxed{E_{\text{bend2D}}(\mathbf{u}, \mathbf{v}) = \int_I ||\nabla_{\mathbf{u}}\mathbf{v}||^2 + ||\nabla_{\mathbf{v}}\mathbf{u}||^2 dA} \tag{3}$$

where I is the image plane, $\mathbf{u} = P(\mathbf{U})$ and $\mathbf{v} = P(\mathbf{V})$ are 2D projections of $\mathbf{U}$ and $\mathbf{V}$, $\nabla_{\mathbf{u}}\mathbf{v}$ and $\nabla_{\mathbf{v}}\mathbf{u}$ are covariant derivatives

$$\nabla_{\mathbf{u}}\mathbf{v} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \end{pmatrix} \mathbf{u}$$

where $\mathbf{v} = (v_x, v_y)^T$. The optimization of Equation (3) is based on our novel non-orthogonal cross field representation, which is the topic of Section 6. Since lengths and angles are not preserved by parallel projection, the unit-length and orthogonality constraints can be omitted for the 2D optimization. However, they will be reinjected in the following step, which lifts the 2D minimizer of Equation (3) to 3D. Note also that we ultimately minimize E_{bend2D} subject to the stroke constraints, which prevents the trivial solution of a null cross-field.

5.3 Lifting the cross field to 3D.

Solving the previous optimization problem provides us with a good estimate of the 2D projection of $\mathbf{U}$ and $\mathbf{V}$. We obtain a local 3D estimate based on the knowledge that, in 3D, $\mathbf{U} \cdot \mathbf{V} = u_x v_x + u_y v_y + u_z v_z = 0$ due to orthogonality, and the additional assumption that designers favor viewpoints that minimize overall foreshortening [Eissen and Steur 2011; Shao et al. 2012]. The minimal foreshortening tells us that $u_z = -v_z$ if the $\mathbf{u}$ and $\mathbf{v}$ vectors form an angle of less than $\frac{\pi}{2}$, while we have $u_z = v_z$ otherwise. Combining both constraints leads to a quadratic equation with two potential solutions $u_z = \pm\sqrt{|\mathbf{u} \cdot \mathbf{v}|}$. These two solutions reflect the global ambiguity between a convex and a concave surface patch, which cannot be resolved from the sketch alone [Shao et al. 2012]. We obtain a globally-consistent solution by selecting for each pixel the candidate that produces the smoothest u_z field overall, subject to a few user indications to distinguish between the convex and concave interpretation (Section 8). We express this problem as a binary labeling, which we solve with [Kolmogorov 2006] as described in the Appendix.

Finally we use the globally consistent and normalized estimates $\mathbf{U} = (u_x, u_y, u_z)^T$ and $\mathbf{V} = (v_x, v_y, v_z)^T$ as an initial solution for optimizing energy (2). While in theory we should constrain $\mathbf{U}$ and $\mathbf{V}$ to have unit length, we found that the optimization can be greatly simplified by ignoring this constraint and by solely optimizing for u_z and v_z while keeping the 2D components constant. As a positive side effect, this choice not only regularizes the resulting cross field in length but also in direction. Therefore in addition we can safely simplify the functional by dropping the non-linear geodesic terms belonging to α. We still don't know the surface $\mathbf{S}$ such that we again approximate the energy over the image domain I, leading to

$$\boxed{E_{\text{bend3D}}(u_z, v_z) = \int_I (\mathbf{N} \cdot \nabla_{\mathbf{u}}\mathbf{V})^2 + (\mathbf{N} \cdot \nabla_{\mathbf{v}}\mathbf{U})^2 + \epsilon_f (u_z^2 + v_z^2) dA} \quad (4)$$

where the last term is weighted by $\epsilon_f = 0.005$ to weakly regularize the solution towards minimal foreshortening, as in [Shao et al. 2012]. This is a hybrid formulation, where we compute a 3D cross field over the 2D image domain. Accordingly, the covariant derivatives are given by

$$\nabla_{\mathbf{u}}\mathbf{V} = \begin{pmatrix} \partial v_x/\partial x & \partial v_x/\partial y \\ \partial v_y/\partial x & \partial v_y/\partial y \\ \partial v_z/\partial x & \partial v_z/\partial y \end{pmatrix} \mathbf{u}$$

with $\mathbf{V} = (v_x, v_y, v_z)^T$. Note that thanks to our simplifications this energy is quadratic in the unknown z components, which can be verified by considering that $\mathbf{N} = \mathbf{U} \times \mathbf{V}$ and that $\mathbf{u}$ and $\mathbf{v}$ are constant. We optimize this functional subject to nonlinear orthogonality constraints $\mathbf{U} \cdot \mathbf{V} = 0$ by an interior point method as described in more detail in Section 8. We finally normalize the resulting $\mathbf{U}$ and $\mathbf{V}$ vectors and compute the surface normal from their cross product.

5.4 Algorithm Overview

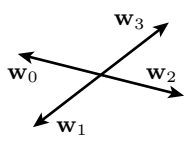
The ultimate goal of our algorithm is to extrapolate the given 2D strokes into a 3D BendField. Since 3D BendFields behave strongly nonlinearly it is necessary to split the overall task into smaller steps that enable mathematical formulations where poor local minima can be avoided. The idea is to start with convex approximations that reliably lead to good initial solutions for the subsequent nonlinear steps. The following overview, which gives a preview to Section 6, clarifies how we split the optimization:

BendField Algorithm

(1) Estimate stroke constraints - Section 4
(2) Optimize 2D BendField - Eqn. (3)
  (i) optimize unit-length harmonic cross field - Eqn. (5)
  (ii) refine to free-length 2D BendField - Eqn. (6)
(3) Optimize 3D BendField - Eqn. (4)
  (i) local 3D estimate - Section 5.3
  (ii) refine to 3D BendField - Eqn. (4)
  (iii) compute normal field from $\mathbf{N} = \mathbf{U} \times \mathbf{V}$

6. NON-ORTHOGONAL 2D CROSS FIELDS

We now describe our representation of non-orthogonal 2D cross fields and their optimization subject to the user constraints. The goal of this step is to find a free-length non-orthogonal 2D cross field that minimizes the complicated nonlinear energy (3) while avoiding local minima. Therefore, we first generate a suitable initial guess by solving for a *unit-length* non-orthogonal cross field that minimizes a harmonic smoothness energy, which is convex up to integer variables. We adopt the established greedy strategy of [Bommes et al. 2009] to solve for the integer unknowns, which are then kept fixed during the subsequent nonlinear optimization. Notice that only the non-linear optimization is specifically tailored for our application, while the rest is a novel generalization of [Bommes et al. 2009] to non-orthogonal cross fields and as such useful for many other applications.



Vector representation. A unit-length non-orthogonal cross corresponds to four unit-length vectors $\mathbf{w}_0$, $\mathbf{w}_1$, $\mathbf{w}_2$ and $\mathbf{w}_3$ with the anti-symmetry conditions $\mathbf{w}_0 = -\mathbf{w}_2$ and $\mathbf{w}_1 = -\mathbf{w}_3$. Due to this anti-symmetry, an ordered tuple $[\mathbf{w}_0, \mathbf{w}_1] \in \mathbb{R}^{2\times 2}$ is locally sufficient to uniquely represent a non-orthogonal cross. However, in the presence of singularities a smooth cross field cannot be globally represented by two smooth vector fields as illustrated in Figure 8(a). In addition, the lines in the sketch are unoriented, which prevents us from knowing which of the four vectors they should locally constrain (Figure 8(b)).

In order to handle such cases, we split the surface into charts which are connected by integer *transition functions* $T_{i\to j}$

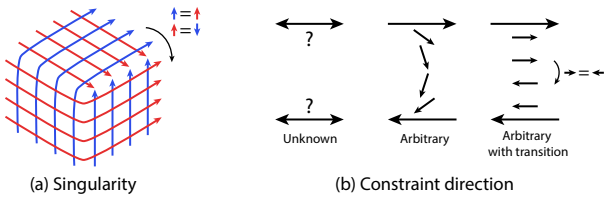


Fig. 8: In the presence of a singularity, a smooth cross field cannot be globally represented by two smooth vector fields (a). The transition function permutes the vectors to best align them. Similarly, assigning the constraints to an arbitrary direction can produce unnecessary variation in a smooth vector field (b), which can be prevented by optimizing the transition functions. Note that while we illustrate the constraint assignment on a vector field, the same principle extends to cross fields where each constraint needs to be assigned to one of four directions.

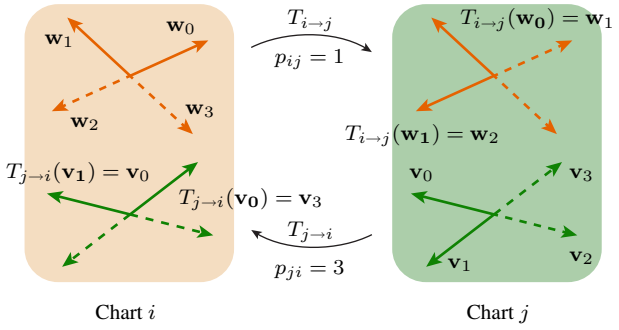


Fig. 9: The transition function permutes the vectors of a cross in one chart to align it with a cross in another chart. The integer variable p_{ij} encodes the number of permutations to align a cross from chart i to chart j.

[Ray et al. 2006; Bommes et al. 2009], which cyclically permute the four vector fields p_{ij} times when moving from chart i to chart j, as illustrated in Figure 9. Formally this means $T_{i\to j}([\mathbf{w}_0, \mathbf{w}_1]) = [\mathbf{w}_{p_{ij}}, \mathbf{w}_{p_{ij}+1}]$ with $p_{ij} \in \mathbb{Z}$ and $\mathbf{w}_i = \mathbf{w}_{i+4}$ for all $i \in \mathbb{Z}$.

If we want to measure the similarity of two crosses $[\mathbf{w}_0, \mathbf{w}_1]$ and $[\mathbf{v}_0, \mathbf{v}_1]$ that are expressed w.r.t. charts i and j respectively, we need to consider the corresponding transition function. A fixed transition function enables the following convex similarity measure based on the Frobenius norm of matrices

$$||T_{i\to j}([\mathbf{w}_0, \mathbf{w}_1]) - [\mathbf{v}_0, \mathbf{v}_1]||_2^2 = ||[\mathbf{w}_{p_{ij}} - \mathbf{v}_0, \mathbf{w}_{p_{ij+1}} - \mathbf{v}_1]||_2^2$$

Angle representation. For unit-length cross fields it is often preferable to optimize in the space of polar coordinates (r, φ), where the unit-length constraint is simply $r = 1$. Consequently, a cross can be uniquely represented by two angles $[\varphi_0, \varphi_1]$. However in case of transition functions it is advantageous to choose a different parametrization of the two angles. We express a cross by the tuple $[\alpha, \beta]$, which is related to φ by

$$\varphi_i = \alpha + (-1)^i \beta + i \cdot \frac{\pi}{2}$$

As illustrated in Figure 10, β describes the deviation from orthogonality while α can be understood as the closest orthogonal cross. One nice property is that for $\beta = 0$, our representation is exactly

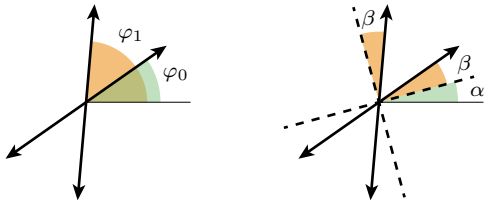


Fig. 10: A non-orthogonal cross can be represented by two angles $[\varphi_0, \varphi_1]$ in polar coordinates (a). To simplify formulas, we instead use one angle α to encode the global orientation of the cross (dashed lines) and one angle β to encode the deviation from orthogonality (b).

the same as the one used in [Ray et al. 2008; Bommes et al. 2009].

Similarly to the vector case we can define a smoothness measure between crosses $[\alpha_i, \beta_i]$ and $[\alpha_j, \beta_j]$ in different charts

$$\begin{aligned} E^{ij}_{\text{smooth}} &= ||T_{i\to j}[\varphi^i_0, \varphi^i_1] - [\varphi^j_0, \varphi^j_1]||^2 \\ &= ||[\varphi^i_{p_{ij}}, \varphi^i_{p_{ij}+1}] - [\varphi^j_0, \varphi^j_1]||^2 \\ &= (\alpha_i + (-1)^{p_{ij}}\beta_i + p_{ij}\frac{\pi}{2} - \alpha_j - \beta_j)^2 \\ &\quad +(\alpha_i - (-1)^{p_{ij}}\beta_i + p_{ij}\frac{\pi}{2} - \alpha_j + \beta_j)^2 \\ &= 2\left[(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2 + ((-1)^{p_{ij}}\beta_i - \beta_j)^2\right] \end{aligned}$$

Unit-length non-orthogonal cross fields in Images. In the image grid we assign one cross $[\alpha_i, \beta_i]$ per pixel p_i and assume that it is expressed w.r.t. its own chart C_i. We obtain a finite difference approximation of the harmonic energy

$$E_h = \int_I ||\nabla\varphi_0||^2 + ||\nabla\varphi_1||^2 dA$$

on the regular image grid by summing the smoothness measure over all pixels i

$$E_{\text{smooth}} = \sum_i \sum_{j\in\mathcal{N}_i} E^{ij}_{\text{smooth}}$$

with $\mathcal{N}_i$ containing the upper and right neighbors of pixel i. Initially all transition functions are unknown. Thus, for an image with $n = w \times h$ pixels, we end up with an optimization problem of $2n$ continuous variables ($\alpha, \beta \in \mathbb{R}$) and $2n - w - h - 4$ discrete variables ($p_{ij} \in \mathbb{Z}$).

Furthermore, for a subset of pixels $S_c \subset I$ we have stroke constraints θ that can be incorporated by means of a penalty energy

$$E_{\text{strokes}} = \sum_{i\in S_c} w_i((\alpha_i + \beta_i) - \theta_i)^2$$

with w_i being the weight of the constraint as described in Section 4. Notice that thanks to the transition functions we can always simply constrain $\varphi_0 = \alpha + \beta$ without worrying about the combinatorial relation between constraints (cf. Figure 8(b)). Depending on the application we want to penalize the deviation from orthogonality which in our representation is simply expressed as

$$E_\beta = \sum_i (\beta_i)^2.$$

Thus in total we optimize

$$E_{\text{angle}} = E_{\text{smooth}} + w_{\text{strokes}} E_{\text{strokes}} + w_\beta E_\beta. \quad (5)$$

In our application we use a very small weight $w_\beta = 10^{-6}$ to just regularize underdetermined cases, in combination with a weight $w_{\text{strokes}} = 1$ to equally balance smoothness and stroke constraints. Other applications such as quad remeshing may benefit from a higher w_β to favor orthogonal crosses away from constraints.

Greedy mixed-integer optimization. In order to efficiently find good solutions for the mixed-integer problem (5) we exploit the following three observations. First, if the integer transition variables p_{ij} are known, E^{ij}_{smooth} is a convex quadratic function. We exploit this by solving a series of quadratic problems which result from the slight modification

$$E_{\text{smooth}} = \sum_i \sum_{j \in \mathcal{N}_i} a_{ij} E^{ij}_{\text{smooth}}$$

with additional boolean variables $a_{ij} \in \{0, 1\}$. Such an a_{ij} is activated, i.e. $a_{ij} = 1$, only if the corresponding p_{ij} is a known constant, while otherwise $a_{ij} = 0$.

We fix the p_{ij} in a greedy order. For each non-activated term we estimate the activation cost as $A_{ij} = \min_{p_{ij}} E^{ij}_{\text{smooth}}$, where this time α and β are kept constant. The best candidate with the smallest cost $\arg\min_{ij} A_{ij}$ is then activated by setting $a_{ij} = 1$ and fixing the corresponding p_{ij}. Subsequently we update the current solution to capture the change due to the newly activated term. We iterate this process until all a_{ij} are activated, i.e. all transition functions are fixed.

Second, determining the p_{ij} that minimizes a A_{ij} can be done by explicitly checking two candidates. Investigating E^{ij}_{smooth} we see that the first term $(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2$ is minimized at $p^*_{ij} = \frac{2}{\pi}(\alpha_j - \alpha_i) \in \mathbb{R}$. Since the second term $((-1)^{p_{ij}}\beta_i - \beta_j)^2$ only changes depending on whether p_{ij} is even or odd, we conclude that the optimal value for p_{ij} can only be either $\lceil p^*_{ij} \rceil$ or $\lfloor p^*_{ij} \rfloor$. In the special case of $p^*_{ij} \in \mathbb{Z}$ it is sufficient to check the two candidates $\{p^*_{ij}, p^*_{ij} + 1\}$ since both integer neighbors of p^*_{ij} are equally good, i.e. $A_{ij}(p^*_{ij} + 1) = A_{ij}(p^*_{ij} - 1)$.

Third, the solution of an unconstrained pixel $i \notin S_c$ is underdetermined if there is no path of activated a_{ij} to one of the constraints. Therefore in order to obtain a unique minimizer, for each pixel we can arbitrarily activate a path of a_{ij} to its closest constraint by fixing the corresponding p_{ij}. This results in a forest, where the root of each spanning tree belongs to a constraint as shown in Figure 11(b). Notice that fixing $p_{ij} = 0$ at the spanning tree edges does not restrict the solution space but induces a good initialization for the subsequent greedy integer estimation (Figure 11(c)).

Figure 6(a) shows an example computation of the greedy mixed-integer optimization. We don't observe undesired singularities in the field, which is a good indicator that our approach effectively avoids local minima. However, the cross field tends to "flatten" away from the strokes, which is a result of optimizing a harmonic energy. We next describe how to optimize for the desired nonlinear energy E_{bend2D} which better mimics the behavior of 3D curvature lines.

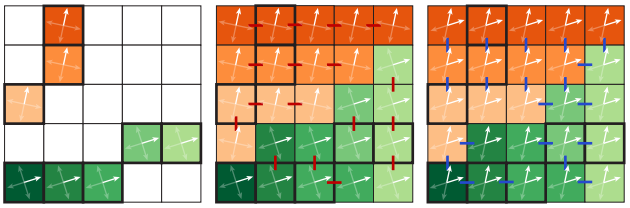


Fig. 11: Main steps of the greedy optimization. Each stroke provides constraints on one of the two representative vectors of a cross (a, outlined pixels). We initialize the cross of each unconstrained pixel to the cross of the closest constraint (b, colors correspond to the closest constraint). We also set $p_{ij} = 0$ between pixels initialized with the same constraint (b, red links). The greedy optimization solves for the remaining p_{ij} (c, blue links) and updates the solution subject to the smoothness energy.

Nonlinear optimization. Since the covariant derivative of E_{bend2D} has a mathematically better behaved expression in vector coefficients, we now switch from the angle representation (α, β) to the vector representation $(\mathbf{u}, \mathbf{v})$. The solution of E_{angle} is used as a starting point for the nonlinear optimization of E_{bend2D}. This refinement is of geometric nature and does not require topological changes. Therefore, we keep the transition functions fixed when discretizing the covariant derivatives with finite differences. The resulting functional can be optimized by a standard Newton method. However, we observed that iterating the optimization of the quadratic approximation

$$(\mathbf{u}^{(i+1)}, \mathbf{v}^{(i+1)}) \leftarrow \min_{u,v} \int_I ||\nabla_{\mathbf{u}} \mathbf{v}^{(i)}||^2 + ||\nabla_{\mathbf{v}} \mathbf{u}^{(i)}||^2 dA$$

where $(\mathbf{u}^{(i)}, \mathbf{v}^{(i)})$ is the solution of iteration i, is sufficient and converges even faster. During this optimization the alignment to strokes is maintained by an additional penalty energy E_{strokes}. We also found that because our smoothness measure is strongly directional, areas away from the flow induced by the constraints can become unstable. We easily cope with such situations by adding a weak harmonic regularization $\epsilon_h E_h$ to the functional, resulting in the following energy, which is iteratively optimized

$$E_{\text{vector}} = E_{\text{bend2D}} + w_{\text{strokes}} E_{\text{strokes}} + \epsilon_h E_h \quad (6)$$

where E_{strokes} and E_h are now expressed in vector coefficients instead of angles. Specifically,

$$E_{strokes} = \sum_{i \in S_c} w_i ||\mathbf{u} - (\cos\theta_i, \sin\theta_i)||^2$$

$$E_h = \int_I ||\nabla \mathbf{u}||^2 + ||\nabla \mathbf{v}||^2 dA.$$

We used $\epsilon_h = 0.1$ for all our results and adjust w_{strokes} according to the sketchiness of the drawing, as discussed in Section 9. In practice we use the binary mask provided as input to optimize Equation 6 only within the region of interest. On the border of the mask we set Neumann boundary conditions $\nabla \mathbf{u} = 0$ and $\nabla \mathbf{v} = 0$. Interestingly, restricting to the mask does not change the result in the region of interest but significantly speeds up the overall computation. This happens because the optimizer converges only slowly in the unimportant boundary regions which are far away from sketch constraints, when applied to the whole image.

Figure 6c shows that our nonlinear energy effectively improves the curvature lines by inflating the surface parts that appeared to be flat before. This Figure also shows the normal field we obtain after optimizing E_{bend3D}. Figure 12 shows the behavior of our method on the sketch of a non-quad patch. The algorithm generates a singularity in the middle of the patch to form a cubic corner.

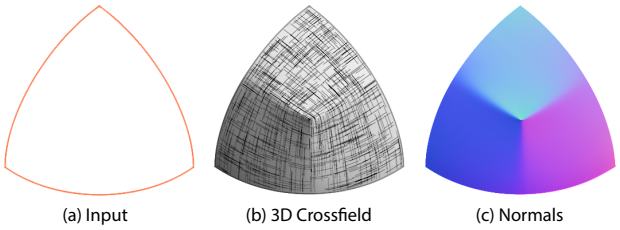


Fig. 12: This drawing of a non-quad surface patch results in a singularity in the cross field. The corresponding normal field forms a cubic corner.

7. RELATION TO PREVIOUS CROSS FIELD APPROACHES

In this section we clarify the relation of our cross field approach w.r.t. other methods.

Comparison to [Bommes et al. 2009; 2012]. Our unit-length harmonic optimization is a generalization of [Bommes et al. 2009; 2012], which is restricted to orthogonal cross fields and relies on continuous relaxation with iterative rounding to solve for integer variables. Nevertheless, our greedy optimizer proceeds in a similar way. In the following we show that for the orthogonal case of $\beta = 0$, the algorithms are identical. The additional difficulty in our case is that the integers p_{ij} contribute nonlinearly due to expressions $(-1)^{p_{ij}}$, which additionally make continuous relaxation impossible without switching to complex numbers. Thus, instead of optimizing the continuous relaxation, we deactivate all terms with unknown p_{ij}. At a first glance this appears suboptimal compared to the continuous relaxation. However, by considering that in the orthogonal case each p_{ij} contributes only to one term $(\alpha_i + p_{ij}\frac{\pi}{2} - \alpha_j)^2$, we see that the continuously relaxed terms also always vanish by the choice $p_{ij} = 2/\pi(\alpha_j - \alpha_i)$. Thus, for $\beta = 0$ our greedy approach based on activation variables is identical to the relaxation method of [Bommes et al. 2009; 2012]. Consequently we also benefit from all performance optimizations that were proposed in [Bommes et al. 2012], including a hierarchy of solvers and multiple activations.

Comparison to CDFs. Since the handling of non-orthogonal cross fields in the context of conjugate direction fields (CDFs) [Liu et al. 2011] is related to our approach, we discuss the differences in more detail. Similarly to us, Liu et al. use an angle-based representation (θ, α), which in our notation corresponds to $\theta = \varphi_0$ and $\alpha = \varphi_1 - \varphi_0$. The transition functions are handled by $p_1, p_2 \in \mathbb{Z}$ and $q \in \{0, 1\}$ while in our case a single $p \in \mathbb{Z}$ is sufficient. Apart from these notational subtleties, which mostly affect the simplicity of formulas, the most important difference is the chosen smoothness measure and the corresponding optimization strategy. The smoothness measure of [Liu et al. 2011] exploits the periodicity of the cos function to eliminate all integer degrees of freedom (DOFs) and it is shown that the resulting functional is a generalization of the one proposed in [Hertzmann and Zorin 2000] for orthogonal cross fields. As discussed in [Liu et al. 2011] these nonlinear functionals induce a tendency to end up in local minima with unsatisfactory additional singularities (cf. Figure 4 in [Liu et al. 2011]), even in case of a good initialization.

On the contrary our smoothness measure contains integer DOFs and is a generalization of Bommes et al. [2009]. Since every step in the greedy integer estimation solves a simple linear problem, similarly to [Bommes et al. 2009], unsatisfactory additional singularities are effectively prevented. Consequently, we believe that our novel non-orthogonal cross field representation and optimization is a valuable general tool with numerous applications apart from concept sketching, such as surface meshing.

8. ADDITIONAL DETAILS

User interface. The accompanying video illustrates a typical interactive session with our tool. Users first load an existing bitmap sketch and its mask and paint over curvature and discontinuity strokes in different colors. Design literature explains that "cross-sections on a surface explain or emphasize its curvature" [Eissen and Steur 2008], which suggests that designers know the difference between lines that convey curvature and other discontinuity lines. In addition to the stroke annotations, we also ask users to select one of the two possible consistent solutions of each surface patch normal field (Section 5.3, Figure 13). We implemented these user indications as unary constraints in the labeling problem (Appendix). In practice, several indications are sometimes necessary to obtain a consistent result over complex patches. Finally, we also provide users the ability to combine layers when sketching complex objects made of independent parts.

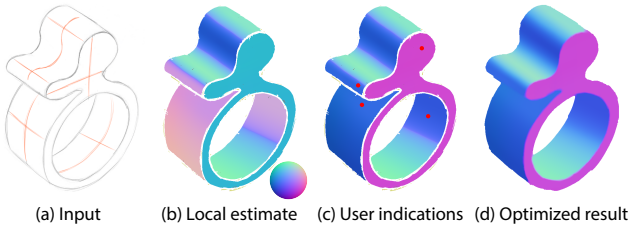


Fig. 13: Curvature lines can be interpreted as running over a convex or a concave surface patch. Based on our local guess (b), the user clicks on inconsistent patches (c, red dots) to flip their orientation. The local guess is then refined by the 3D BendField energy (d).

Pixels on discontinuity strokes do not constrain the orientation and smoothness of the cross field and are also not considered when solving for the globally consistent 3D solution. As a result, regions bounded by discontinuity strokes form isolated patches in the solution and discontinuity strokes do not receive values. We assign normals to discontinuity pixels as a post-process by diffusing nearby normals [Winnemöller et al. 2009].

Sketch pre-processing. We apply a few simple image processing operations on the input image before running our algorithm. We found that applying a small Gaussian blur to remove noise produces a more accurate estimation of local orientations. We also observe that artists draw strokes with varying strength, darker strokes denoting more confidence. We apply a permissive threshold to select dark and lighter strokes (0.8 in our implementation). The constraint

weight w_i then implicitly accounts for the strength of the strokes as darker ones have a smaller minor eigenvalue.

Texture Mapping. For texture mapping we adapt the parametrization step of [Bommes et al. 2009] to non-orthogonal cross fields with anisotropic sizing. In order to get the desired parametrization (s, t) of our surface we optimize

$$E_{\text{param}} = \int_I \left([\nabla s, \nabla t]^T [\mathbf{u}, \mathbf{v}] - I_2 \right)^2 dA$$

with I_2 being the two dimensional identity matrix and

$$[\nabla s, \nabla t]^T = \begin{pmatrix} \partial s/\partial x & \partial s/\partial y \\ \partial t/\partial x & \partial t/\partial y \end{pmatrix}, [\mathbf{u}, \mathbf{v}] = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix}$$

being the differential which transforms vectors from image space to texture space. Intuitively this energy states how well the given $\mathbf{u}, \mathbf{v}$ vectors map to the Cartesian s, t axes in texture space. Thus the inverse mapping tries to aligns the texture with the non-orthogonal cross field. The length distortion due to foreshortening is encoded into the length of $\mathbf{u}$ and $\mathbf{v}$. While our cross field is not guaranteed to be integrable, we didn't observe significant distortions of the parameterization in practice.

Optimization. For the greedy mixed-integer optimization we use a hierarchy of solvers, i.e. local Gauss Seidel, Conjugate Gradient and Sparse Cholesky, as explained in [Bommes et al. 2012]. The Conjugate Gradient and Sparse Cholesky is taken from Eigen3 [Guennebaud et al. 2010], which is also used for the optimization of all quadratic energy functionals. For the nonlinear optimization of Section 5.3 we apply the interior point method of IPOPT [Wächter and Biegler 2006]. We provide source code as supplemental materials to facilitate reproduction.

Table I details the time spent by our implementation on each step of the optimization, for two sketches. The current bottleneck resides in the nonlinear optimization of the 2D and 3D BendField energies (Equations 4 and 6). However, we note that our cross fields are piece-wise smooth, which makes our problem an ideal candidate for hierarchical algorithms like multigrid [Briggs et al. 2000], although care should be taken to properly handle transition functions between levels of the hierarchy [Bommes et al. 2013].

Sketch	Resolution	Harmonic	2D BendField	3D BendField
Kettle	900×800	1 min. 30 sec.	32 min.	9 min.
Vacuum	900×700	2 min. 17 sec.	21 min.	24 min.

Table I. : Detailed timing for the main steps of our method for a simple sketch (water kettle, Figure 17) and a complex one (vacuum cleaner, Figure 1). We used 10 iterations to optimize the 2D BendField with Eqn. (6), although we observed that the energy decreases quickly during the first 3 iterations, then converges to a plateau value.

9. RESULTS AND EVALUATION

Figure 1, 3, 13, 16, 17, 21, 20 illustrate the results of our method on a range of concept sketches. A major advantage of our method is its ability to process existing sketches. We produced all our inputs by reproducing drawings from design books and websites that are representative of the distortions and inaccuracy found in real sketches. We reproduced these drawings to avoid copyright issues and to remove decorative lines (cross-hatching and texture) that users would not draw in our context. We provide links to the original drawings as supplemental material. In theory, our algorithm requires at least two intersecting curvature lines per isolated surface patch, although users can draw more lines to refine bending over complex surfaces.

We visualize our cross-fields with hatching [Hertzmann and Zorin 2000] and provide a color-coded visualization of the surface normals estimated by our algorithm. The cross-fields and normals provide a vivid sense of the 3D shapes depicted by the sketches, capturing the overall orientation of the surfaces as well as subtle bending, such as the folding shape of the stool (Figure 3) and the wavy handle of the bag (Figure 21). Figure 21 additionally shows the use of our normals and cross-fields for shading and texture mapping. Figure 14 shows how a harmonic smoothness energy does not capture surface bending as well as our BendField energy.

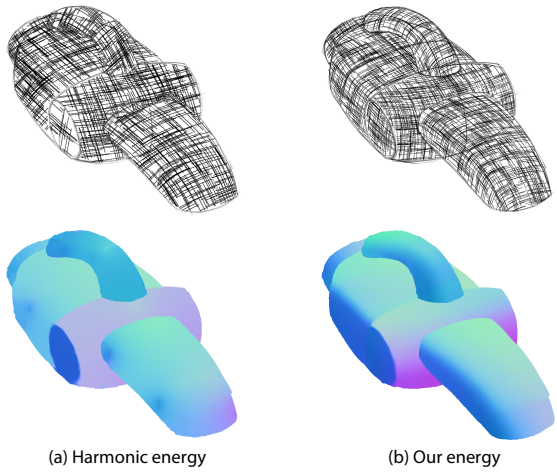


Fig. 14: Comparison between a harmonic energy and our energy on the same input as Figure 1. A harmonic energy tends to flatten the shape away from user strokes, producing "tent" artifacts at stroke intersections (a). Our energy better captures the bending of the surface (b). We used the same weights for the two versions of the algorithm.

Limitations. Because our approach works on bitmaps, the accuracy of the result is dependent of the image resolution. Fine details, such as the thin legs of the chair in Figure 21, are not well captured by the structure tensor. While we added these details with decorative strokes, an alternative would have been to sketch them at higher resolution in a separate layer and then composite them in the final image.

Our algorithm infers the directions of the cross-field from the local orientation of the strokes. As a result, our approach sometimes fails to distinguish a strongly foreshortened crossing between two lines from a bend on a single line, since the two configurations locally form a similar wide angle. In theory, our algorithm will always interpret intersecting lines as two different directions if they deviate by less than 45° from orthogonality, while greater deviations can be compensated for by the transition function and be interpreted as a bending over a single direction. In practice the global configuration of the cross-field can sometimes result in angles that deviate more than this lower bound. Figure 15(a) illustrates the behavior of our algorithm on a strongly foreshortened cube, where the curvature lines on the top face are interpreted as constraints on one of the two directions of the cross-field, the other direction being free. Figure 15(b) shows a cube from a less foreshortened view

where the crossing lines form an angle closer to 90° and as such are better captured by our approach. Fortunately, designers are trained to draw objects from *informative viewpoints* that minimize foreshortening over most surfaces [Eissen and Steur 2011; Shao et al. 2012], as demonstrated by our results over typical sketches.

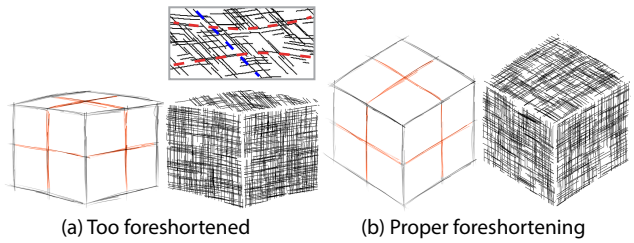


Fig. 15: In the presence of strong foreshortening, such as on the top face of the cube in (a), intersecting curvature lines form the same local configuration as overlapping strokes on a curvy line. As a result, our algorithm interpret these lines as constraints on only one of the two directions of the cross-field. Designers usually avoid strong foreshortening to maximize information, providing proper constraints for our algorithm (b).

Effect of parameters. Figure 16 illustrates the effect of the two main parameters of our algorithm. These parameters offer a trade-off between fidelity to the input strokes and smoothness of the solution. In particular, sketchy lines can result in wiggles in the normal field, which can be removed by increasing the spatial extent σ_s of the bilateral filter during orientation estimation (Section 4) and by reducing the constraint weight w_{strokes} in Equation 6. However, too much filtering can result in a loss of detail, while too much smoothing tends to flatten the surface. We used the same preset of $\sigma_s = 13$ and $w_{\text{strokes}} = 0.25$ for most sketches, except for the very sketchy drawings (Figure 3 and 17) for which we use $\sigma_s = 23$ and the clean CrossShade curves (Figure 20) for which we used $w_{\text{strokes}} = 0.1$.

Robustness to sketchy lines. We designed our method to be robust to the sketchy lines typical of concept drawings. Figure 17 evaluates this robustness on four versions of the same sketch, with an increasing density of strokes. Our method produces similar cross-fields and normals for the various levels of sketchiness, although fine details are lost for very sketchy drawings. While our scattered interpolation handles sparse and incomplete curvature constraints, holes in discontinuity lines can result in smooth transitions across surface patches.

Comparison to ground truth. We derived our BendField energy from properties of curvature lines. Figure 18 compares our interpolated cross fields to ground truth curvature lines generated from 3D surfaces. We chose these surfaces to have no umbilical points and to be perfect minimizers of the BendFields energy since their lines of curvature are also geodesics. We applied our complete pipeline to the rasterised sparse curvature lines using a small stroke smoothness $\sigma_s = 3$ and $w_{\text{strokes}} = 0.1$.

Our cross-field closely matches the projected curvature lines with a mean error of 2.49 degrees on the directions (standard deviation of 4.52 degrees), resulting in visually similar normals with a mean error of less than 1.12 degrees (standard deviation of 2.09 degrees). Small wiggles are noticeable in the normal field, which can be removed by increasing σ_s at the cost of flattening the shape.

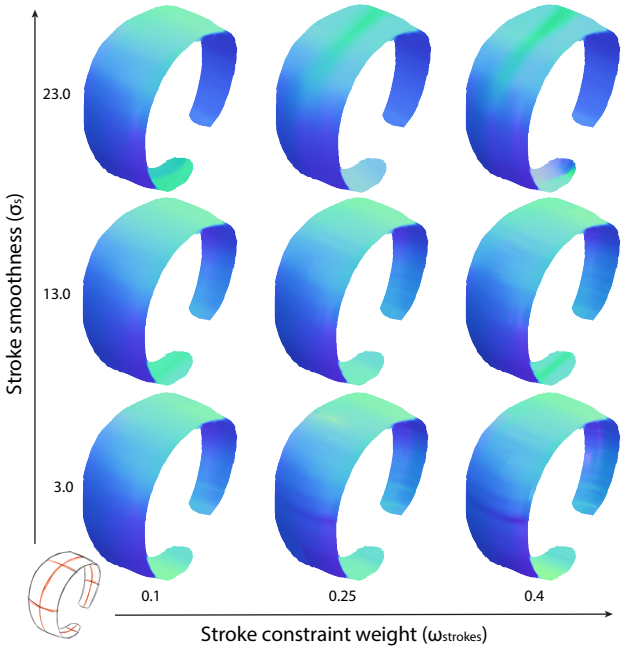


Fig. 16: The stroke smoothness and constrain weights offer a trade-off between fidelity to the input drawing and smoothness of the normal field. While a range of parameters produce similar results, too much smoothing removes details and flatten the surface (top left corner), while too strong constraints produce wiggles because of the sketchy strokes (right column).

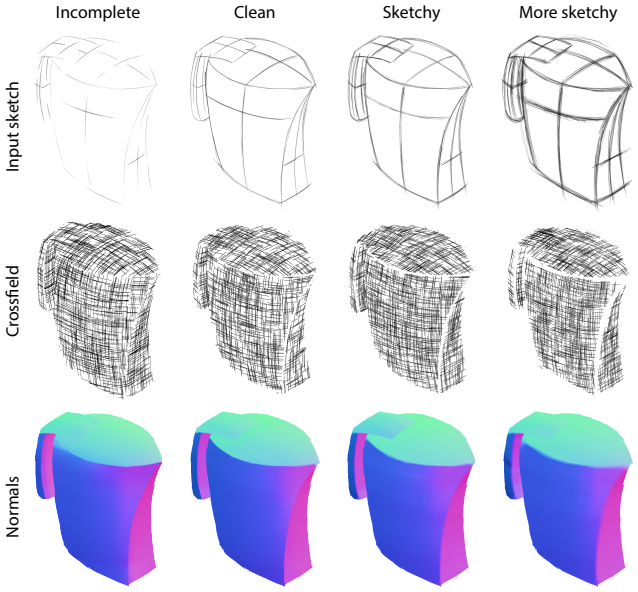


Fig. 17: Our approach is robust to different levels of sketchiness, from sparse strokes with holes (left) to many overlapping strokes (right). Despite these drastic differences in input style, our method produces consistent cross-fields and normals.

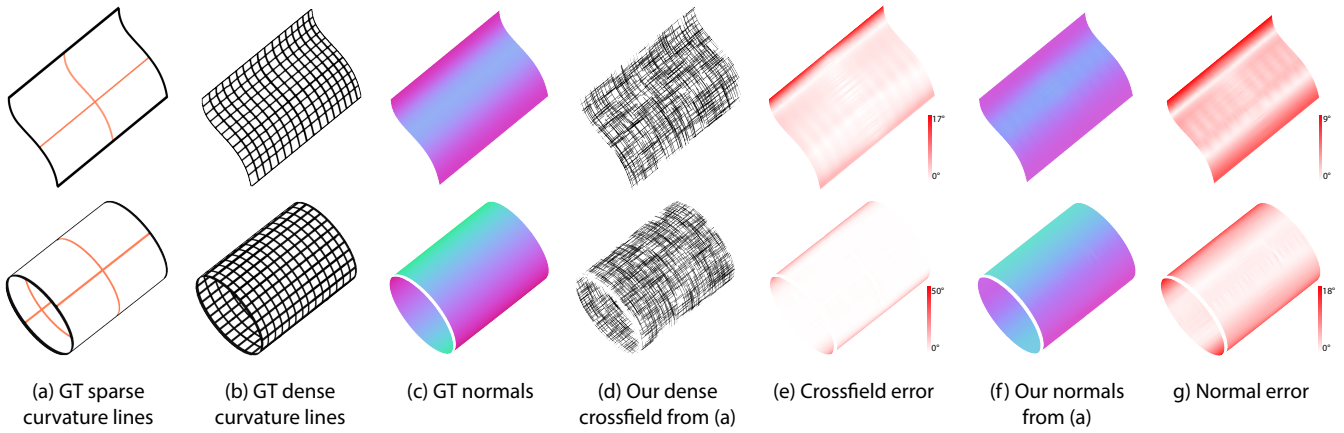


Fig. 18: Comparison with ground truth curvature lines and normals. Our cross field aligns with the ground truth dense curvature field (b,d), producing normals that closely match the ones of the ground truth surface (c,e). Our cross field and normals are less accurate near boundaries and silhouettes where junctions make the estimation of orientation of curvature lines less accurate and regularization penalizes strong foreshortening and non-orthogonality. The cylinder has a mean error on cross field directions of 2.86° and standard deviation of 5.43° and on normals of 1.26° and standard deviation of 2.4°, the wave has a mean error of 2.00° and standard deviation of 2.43° on directions and of 0.85° and standard deviation of 1.64° on normals.

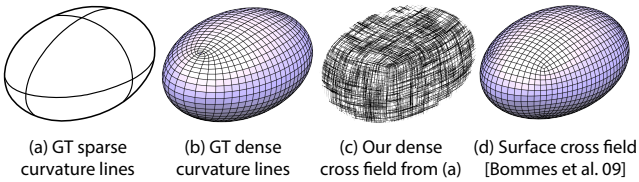


Fig. 19: Comparison with ground truth curvature lines in the presence of umbilical points. Our regularized cross field deviates from the non-geodesic curvature lines (b,c) and positions singularities similarly to the surface cross field of [Bommes et al. 2009] (d).

We provide as supplemental materials the results of the same experiment using ground truth 2D constraints as input to bypass the initial estimation of stroke orientation. The errors in this experiment are lower than when running the complete pipeline, yet distributed similarly with a mean error of 1.87 degrees on the cross field directions (standard deviation of 3.94 degrees) and 1.06 degrees on the normals (standard deviation of 1.97 degrees). Most errors occur near discontinuity lines and silhouettes where the estimation of local orientation is less accurate and our regularizers penalize strong foreshortening and non-orthogonal crosses (Equation 4 and 5). An interesting direction for future research would consist in combining our approach with inflation methods [Johnston 2002] in order to leverage both the 3D cues provided by curvature lines and smooth silhouettes.

Figure 19 provides an evaluation against a more complex surface with umbilical points. Our regularized cross field positions a singularity in the center of the triangular face of the sketch, while the ground truth singular points lie on the great circles of the ellipsoid. Figure 19(d) shows that our algorithm actually behaves similarly to the surface cross field algorithm of Bommes et al. [2009], which favors smooth geodesic curves away from regions with high anisotropic curvature.

Comparison to prior work. Figure 20 provides a comparison of our normal fields and shading with the ones generated by the CrossShade algorithm [Shao et al. 2012]. While both algorithms estimate normals by leveraging orthogonality of curvature lines, they target different input and perform data interpolation in a different order. CrossShade requires clean vectorial curves as input, which provide a high degree of precision and smoothness. In contrast, our method processes bitmap drawings with a finite resolution and sketchy lines. From an algorithmic point of view, CrossShade estimates normals solely at curve intersections and propagates these estimates along and in-between curves using parametric Coons patches. Our algorithm operates in a different order, first applying scattered interpolation on the strokes to form a dense curvature field and then estimating normals at each pixel. CrossShade also enforces planar cross-section curves, which our local formulation cannot do. In practice, our algorithm tends to produce a flatter result near boundaries of curved surface patches, such as on the lens of the camera in Figure 20, where the junctions make the orientation estimation less reliable (Section 4). We plan to explore the estimation of two directions near junctions to address this issue [Aach et al. 2006]. Nevertheless, our approach produces results visually similar to CrossShade, without requiring users to be familiar with vector drawing tools.

10. CONCLUSION AND FUTURE WORK

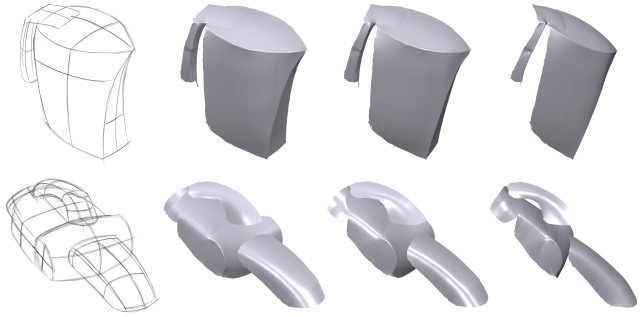
Sketch-based modeling systems traditionally take clean vectorial curves as input. In this paper we have explored an alternative approach by extrapolating curvature lines from bitmap drawings. Our approach relies on a scattered-data interpolation to be robust to the rough drawings common in concept sketching. The resulting 2.5D cross fields, which we call Bend Fields, allow a range of sketch-editing applications originally developed for 3D surfaces, such as local shading, texture mapping and cross-hatching.

Our algorithm relies on two technical contributions, the formulation of a suitable smoothness energy that captures the behavior of curvature fields, and a non-orthogonal cross field represen-

(a) Rasterized CrossShade curves (b) Our normals (c) Our shading (d) CrossShade normals (e) CrossShade shading

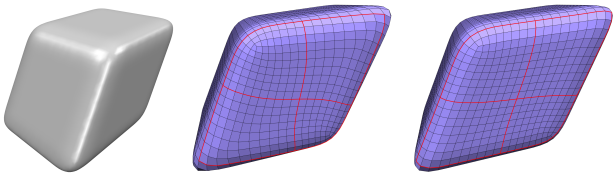
Fig. 20: Comparison with CrossShade [Shao et al. 2012]. Our method produces qualitatively similar results without the need for vectorial curves. Note that we rasterized the CrossShade curves as polylines and did no attempt to remove extraneous dangling segments at extremities.

tation. We foresee numerous applications of these two contributions in other research domains. We first plan to explore the use of our approach for 3D sketch-based modeling. While our normal-fields can be integrated to form height-fields (Figure 22), the resulting surface is often distorted due to perspective inaccuracy in sketches [Schmidt et al. 2009], which would require special treatment, for instance by detecting and enforcing regularization constraints over the cross field [Xu et al. 2014]. Our smoothness energy could also represent a powerful regularizer for shape-from-shading, shape-from-texture and shape-from specularity algorithms [Tappen 2011]. Finally, our non-orthogonal cross field representation can also contribute to quad-meshing algorithms in applications where the orthogonality constraint is not desirable, as shown in Figure 23. For such applications, an interesting direction for future work would be to add two length variables to each cross in order to support variable-length fields in the optimization.



(a) Input (b) Heightfield from different viewpoints

Fig. 22: Our normal-fields can be integrated to form height-fields [Nehab et al. 2005]. Here we reconstructed each layer separately and positioned them in depth manually. While the resulting surfaces suggest the potential of our approach for 3D sketch-based modeling, additional work is needed to deal with hidden parts of the model and to correct for distortions due to perspective and sketch inaccuracy.



(a) Input shape (b) Orthogonal mesh (c) Non-orthogonal mesh

Fig. 23: Application to geometry processing. We used our generalization of [Bommes et al. 2009] to generate a non-orthogonal quad mesh over this skewed cube (c). Our approach better captures the skewness of the shape compared to orthogonal quads (b), resulting in smoother flowlines.

Acknowledgments

We thank the authors of [Noris et al. 2013] and [Bartolo et al. 2007] for running their algorithms on our sketches. We also thank Mathieu Desbrun for feedback and Pascal Barla for discussions and help on rendering. Finally, we are grateful to Spencer Nugent for letting us use his inspiring sketches.

This work was supported by ANR-12-JS02-003-01 DRAO, ERC Starting Grant Robust Geometry Processing (Grant agreement 257474), DFG grant GSC 111 (Aachen Institute for Advanced Study in Computational Engineering Science) and software and research donations from Adobe.

REFERENCES

AACH, T., MOTA, C., STUKE, I., MUHLICH, M., AND BARTH, E. 2006. Analysis of superimposed oriented patterns. *IEEE Trans. on Image Processing 15,* 12, 3690–3700.

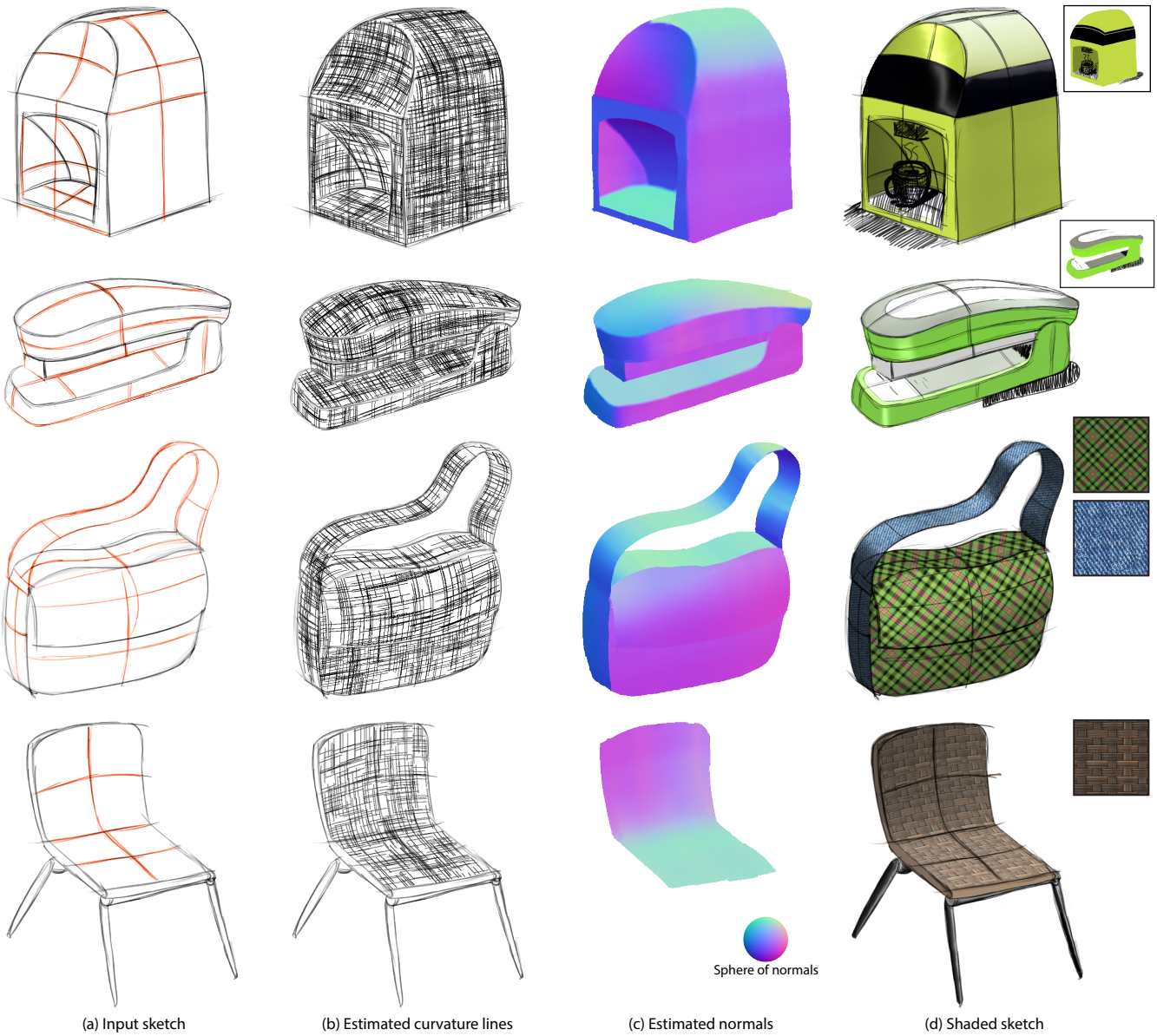


Fig. 21: Cross-fields and normals generated by our method for a coffee machine, a stapler, a bag and a chair. The bag and chair also show the use of texture mapping.

ALLIEZ, P., COHEN-STEINER, D., DEVILLERS, O., LÉVY, B., AND DESBRUN, M. 2003. Anisotropic polygonal remeshing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 22,* 3.

BAE, S., BALAKRISHNAN, R., AND SINGH, K. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proc. User Interface Software and Technology (UIST)*.

BARTOLO, A., CAMILLERI, K. P., FABRI, S. G., BORG, J. C., AND FARRUGIA, P. J. 2007. Scribbles to vectors: Preparation of scribble drawings for cad interpretation. In *Proc. of Eurographics Workshop on Sketch-based Interfaces and Modeling (SBIM)*. ACM, 123–130.

BESSMELTSEV, M., WANG, C., SHEFFER, A., AND SINGH, K. 2012. Design-driven quadrangulation of closed 3d curves. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31,* 6, 178.

BIARD, L., FAROUKI, R. T., AND SZAFRAN, N. 2010. Construction of rational surface patches bounded by lines of curvature. *Computer Aided Geometric Design 27*, 359–371.

BOMMES, D., CAMPEN, M., EBKE, H.-C., ALLIEZ, P., AND KOBBELT, L. 2013. Integer-grid maps for reliable quad meshing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32,* 4 (July), 98:1–98:12.

BOMMES, D., LÉVY, B., PIETRONI, N., PUPPO, E., SILVA, C., TARINI, M., AND ZORIN, D. 2013. Quad-Mesh Generation and Processing: A Survey. *Computer Graphics Forum 32,* 6, 51–76.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2009. Mixed-integer quadrangulation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28,* 3,

77.

BOMMES, D., ZIMMER, H., AND KOBBELT, L. 2012. Practical mixed-integer optimization for geometry processing. In *Proceedings of the 7th international conference on Curves and Surfaces*. Springer-Verlag, Berlin, Heidelberg, 193–206.

BRIGGS, W. L., HENSON, V. E., AND MCCORMICK, S. F. 2000. *A multigrid tutorial (2nd ed.)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

COLE, F., ISOLA, P., FREEMAN, W. T., DURAND, F., AND ADELSON, E. H. 2012. Shapecollage: Occlusion-aware, example-based shape interpretation. In *European Conference on Computer Vision (ECCV)*. Springer, 665–678.

DIAMANTI, O., VAXMAN, A., PANOZZO, D., AND SORKINE-HORNUNG, O. 2014. Designing n-PolyVector fields with complex polynomials. *Computer Graphics Forum (proceedings of EUROGRAPHICS Symposium on Geometry Processing) 33,* 5.

DO CARMO, M. P. 1976. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Englewood Cliffs, NJ.

EISSEN, K. AND STEUR, R. 2008. *Sketching: Drawing Techniques for Product Designers*. Bis Publishers.

EISSEN, K. AND STEUR, R. 2011. *Sketching: The Basics*. Bis Publishers.

FARIN, G. AND HANSFORD, D. 1999. Discrete coons patches. *Computer Aided Geometric Design 16*, 691–700.

FISHER, M., SCHRÖDER, P., DESBRUN, M., AND HOPPE, H. 2007. Design of tangent vector fields. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26,* 3 (July).

FREEMAN, W. T. AND ADELSON, E. H. 1991. The design and use of steerable filters. *IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI) 13*, 891–906.

GUENNEBAUD, G., JACOB, B., ET AL. 2010. Eigen v3. http://eigen.tuxfamily.org.

HAEBERLI, P. 1990. Paint by numbers: Abstract image representations. *SIGGRAPH 24,* 4.

HARRIS, C. AND STEPHENS, M. 1988. A Combined Corner and Edge Detection. In *Proceedings of The Fourth Alvey Vision Conference*.

HERTZMANN, A. AND ZORIN, D. 2000. Illustrating smooth surfaces. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 517–526.

IGARASHI, T., MATSUOKA, S., AND TANAKA, H. 1999. Teddy: A sketching interface for 3d freeform design. *SIGGRAPH*, 409–416.

JOHNSTON, S. F. 2002. Lumo: illumination for cel animation. In *Proc. Symp. on Non-Photorealistic Animation and Rendering (NPAR)*.

JOSHI, P. AND CARR, N. A. 2008. Repoussé: Automatic inflation of 2d artwork. In *Proc. of Sketch Based Interfaces and Modeling (SBIM)*.

KÄLBERER, F., NIESER, M., AND POLTHIER, K. 2007. Quadcover - surface parameterization using branched coverings. *Computer Graphics Forum 26,* 3 (Sept.), 375–384.

KANG, H., LEE, S., AND CHUI, C. K. 2007. Coherent line drawing. In *ACM Symp. on Non-photorealistic Animation and Rendering (NPAR)*. 43–50.

KANG, H., LEE, S., AND CHUI, C. K. 2009. Flow-based image abstraction. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 15,* 1, 62–76.

KASS, M. AND WITKIN, A. 1987. Analyzing oriented patterns. *Computer vision, graphics, and image processing 37,* 3, 362–385.

KNILL, D. C. 1992. Perception of surface contours and surface shape: from computation to psychophysics. *Journal of Optical Society of America 9,* 9, 1449–1464.

KNÖPPEL, F., CRANE, K., PINKALL, U., AND SCHRÖDER, P. 2013. Globally optimal direction fields. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32,* 4, 59.

KOLMOGOROV, V. 2006. Convergent tree-reweighted message passing for energy minimization. *IEEE Trans. Pattern Analysis Machine Intelligence (PAMI) 28,* 10 (Oct.), 1568–1583.

KYPRIANIDIS, J. E. AND KANG, H. 2011. Image and video abstraction by coherence-enhancing filtering. *Computer Graphics Forum 30,* 2, 593–602. Proceedings Eurographics 2011.

LEFEBVRE, S. AND HOPPE, H. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25,* 3.

LIU, Y., XU, W., WANG, J., ZHU, L., GUO, B., CHEN, F., AND WANG, G. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 30,* 6, 140.

LOPEZ-MORENO, J., POPOV, S., BOUSSEAU, A., AGRAWALA, M., AND DRETTAKIS, G. 2013. Depicting stylized materials with vector shade trees. *ACM Transactions on Graphics (Proc. SIGGRAPH) 32,* 4.

MAMASSIAN, P. AND LANDY, M. S. 1998. Observer biases in the 3D interpretation of line drawings. *Vision research 38,* 18, 2817–2832.

NEALEN, A., IGARASHI, T., SORKINE, O., AND ALEXA, M. 2007. Fibermesh: Designing freeform surfaces with 3d curves. *ACM Transactions on Graphics (Proc. SIGGRAPH) 26,* 3 (July).

NEHAB, D., RUSINKIEWICZ, S., DAVIS, J., AND RAMAMOORTHI, R. 2005. Efficiently combining positions and normals for precise 3d geometry. *ACM Transactions on Graphics (Proc. SIGGRAPH) 24,* 3 (Aug.).

NORIS, G., HORNUNG, A., SUMNER, R. W., SIMMONS, M., AND GROSS, M. 2013. Topology-driven vectorization of clean line drawings. *ACM Transactions on Graphics 32,* 1, 4.

ORBAY, G. AND KARA, L. B. 2011. Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Trans. on Visualization and Computer Graphics (TVCG) 17,* 5, 694–708.

PANOZZO, D., PUPPO, E., TARINI, M., AND SORKINE-HORNUNG, O. 2014. Frame fields: Anisotropic and non-orthogonal cross fields. *ACM Transactions on Graphics (Proc. SIGGRAPH) 33,* 4.

PIPES, A. 2007. *Drawing for Designers*. Laurence King.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. *SIGGRAPH*.

QU, Y., WONG, T.-T., AND HENG, P.-A. 2006. Manga colorization. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25,* 3 (July).

RAY, N., LI, W. C., LÉVY, B., SHEFFER, A., AND ALLIEZ, P. 2006. Periodic global parameterization. *ACM Transactions on Graphics 25,* 4, 1460–1485.

RAY, N., VALLET, B., ALONSO, L., AND LEVY, B. 2009. Geometry-aware direction field processing. *ACM Transactions on Graphics 29,* 1, 1.

RAY, N., VALLET, B., LI, W. C., AND LÉVY, B. 2008. N-symmetry direction field design. *ACM Transactions on Graphics 27,* 2.

SCHMIDT, R., KHAN, A., KURTENBACH, G., AND SINGH, K. 2009. On expert performance in 3D curve-drawing tasks. In *Proc. Symp. Sketch-Based Interfaces and Modeling (SBIM)*.

SHAO, C., BOUSSEAU, A., SHEFFER, A., AND SINGH, K. 2012. Crossshade: shading concept sketches using cross-section curves. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31,* 4, 45.

STEVENS, K. A. 1981. The visual interpretation of surface contours. *Artificial Intelligence 17,* 1, 47–73.

SÝKORA, D., BEN-CHEN, M., ČADÍK, M., WHITED, B., AND SIMMONS, M. 2011. Textoons: Practical texture mapping for hand-drawn cartoon animations. In *Proc. Symp. on Non-photorealistic Animation and Rendering (NPAR)*. 75–83.

SÝKORA, D., DINGLIANA, J., AND COLLINS, S. 2009. LazyBrush: Flexible painting tool for hand-drawn cartoons. *Computer Graphics Forum (Proc. EUROGRAPHICS) 28,* 2.

SÝKORA, D., KAVAN, L., ČADÍK, M., JAMRIŠKA, O., JACOBSON, A., WHITED, B., SIMMONS, M., AND SORKINE-HORNUNG, O. 2014. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transaction on Graphics 33*.

TAPPEN, M. F. 2011. Recovering shape from a single image of a mirrored surface from curvature constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Washington, DC, USA, 2545–2552.

VERGNE, R., BARLA, P., FLEMING, R. W., AND GRANIER, X. 2012. Surface flows for image-based shading design. *ACM Transactions on Graphics (Proc. SIGGRAPH) 31,* 4 (July), 94:1–94:9.

WÄCHTER, A. AND BIEGLER, L. T. 2006. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program. 106*, 25–57.

WEICKERT, J. 1999. Coherence-enhancing diffusion filtering. *International Journal of Computer Vision 31,* 2-3.

WINNEMÖLLER, H., ORZAN, A., BOISSIEUX, L., AND THOLLOT, J. 2009. Texture design and draping in 2d images. *Computer Graphics Forum (Proc. Symp. on Rendering) 28,* 4.

XU, B., CHANG, W., SHEFFER, A., BOUSSEAU, A., MCCRAE, J., AND SINGH, K. 2014. True2form: 3d curve networks from 2d sketches via selective regularization. *Transactions on Graphics (Proc. SIGGRAPH) 33,* 4.

ZHUANG, Y., ZOU, M., CARR, N., AND JU, T. 2013. A general and efficient method for finding cycles in 3d curve networks. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 32,* 6.

Appendix

We describe in this appendix our implementation of the binary-labeling problem to find a consistent orientation of directions over a surface patch. We express the image as a graph, where each pixel is a node connected to its upper and right neighbor by an edge. Each node i stores two candidate values $u_z(i)$ and $-u_z(i)$, and each edge (i, j) stores the 2×2 pairwise cost matrix of assigning each possible pair of values to the nodes i and j. We express these costs as the absolute difference between the two values. Our goal is to select the value of each node that minimize the cost over all edges, which we solve using the *Convergent Tree-reweighted Message Passing* algorithm[1] [Kolmogorov 2006]. We optionally provide users the ability to constrain one of the two values at a pixel, which the algorithm then propagates to other pixels. This feature is particularly useful to resolve the ambiguity between convex and concave surface patches, which have opposite orientations. We express these constraints as a unary penalty term that we set to 0 for the solution we want to favor and to 1 for the solution we want to penalize. We also account for these constraints in the pairwise terms that we set to 1 for the solution we want to penalize. Finally, care should be taken to properly handle the transition functions when computing the pairwise term.

[1] Implementation available at http://research.microsoft.com/en-us/downloads/dad6c31e-2c04-471f-b724-ded18bf70fe3/